# Contents

Chapter 1. Introduction .......................................................................................................................... 28

Chapter 2. Getting Started ..................................................................................................................... 29

What is Oxygen XML Editor.................................................................................................................. 29
Getting Familiar with the Interface......................................................................................................... 30
Supported Document Types.................................................................................................................. 31
Resources to Help You Get Started Using Oxygen XML Editor ...................................................... 32
Your First Document or Project ............................................................................................................ 34
  Your First XML Document.................................................................................................................. 34
  Getting Started with DITA.................................................................................................................. 39
  Creating a New Project....................................................................................................................... 47
Getting Help........................................................................................................................................... 51
Help Menu............................................................................................................................................... 51
Frequently Used Shortcut Keys............................................................................................................... 55
Accessibility Support in Oxygen........................................................................................................... 60
Oxygen XML Editor VPAT Accessibility Conformance Report .............................................................. 63

Chapter 3. Installation ............................................................................................................................ 85

Installing Oxygen XML Editor on Windows....................................................................................... 86
Installing Oxygen XML Editor on macOS.......................................................................................... 90
Installing Oxygen XML Editor on Linux............................................................................................. 92
Installing Oxygen XML Editor on Windows Server........................................................................... 97
Installing Oxygen XML Editor on a Linux / UNIX Server................................................................. 98
Site-Wide Deployment.......................................................................................................................... 101
Licensing................................................................................................................................................ 101
  Choosing a License Type.................................................................................................................... 101
  Obtaining a License Key..................................................................................................................... 102
  Register a Named-User or Subscription License............................................................................. 102
  Floating Licenses.................................................................................................................................. 103
  Transferring a License Key.................................................................................................................. 106
License Servers....................................................................................................................................... 106
  Setting up an HTTP License Server (Floating or Named-User Licenses)............................................ 107
  Common Problems: License Server Errors....................................................................................... 117
Upgrading..................................................................................................................118
Upgrading Oxygen XML Editor on Windows/Linux......................................................119
Upgrading Oxygen XML Editor on macOS..................................................................120
Installing and Updating Add-ons.................................................................................120
Uninstalling..................................................................................................................122

Chapter 4. Configuration.............................................................................................124
Preferences.................................................................................................................124
Global Preferences......................................................................................................126
Appearance Preferences..............................................................................................129
Application Layout Preferences.................................................................................135
Add-ons Preferences..................................................................................................136
Project Level Settings Preferences.............................................................................137
Document Type Association Preferences..................................................................138
Document Templates Preferences..............................................................................167
Encoding Preferences.................................................................................................168
Editor Preferences.......................................................................................................170
CSS Validator Preferences..........................................................................................234
XML Preferences........................................................................................................235
DITA Preferences.........................................................................................................269
Markdown Preferences...............................................................................................275
Data Sources Preferences...........................................................................................276
SVN Preferences..........................................................................................................281
diff Preferences...........................................................................................................286
Archive Preferences.....................................................................................................290
Plugins Preferences......................................................................................................291
External Tools Preferences.........................................................................................292
Menu Shortcut Keys Preferences..............................................................................294
File Types Preferences...............................................................................................297
Open/Find Resource Preferences Page........................................................................298
Custom Editor Variables Preferences.........................................................................300
Network Connection Settings Preferences..................................................................301
XML Structure Outline Preferences..........................................................................306
Views Preferences.......................................................................................................306
Chapter 5. Perspectives ................................................................. 344

Editor Perspective ........................................................................ 344
DITA Perspective .......................................................................... 346
XSLT Debugger Perspective .......................................................... 347
XQuery Debugger Perspective ....................................................... 349
Database Perspective .................................................................... 350

Chapter 6. Editing Modes ............................................................... 353

Text Editing Mode ....................................................................... 353
Grid Editing Mode ....................................................................... 354
Author Editing Mode .................................................................... 354
Design Editing Mode (Schema Diagram Editor) .......................... 355

Chapter 7. Working With Any Type of Document ......................... 358

Getting Familiar with the Interface ........................................... 358
Configuring the Layout of the Views and Editors ...................... 360
Configuring Toolbars .................................................................. 365
Creating, Opening, Saving, and Closing Documents

Creating New Documents and Templates

Opening Documents

Saving Documents

Auto Recover Documents

Closing Documents

Working with Remote Documents

Open URL

Changing File Permissions on a Remote FTP Server

WebDAV over HTTPS

HTTP Authentication Schemes

Switching, Moving, or Hiding Editor Tabs

Contextual Menu of the Current Editor Tab

Viewing File Properties

Simple Text Editor

Using Projects to Group Documents

Creating a New Project

Project View

Batch Validation and Transformation

Sharing a Project - Team Collaboration

Contextual Project Operations Using 'Main Files' Support

Search and Find/Replace Features

Open/Find Resource View

Open/Find Resource Dialog Box

Searching in Content

Searching in File Paths

Searching in Reviews

Find/Replace Dialog Box

Find/Replace in Multiple Files

Find All Elements Dialog Box

Find and Invoke Actions

Quick Find Toolbar

Keyboard Shortcuts for Finding the Next and Previous Match
<table>
<thead>
<tr>
<th>Contents</th>
<th>vi</th>
</tr>
</thead>
<tbody>
<tr>
<td>Regular Expressions Syntax</td>
<td>446</td>
</tr>
<tr>
<td>Spell Checking</td>
<td>447</td>
</tr>
<tr>
<td>Spell Check Dictionaries and Term Lists</td>
<td>449</td>
</tr>
<tr>
<td>Learned Words</td>
<td>456</td>
</tr>
<tr>
<td>Ignored Words (Elements)</td>
<td>456</td>
</tr>
<tr>
<td>Automatic Spell Check</td>
<td>456</td>
</tr>
<tr>
<td>Spell Check Multiple Files</td>
<td>458</td>
</tr>
<tr>
<td>AutoCorrect Misspelled Words</td>
<td>460</td>
</tr>
<tr>
<td>Add Dictionaries for the AutoCorrect Feature</td>
<td>461</td>
</tr>
<tr>
<td>Working with Special Characters and Encoding</td>
<td>462</td>
</tr>
<tr>
<td>Unicode Support</td>
<td>463</td>
</tr>
<tr>
<td>Opening and Saving Documents with Unsupported Characters</td>
<td>464</td>
</tr>
<tr>
<td>Unicode Fallback Font Support</td>
<td>464</td>
</tr>
<tr>
<td>Inserting Special Characters with the Character Map</td>
<td>465</td>
</tr>
<tr>
<td>Image Preview</td>
<td>469</td>
</tr>
<tr>
<td>Loading Large Documents</td>
<td>469</td>
</tr>
<tr>
<td>Optimize Loading for Large Files</td>
<td>470</td>
</tr>
<tr>
<td>Optimize Loading for Huge Files</td>
<td>471</td>
</tr>
<tr>
<td>Documents with Long Lines</td>
<td>471</td>
</tr>
<tr>
<td>Handling Read-Only Files</td>
<td>472</td>
</tr>
<tr>
<td>Scratch Buffer</td>
<td>472</td>
</tr>
<tr>
<td>Compare Files or Directories</td>
<td>473</td>
</tr>
<tr>
<td>Compare Files Tool</td>
<td>474</td>
</tr>
<tr>
<td>Compare Directories Tool</td>
<td>494</td>
</tr>
<tr>
<td>Compare Directories Against a Base (3-Way) Tool</td>
<td>500</td>
</tr>
<tr>
<td>Generate HTML Report for Directory Comparison</td>
<td>509</td>
</tr>
<tr>
<td>Viewing Status Information</td>
<td>512</td>
</tr>
<tr>
<td>Editor Highlights</td>
<td>512</td>
</tr>
<tr>
<td>Printing a Document</td>
<td>513</td>
</tr>
<tr>
<td><strong>Chapter 8. Editing Documents</strong></td>
<td>516</td>
</tr>
<tr>
<td>Editing XML Documents</td>
<td>516</td>
</tr>
<tr>
<td>Editing XML Documents in Text Mode</td>
<td>517</td>
</tr>
<tr>
<td>Editing XML Documents in Grid Mode</td>
<td>578</td>
</tr>
</tbody>
</table>
Editing XML Documents in Author Mode ........................................................................588
Validating XML Documents .........................................................................................777
XML Quick Fixes .............................................................................................................813
Associating a Schema to XML Documents ...................................................................816
Working with XML Catalogs ......................................................................................826
Modular Contextual XML Editing Using 'Main Files' Support .........................................829
Search and Refactoring Actions for IDs and IDREFS .....................................................830
XML Resource Hierarchy/Dependencies View ..............................................................832
Combining XML Content Using DTD Entities and XInclude .........................................835
Refactoring XML Documents ......................................................................................840
XML Digital Signatures ................................................................................................877
Editing XSLT Stylesheets .............................................................................................886
Modular Contextual XSLT Editing Using 'Main Files' Support .........................................887
Validating XSLT Stylesheets ........................................................................................887
XSLT Quick Fix Support ...............................................................................................890
Content Completion in XSLT Stylesheets ....................................................................892
Syntax Highlighting in XSLT .........................................................................................899
XSLT Outline View ......................................................................................................899
XSLT Input View ..........................................................................................................904
XSLT Resource Hierarchy/Dependencies View ............................................................906
XSLT Component Dependencies View .........................................................................909
Highlight Component Occurrences .............................................................................911
Finding XSLT References and Declarations ...............................................................911
XSLT Stylesheet Component Documentation Support ..................................................912
XSLT 3.0 Text Value Templates ...................................................................................914
XSLT 3.0 Packages (xsl:package Element) ................................................................915
XSLT Refactoring Actions ............................................................................................915
XSLT Quick Assist Support ..........................................................................................920
XSLT Unit Test (XSpec) ...............................................................................................922
Generating Documentation for an XSLT Stylesheet ....................................................925
Compiling an XSLT Stylesheet for Saxon .....................................................................932
Editing Ant Build Files ..............................................................................................934
Modular Contextual Ant Build File Editing Using 'Main Files' Support ..........................934
Validating Ant Build Files........................................................................................................ 935
Transforming Ant Build Files.................................................................................................. 936
Ant Quick Fix Support............................................................................................................... 937
Content Completion in Ant Build Files..................................................................................... 937
Syntax Highlighting in Ant Files............................................................................................... 938
Ant Outline View.................................................................................................................... 938
Ant Resource Hierarchy/Dependencies View............................................................................ 942
Ant Component Dependencies View....................................................................................... 943
Highlight Component Occurrences.......................................................................................... 944
Find References and Declarations of Ant Components.......................................................... 945
Ant Refactoring Actions........................................................................................................... 945
Ant Quick Assist Support......................................................................................................... 946

Editing XML Schemas........................................................................................................... 947
XML Schema Design Mode (XML Schema Diagram Editor)..................................................... 948
Editing XML Schema in Text Editing Mode............................................................................ 989
Modular Contextual XML Schema Editing Using ‘Main Files’ Support.................................. 989
Validating XML Schema Documents.................................................................................... 990
Quick Fixes for DTD, XSD, and Relax NG Errors................................................................... 991
Content Completion in XML Schema....................................................................................... 992
Syntax Highlighting in XML Schema....................................................................................... 992
XML Schema Outline View....................................................................................................... 993
XML Schema Attributes View.................................................................................................... 995
XML Schema Resource Hierarchy / Dependencies View......................................................... 997
XML Schema Component Dependencies View......................................................................... 1000
Highlight Component Occurrences......................................................................................... 1002
Searching and Refactoring Actions in XML Schemas............................................................. 1002
XML Schema Quick Assist Support......................................................................................... 1004
Generating Sample XML Files............................................................................................... 1006
Generating Documentation for an XML Schema.................................................................... 1011
Converting Schema to Another Schema Language.................................................................. 1021
Converting Database to XML Schema.................................................................................... 1024
Flatten an XML Schema.......................................................................................................... 1025
Generating Java Classes from XML Schema........................................................................... 1027
XML Schema Regular Expressions Builder Tool.......................................................... 1028
XML Schema 1.1.............................................................................................................. 1030
Setting the XML Schema Version.............................................................................. 1032
Editing XQuery Documents........................................................................................ 1033
XQuery Validation......................................................................................................... 1034
Content Completion in XQuery...................................................................................... 1035
Syntax Highlighting in XQuery....................................................................................... 1036
Formatting and Indenting XQuery Documents.............................................................. 1036
Folding in XQuery Documents...................................................................................... 1037
XQuery Outline View.................................................................................................... 1037
XQuery Builder View.................................................................................................... 1039
XQuery Input View........................................................................................................ 1043
Generating HTML Documentation for an XQuery Document...................................... 1045
Transforming XML Documents Using XQuery............................................................ 1046
XQuery Unit Test (XSpec).............................................................................................. 1050
Editing WSDL Documents............................................................................................ 1051
Modular Contextual WSDL Editing Using 'Main Files' Support................................. 1052
Validating WSDL Documents......................................................................................... 1053
Content Completion Assistance in WSDL Documents................................................ 1053
WSDL Syntax Highlighting.............................................................................................. 1054
WSDL Outline View....................................................................................................... 1055
WSDL Resource Hierarchy/Dependencies View in WSDL Documents........................ 1059
WSDL Component Dependencies View......................................................................... 1062
Highlight Component Occurrences in WSDL Documents........................................... 1063
Searching and Refactoring Operations in WSDL Documents...................................... 1064
Quick Assist Support in WSDL Documents.................................................................. 1066
Generating Documentation for WSDL Documents..................................................... 1067
WSDL SOAP Analyzer Tool......................................................................................... 1072
Editing CSS Stylesheets................................................................................................. 1076
Validating CSS Stylesheets......................................................................................... 1076
Content Completion in CSS Stylesheets....................................................................... 1077
Syntax Highlighting in CSS Files.................................................................................. 1078
CSS Outline View......................................................................................................... 1078
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Folding in CSS Stylesheets</td>
<td>1079</td>
</tr>
<tr>
<td>Formatting and Indenting CSS Stylesheets (Pretty Print)</td>
<td>1079</td>
</tr>
<tr>
<td>Minifying CSS Stylesheets</td>
<td>1079</td>
</tr>
<tr>
<td>Editing LESS Stylesheets</td>
<td>1080</td>
</tr>
<tr>
<td>Validating LESS Stylesheets</td>
<td>1081</td>
</tr>
<tr>
<td>Content Completion in LESS Stylesheets</td>
<td>1081</td>
</tr>
<tr>
<td>Syntax Highlighting in LESS Files</td>
<td>1082</td>
</tr>
<tr>
<td>Compiling LESS Stylesheets to CSS</td>
<td>1082</td>
</tr>
<tr>
<td>Editing Relax NG Schemas</td>
<td>1083</td>
</tr>
<tr>
<td>Modular Contextual Relax NG Schema Editing Using 'Main Files' Support</td>
<td>1083</td>
</tr>
<tr>
<td>Relax NG Schema Diagram Editor</td>
<td>1084</td>
</tr>
<tr>
<td>Validating Relax NG Schema Documents</td>
<td>1088</td>
</tr>
<tr>
<td>Content Completion in Relax NG Schemas</td>
<td>1089</td>
</tr>
<tr>
<td>Syntax Highlighting in Relax NG Schemas</td>
<td>1090</td>
</tr>
<tr>
<td>Quick Fixes for DTD, XSD, and Relax NG Errors</td>
<td>1090</td>
</tr>
<tr>
<td>Relax NG Outline View</td>
<td>1091</td>
</tr>
<tr>
<td>RNG Resource Hierarchy/Dependencies View</td>
<td>1094</td>
</tr>
<tr>
<td>Relax NG Schema Component Dependencies View</td>
<td>1097</td>
</tr>
<tr>
<td>Searching and Refactoring Actions in RNG Schemas</td>
<td>1098</td>
</tr>
<tr>
<td>RNG Quick Assist Support</td>
<td>1100</td>
</tr>
<tr>
<td>Configuring a Custom Datatype Library for a RELAX NG Schema</td>
<td>1102</td>
</tr>
<tr>
<td>Editing NVDL Schemas</td>
<td>1102</td>
</tr>
<tr>
<td>NVDL Schema Diagram</td>
<td>1102</td>
</tr>
<tr>
<td>Validating NVDL Schema Documents</td>
<td>1105</td>
</tr>
<tr>
<td>Content Completion in NVDL Schemas</td>
<td>1105</td>
</tr>
<tr>
<td>Syntax Highlighting in NVDL Schemas</td>
<td>1106</td>
</tr>
<tr>
<td>NVDL Outline View</td>
<td>1107</td>
</tr>
<tr>
<td>NVDL Schema Component Dependencies View</td>
<td>1107</td>
</tr>
<tr>
<td>Searching and Refactoring Actions in NVDL Schemas</td>
<td>1108</td>
</tr>
<tr>
<td>Editing JSON Documents</td>
<td>1110</td>
</tr>
<tr>
<td>JSON Editor</td>
<td>1110</td>
</tr>
<tr>
<td>Navigating References in JSON Documents</td>
<td>1113</td>
</tr>
<tr>
<td>Validating JSON Documents</td>
<td>1113</td>
</tr>
</tbody>
</table>
Content Completion Assistant in JSON................................................................. 1126
Syntax Highlighting in JSON Documents.......................................................... 1128
Folding in JSON................................................................................................. 1128
JSON Outline View............................................................................................. 1128
JSON to XML Converter..................................................................................... 1131
XML to JSON Converter..................................................................................... 1134
JSON to YAML Converter.................................................................................. 1137
YAML to JSON Converter.................................................................................. 1137
Contextual Menu Actions in JSON Documents.................................................. 1138
Transforming and Querying JSON Documents................................................... 1143
Editing JSON Schema Documents...................................................................... 1146
JSON Schema Editor.......................................................................................... 1147
JSON Schema Design Mode (JSON Schema Diagram Editor)............................. 1149
Generating JSON Schema from a JSON File...................................................... 1165
Generating JSON Schema Documentation....................................................... 1167
Generating Sample JSON Files from a JSON Schema....................................... 1170
XSD to JSON Schema Converter......................................................................... 1172
Validating JSON Schema Documents............................................................... 1176
Syntax Highlighting in JSON Schema Documents............................................. 1177
Editing YAML Documents.................................................................................. 1177
Validating YAML Documents............................................................................ 1178
Syntax Highlighting in YAML Documents.......................................................... 1183
Editing StratML Documents................................................................................ 1183
Editing XLIFF Documents................................................................................... 1184
Editing JavaScript Documents............................................................................ 1184
JavaScript Editing Actions.................................................................................. 1185
Validating JavaScript Files.................................................................................. 1187
Content Completion in JavaScript Documents.................................................. 1187
Syntax Highlighting in JavaScript Documents.................................................... 1188
JavaScript Outline View...................................................................................... 1189
Editing XProc Scripts........................................................................................ 1190
Editing Schematron Schemas............................................................................ 1192
Examples of Schematron Rules and Quick Fixes.............................................. 1194
Querying HTML Documents with XPath................................................................. 1261
Associating a CSS with an HTML Document.................................................. 1262
Editing Markdown Documents............................................................................ 1262
Markdown Editor................................................................................................. 1263
Creating New Markdown Documents............................................................... 1265
Actions Available in the Markdown Editor...................................................... 1266
Syntax Highlighting in the Markdown Editor.................................................... 1274
Automatic Validation in Markdown Documents.............................................. 1274
Working with Markdown Documents in DITA.................................................. 1275
Markdown Editor Syntax Rules and Specifications.......................................... 1277

Chapter 9. Built-in XML Frameworks (Document Types)................................. 1291
DocBook 4 Document Type (Framework)......................................................... 1291
DocBook 4 Author Mode Actions....................................................................... 1293
Inserting an Olink in DocBook Documents...................................................... 1309
DocBook 5 Document Type (Framework).......................................................... 1312
DocBook 5 Author Mode Actions....................................................................... 1314
Inserting an Olink in DocBook Documents...................................................... 1331
DocBook 5.1 Assembly....................................................................................... 1334
DocBook 5.1 Topic.............................................................................................. 1335
DocBook Targetset Document Type (Framework)............................................ 1336
DITA Topics Document Type (Framework)....................................................... 1337
DITA Topic Author Mode Actions...................................................................... 1338
DITA Map Document Type (Framework).......................................................... 1361
DITA Map Author Mode Actions....................................................................... 1362
XHTML Document Type (Framework).............................................................. 1374
XHTML Validation............................................................................................. 1375
XHTML Author Mode Actions......................................................................... 1375
TEI P5 Document Type (Framework)............................................................... 1388
TEI P5 Author Mode Actions............................................................................ 1389
Customization of TEI Frameworks Using the Latest Sources.......................... 1400
Customization of TEI Frameworks Using the Compiled Sources....................... 1401
TEI ODD Document Type (Framework)............................................................ 1401
TEI ODD Author Mode Actions....................................................................... 1402
<table>
<thead>
<tr>
<th>Chapter 10. Additional XML Editing Frameworks (Document Types)</th>
<th>1430</th>
</tr>
</thead>
<tbody>
<tr>
<td>S1000D Document Type (Framework)</td>
<td>1430</td>
</tr>
<tr>
<td>OpenAPI (Swagger) Document Type (Framework)</td>
<td>1431</td>
</tr>
<tr>
<td>Chapter 11. Publishing</td>
<td>1432</td>
</tr>
<tr>
<td>Transformation Scenarios</td>
<td>1432</td>
</tr>
<tr>
<td>Built-in Transformation Scenarios</td>
<td>1433</td>
</tr>
<tr>
<td>Creating New Transformation Scenarios</td>
<td>1464</td>
</tr>
<tr>
<td>Editing a Transformation Scenario</td>
<td>1545</td>
</tr>
<tr>
<td>Duplicating a Transformation Scenario</td>
<td>1547</td>
</tr>
<tr>
<td>Applying Associated Transformation Scenarios</td>
<td>1548</td>
</tr>
<tr>
<td>Configure Transformation Scenario(s) Dialog Box</td>
<td>1548</td>
</tr>
<tr>
<td>Batch Transformations</td>
<td>1553</td>
</tr>
<tr>
<td>Sharing Transformation Scenarios</td>
<td>1555</td>
</tr>
<tr>
<td>Transformation Scenarios View</td>
<td>1555</td>
</tr>
<tr>
<td>WebHelp Output Customization</td>
<td>1559</td>
</tr>
<tr>
<td>WebHelp Responsive Output for DITA</td>
<td>1560</td>
</tr>
<tr>
<td>WebHelp Classic Output for DocBook</td>
<td>1734</td>
</tr>
<tr>
<td>DITA to PDF Output Customization</td>
<td>1767</td>
</tr>
<tr>
<td>CSS-based PDF Customization</td>
<td>1768</td>
</tr>
<tr>
<td>XSL-FO to PDF Customization</td>
<td>1986</td>
</tr>
<tr>
<td>DocBook to PDF Output Customization</td>
<td>1997</td>
</tr>
<tr>
<td>Chapter 12. Working with XPath Expressions</td>
<td>1999</td>
</tr>
<tr>
<td>XPath Toolbar</td>
<td>2000</td>
</tr>
<tr>
<td>XPath Builder View</td>
<td>2002</td>
</tr>
<tr>
<td>XPath Expression Results View</td>
<td>2005</td>
</tr>
<tr>
<td>XPath and XML Catalogs</td>
<td>2007</td>
</tr>
<tr>
<td>XPath Prefix Mapping</td>
<td>2007</td>
</tr>
<tr>
<td>Chapter 13. Working with Archives</td>
<td>2008</td>
</tr>
</tbody>
</table>

- JTEI Document Type (Framework) .......................................................... 1413
- JATS Document Type (Framework) ............................................................ 1414
- JATS Author Mode Actions ................................................................. 1415
- EPUB Document Type (Framework) ......................................................... 1426
- Other Supported Document Types ......................................................... 1428
<table>
<thead>
<tr>
<th>Contents</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>General Configuration of an Oxygen XML Editor Plugin</td>
<td>2411</td>
</tr>
<tr>
<td>Installing an Oxygen XML Editor Plugin</td>
<td>2413</td>
</tr>
<tr>
<td>Types of Plugin Extensions Available with the SDK</td>
<td>2414</td>
</tr>
<tr>
<td>How to Write a CMS Integration Plugin</td>
<td>2439</td>
</tr>
<tr>
<td>How to Write A Custom Protocol Plugin</td>
<td>2444</td>
</tr>
<tr>
<td>How to Share a Class Loader Between a Framework and Plugin</td>
<td>2444</td>
</tr>
<tr>
<td>Packing and Deploying Plugins as Add-ons</td>
<td>2445</td>
</tr>
<tr>
<td>Testing Plugins and Java Extensions</td>
<td>2446</td>
</tr>
<tr>
<td>Disabling a Plugin</td>
<td>2451</td>
</tr>
<tr>
<td><strong>Oxygen XML Author Component</strong></td>
<td>2451</td>
</tr>
<tr>
<td>Licensing</td>
<td>2451</td>
</tr>
<tr>
<td>Installation Requirements</td>
<td>2452</td>
</tr>
<tr>
<td>Customization</td>
<td>2452</td>
</tr>
<tr>
<td>Adding MathML support in the Oxygen XML Author Component</td>
<td>2455</td>
</tr>
<tr>
<td>Adding Support to Insert References from a WebDAV Connection</td>
<td>2457</td>
</tr>
<tr>
<td>Using Plugins with the Oxygen XML Author Component</td>
<td>2457</td>
</tr>
<tr>
<td>Frequently Asked Questions</td>
<td>2458</td>
</tr>
<tr>
<td><strong>Oxygen XML Web Author Component</strong></td>
<td>2461</td>
</tr>
<tr>
<td>Web Author vs. Web Author Component</td>
<td>2461</td>
</tr>
<tr>
<td><strong>Developer Quick Start Guide</strong></td>
<td>2464</td>
</tr>
<tr>
<td>Plugins</td>
<td>2464</td>
</tr>
<tr>
<td>Frameworks</td>
<td>2470</td>
</tr>
<tr>
<td>Difference Between a Framework (Document Type) and a Plugin Extension</td>
<td>2475</td>
</tr>
<tr>
<td><strong>SDK Common Use Cases</strong></td>
<td>2476</td>
</tr>
<tr>
<td>Add Custom Actions to the Contextual Menu</td>
<td>2476</td>
</tr>
<tr>
<td>Add Custom Callouts</td>
<td>2478</td>
</tr>
<tr>
<td>Add Custom Highlights to Content</td>
<td>2484</td>
</tr>
<tr>
<td>Auto-Generate an ID When a Document is Opened or Created</td>
<td>2484</td>
</tr>
<tr>
<td>Change the Default Track Changes (Review) Author Name</td>
<td>2486</td>
</tr>
<tr>
<td>Change the DOCTYPE of an Open XML Document</td>
<td>2486</td>
</tr>
<tr>
<td>Control XML Serialization in the Oxygen XML Author Component</td>
<td>2487</td>
</tr>
<tr>
<td>Customize the Outline View in Text Mode</td>
<td>2488</td>
</tr>
<tr>
<td>Disable Context-Sensitive Menu Items for Custom Author Actions</td>
<td>2494</td>
</tr>
</tbody>
</table>
Dynamically Add Form Controls Using a Styles Filter ................................................................. 2495
Dynamically Modify the Content Inserted by the Author ................................................................. 2497
Extend the Java Functionality of an Existing Framework (Document Type) ........................................ 2499
Impose Custom Options for Authors .................................................................................................. 2500
Insert an Element with all the Required Content .............................................................................. 2500
Modify the XML Content on Open .................................................................................................... 2501
Modify the XML Content on Save ...................................................................................................... 2503
Multiple Rendering Modes for the Same Document in Author Mode ................................................ 2504
Obtain the Currently Selected Element Using the Author API ........................................................ 2505
Open a Document from Another Application ..................................................................................... 2506
Run XSLT or XQuery Transformations .............................................................................................. 2506
Save a New Document with a Predefined File Name Pattern ............................................................ 2507
Split Paragraph on Enter (Instead of Showing Content Completion List) ........................................ 2508
Use Custom Rendering Styles for Entity References, Comments, or PIIs ......................................... 2509

Chapter 19. Add-ons ....................................................................................................................... 2514

Collaboration ..................................................................................................................................... 2515
Git Client Add-on ............................................................................................................................... 2515
Content Fusion Connector Add-on ...................................................................................................... 2527
Integrating Oxygen Feedback with Oxygen XML Editor/Author ..................................................... 2527
Migration/Conversions ....................................................................................................................... 2533
Batch Documents Converter Add-on ................................................................................................. 2533
DITA Editing ....................................................................................................................................... 2538
DITA Prolog Updater Add-on ............................................................................................................... 2538
DITA References View Add-on ............................................................................................................ 2540
Terminology ....................................................................................................................................... 2541
Terminology Checker Add-on ............................................................................................................ 2541
Vale Linter for Markdown and HTML Validation Add-on ................................................................. 2546
Translation .......................................................................................................................................... 2547
Translator Helper Add-on .................................................................................................................... 2547
DITA Translation Package Builder Add-on .......................................................................................... 2548
Productivity ......................................................................................................................................... 2550
Smart Autocomplete Add-on (Experimental) .................................................................................... 2550
Oxygen Emmet Plugin ......................................................................................................................... 2557
Writer Helper Add-on (Experimental)..................................................................................2561
Development..........................................................................................................................2562
XSpec Helper View Add-on.....................................................................................................2562
Saxon XSLT and XQuery Transformer Add-on........................................................................2563
ICU4J Library Add-on............................................................................................................2564
XSD to JSON Schema Converter............................................................................................2564
Generating Java Classes from XML Schema.........................................................................2568
Generating JSON Schema Documentation............................................................................2569
OpenAPI Tester.......................................................................................................................2572
Generating OpenAPI Documentation......................................................................................2575
Others........................................................................................................................................2578
DocBook Checker Add-on........................................................................................................2578
CGM Image Support Add-on....................................................................................................2579

Chapter 20. Tools....................................................................................................................2580

XML Refactoring....................................................................................................................2580
Built-in Refactoring Operations...............................................................................................2583
Custom Refactoring Operations...............................................................................................2596
Storing and Sharing Refactoring Operations...........................................................................2610
Localizing XML Refactoring Operations..................................................................................2611
Generate Sample XML Files.......................................................................................................2612
Generate/Convert Schema........................................................................................................2618
Convert DB Structure to XML Schema....................................................................................2620
Flatten Schema.......................................................................................................................2621
Generate Java Classes from XSD...............................................................................................2623
Compile XSL Stylesheet for Saxon...........................................................................................2625

JSON Tools.............................................................................................................................2627
Generate Sample JSON Files......................................................................................................2627
Generate JSON Schema............................................................................................................2629
JSON to YAML..........................................................................................................................2631
YAML to JSON..........................................................................................................................2632
JSON to XML............................................................................................................................2632
XML to JSON............................................................................................................................2636
XSD to JSON Schema................................................................................................................2639
Chapter 21. Troubleshooting ............................................................... 2855

Performance Problems and Solutions ............................................ 2855
  Display Problems on Linux or Solaris ........................................ 2855
  Out of Memory on External Processes ........................................ 2855
  Too many nested apply-templates calls Error When Running a Transformation ........................................ 2856
  Performance Issues with Large Documents ................................ 2856
  Performance Issues when Using Oxygen XML Editor with Remote Desktop ........................................ 2857

Misc Problems and Solutions .......................................................... 2857
  Address Family Not Supported by Protocol Family ....................... 2858
  Application Reports Errors During Startup After Installing a New Version ........................................ 2858
  Application Takes Several Minutes to Start .............................. 2859
  Archive Distribution Fails to Run on macOS 10.12 (Sierra) ........ 2859
  Cannot Connect to SVN Repository from Repositories View .......... 2860
  Cannot Uninstall Oxygen XML Editor in Windows ...................... 2861
  Compatibility Issue Between Java and Certain Graphics Card Drivers ........................................ 2862
  Crash at Startup on Windows with an Error About the nvglv32.dll File ........................................ 2862
  Crash in Java Runtime Thread .................................................... 2862
  Damaged File Associations on macOS ......................................... 2864
  Details to Submit in a Request for Technical Support Using the Online Form ........................................ 2864
  Dialog Boxes Cannot Be Resized on Mac .................................... 2866
  DITA Map Transformation Fails (Cannot Connect to External Location) ........................................ 2866
  DITA Map WebHelp Transformation Fails (Duplicate Topic References Found) ........................................ 2866
  DITA-OT Transformation Takes a Long Time to Process ............... 2867
  DITA PDF Transformation Fails .................................................. 2868
  DITA PDF Processing Common Errors .......................................... 2869
  DITA to CHM Transformation Fails - Cannot Open File ................. 2870
<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>DITA to CHM Transformation Fails - Compilation Failed</td>
<td>2871</td>
</tr>
<tr>
<td>Handshake Failure Error When Accessing an HTTPS (SSL) Resource</td>
<td>2872</td>
</tr>
<tr>
<td>Hunspell Spell Checker is Unusable on Your Platform Error</td>
<td>2872</td>
</tr>
<tr>
<td>High Resolution Scaling Issues</td>
<td>2873</td>
</tr>
<tr>
<td>High Resolution Scaling Issues on Linux</td>
<td>2873</td>
</tr>
<tr>
<td>Images Appear Stretched Out in the PDF Output</td>
<td>2874</td>
</tr>
<tr>
<td>Increasing the Memory for the Ant Process</td>
<td>2875</td>
</tr>
<tr>
<td>Java Virtual Machine (JVM) Crash on macOS</td>
<td>2875</td>
</tr>
<tr>
<td>JPEG CMYK Color Space Issues</td>
<td>2875</td>
</tr>
<tr>
<td>Keyboard Language Resets to Default on Windows</td>
<td>2876</td>
</tr>
<tr>
<td>Keyboard Shortcuts Do Not Work At All</td>
<td>2876</td>
</tr>
<tr>
<td>Keyboard Shortcuts Result in Unexpected Behavior</td>
<td>2876</td>
</tr>
<tr>
<td>Mac Touch Bar Function Keys Do Not Work</td>
<td>2877</td>
</tr>
<tr>
<td>Keyboard no Longer Works After Inserting Accented Characters on Mac</td>
<td>2877</td>
</tr>
<tr>
<td>Server Signature Mismatch Error</td>
<td>2878</td>
</tr>
<tr>
<td>MSXML 4.0 Transformation Issues</td>
<td>2879</td>
</tr>
<tr>
<td>Navigation to a Web Page is Canceled when Viewing CHM on a Network Drive</td>
<td>2879</td>
</tr>
<tr>
<td>Out Of Memory Error When Opening Large Documents</td>
<td>2880</td>
</tr>
<tr>
<td>Problem With Shortcut</td>
<td>2880</td>
</tr>
<tr>
<td>Rectangular Selection Feature Does Not Work On Windows</td>
<td>2881</td>
</tr>
<tr>
<td>References Outside the Main DITA Map Folder</td>
<td>2881</td>
</tr>
<tr>
<td>Saxon 9.7 Transformer Issues</td>
<td>2882</td>
</tr>
<tr>
<td>Scroll Function of my Notebook Trackpad is Not Working</td>
<td>2882</td>
</tr>
<tr>
<td>Special Characters are Replaced with a Square</td>
<td>2883</td>
</tr>
<tr>
<td>TocJS Transformation Does not Generate All Files for a Tree-Like TOC</td>
<td>2883</td>
</tr>
<tr>
<td>Text Rendering Issues on macOS</td>
<td>2884</td>
</tr>
<tr>
<td>Wrong Words are Highlighted when Searching in a User Manual</td>
<td>2884</td>
</tr>
<tr>
<td>XML Document Takes a Long Time to Open</td>
<td>2885</td>
</tr>
<tr>
<td>XSLT Debugger Is Very Slow</td>
<td>2885</td>
</tr>
</tbody>
</table>

**Chapter 22. DITA Authoring** ................................................................. 2886

<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Getting Started with DITA</td>
<td>2887</td>
</tr>
<tr>
<td>Working with Projects in DITA</td>
<td>2894</td>
</tr>
<tr>
<td>Working with DITA Maps</td>
<td>2895</td>
</tr>
<tr>
<td>Title</td>
<td>Page</td>
</tr>
<tr>
<td>----------------------------------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>DITA Maps Manager.</td>
<td>2897</td>
</tr>
<tr>
<td>Creating a Map.</td>
<td>2913</td>
</tr>
<tr>
<td>Managing DITA Maps.</td>
<td>2916</td>
</tr>
<tr>
<td>Chunking DITA Topics.</td>
<td>2941</td>
</tr>
<tr>
<td>DITA Map Validation and Completeness Check.</td>
<td>2941</td>
</tr>
<tr>
<td>DITA Map Author Mode Actions.</td>
<td>2947</td>
</tr>
<tr>
<td>Opening a DITA Map With Topic Content Resolved.</td>
<td>2958</td>
</tr>
<tr>
<td>Working with DITA Topics.</td>
<td>2959</td>
</tr>
<tr>
<td>Creating a New DITA Topic.</td>
<td>2961</td>
</tr>
<tr>
<td>Fast Create Multiple DITA Topics.</td>
<td>2964</td>
</tr>
<tr>
<td>Editing DITA Topics.</td>
<td>2967</td>
</tr>
<tr>
<td>Converting DITA Topics to Another Type.</td>
<td>2970</td>
</tr>
<tr>
<td>Working with Images in DITA Topics.</td>
<td>2973</td>
</tr>
<tr>
<td>Adding Video, Audio, and Embedded HTML Resources in DITA Topics.</td>
<td>2977</td>
</tr>
<tr>
<td>Image Maps in DITA.</td>
<td>2980</td>
</tr>
<tr>
<td>Adding Tables in DITA Topics.</td>
<td>2986</td>
</tr>
<tr>
<td>Adding MathML Equations in DITA Topics.</td>
<td>3000</td>
</tr>
<tr>
<td>Adding LaTeX Equations in DITA Topics.</td>
<td>3001</td>
</tr>
<tr>
<td>DITA Topic Author Mode Actions.</td>
<td>3001</td>
</tr>
<tr>
<td>Working with Markdown Documents in DITA.</td>
<td>3024</td>
</tr>
<tr>
<td>Working with Keys in DITA.</td>
<td>3026</td>
</tr>
<tr>
<td>Working with a Glossary of Terms in DITA.</td>
<td>3028</td>
</tr>
<tr>
<td>Reusing DITA Content.</td>
<td>3031</td>
</tr>
<tr>
<td>Reusing DITA Topics in Multiple Maps.</td>
<td>3034</td>
</tr>
<tr>
<td>Working with Content References.</td>
<td>3035</td>
</tr>
<tr>
<td>Working with Code References.</td>
<td>3051</td>
</tr>
<tr>
<td>Working with the Conref Push Mechanism.</td>
<td>3052</td>
</tr>
<tr>
<td>Working with Reusable Components.</td>
<td>3054</td>
</tr>
<tr>
<td>Working with Variable Text in DITA.</td>
<td>3056</td>
</tr>
<tr>
<td>Working with DITA 1.3 Key Scopes.</td>
<td>3058</td>
</tr>
<tr>
<td>Working with DITA 1.3 Branch Filtering.</td>
<td>3060</td>
</tr>
<tr>
<td>DITA Reusable Components View.</td>
<td>3061</td>
</tr>
<tr>
<td>Linking in DITA.</td>
<td>3071</td>
</tr>
<tr>
<td>Topic</td>
<td>Page</td>
</tr>
<tr>
<td>----------------------------------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>Hierarchical Linking in DITA Maps</td>
<td>3072</td>
</tr>
<tr>
<td>Linking in DITA Topics</td>
<td>3072</td>
</tr>
<tr>
<td>Linking with Relationship Tables in DITA</td>
<td>3078</td>
</tr>
<tr>
<td>Publishing DITA Output</td>
<td>3080</td>
</tr>
<tr>
<td>Built-in DITA Map Transformation Scenarios</td>
<td>3081</td>
</tr>
<tr>
<td>Built-in DITA Topic Transformation Scenarios</td>
<td>3103</td>
</tr>
<tr>
<td>Running a DITA Transformation Scenario</td>
<td>3104</td>
</tr>
<tr>
<td>Creating or Editing a DITA-OT Transformation</td>
<td>3104</td>
</tr>
<tr>
<td>Customizing DITA Transformations</td>
<td>3120</td>
</tr>
<tr>
<td>Publishing with a DITA-OT Project File</td>
<td>3124</td>
</tr>
<tr>
<td>Dynamic Word/Excel/HTML/Markdown to DITA Converter</td>
<td>3125</td>
</tr>
<tr>
<td>Troubleshooting DITA Transformation Problems</td>
<td>3126</td>
</tr>
<tr>
<td>DITA Profiling / Conditional Text</td>
<td>3133</td>
</tr>
<tr>
<td>Creating and Editing Profiling Attributes in DITA</td>
<td>3134</td>
</tr>
<tr>
<td>Applying Profiling Attributes in DITA</td>
<td>3137</td>
</tr>
<tr>
<td>Creating and Editing Profiling Condition Sets in DITA</td>
<td>3140</td>
</tr>
<tr>
<td>Applying Profiling Condition Sets in DITA</td>
<td>3142</td>
</tr>
<tr>
<td>Showing and Filtering Profiled Content in DITA</td>
<td>3144</td>
</tr>
<tr>
<td>Customizing Colors and Styles for Rendering Profiling in Author Mode</td>
<td>3147</td>
</tr>
<tr>
<td>Conditional Profiling Attribute Groups</td>
<td>3149</td>
</tr>
<tr>
<td>Customizing Profiling Values with a Subject Scheme Map</td>
<td>3151</td>
</tr>
<tr>
<td>Filtering Profiling Values with a DITAVAL File</td>
<td>3155</td>
</tr>
<tr>
<td>Styling the Rendering of Profiled Content Using a DITAVAL File</td>
<td>3158</td>
</tr>
<tr>
<td>Publishing Profiled DITA Content</td>
<td>3159</td>
</tr>
<tr>
<td>Conditional Processing to Generate Multiple Deliverables</td>
<td>3159</td>
</tr>
<tr>
<td>DITA Open Toolkit Support</td>
<td>3160</td>
</tr>
<tr>
<td>DITA-OT Plugins</td>
<td>3160</td>
</tr>
<tr>
<td>Using an External DITA Open Toolkit in Oxygen XML Editor</td>
<td>3170</td>
</tr>
<tr>
<td>DITA Open Toolkit Project</td>
<td>3171</td>
</tr>
<tr>
<td>DITA Specialization Support</td>
<td>3176</td>
</tr>
<tr>
<td>Integrating a DITA Specialization</td>
<td>3176</td>
</tr>
<tr>
<td>Editing DITA Map Specializations</td>
<td>3178</td>
</tr>
<tr>
<td>Editing DITA Topic Specializations</td>
<td>3179</td>
</tr>
</tbody>
</table>
Translating DITA Projects Overview...............................................................3179
Main Files Support in DITA............................................................................3181
DITA Resource Hierarchy/Dependencies View..............................................3183
Search and Rename Actions for IDs in DITA..................................................3186
Metadata.........................................................................................................3187
Migrating MS Office Documents to DITA.......................................................3188
Migrating Various Document Formats to and from DITA...............................3190
How to Count Words in DITA Topics or Maps...............................................3192
DITA 1.3 Support...........................................................................................3192
DITA 2.0 Support...........................................................................................3194

Chapter 23. Scripting Oxygen........................................................................3196
DITA Validate and Check For Completeness....................................................3196
Transform.......................................................................................................3197
XML Refactoring.............................................................................................3198
DITA Translation Package Builder.................................................................3199
Batch Converter.............................................................................................3201
Compile Framework Script..............................................................................3202
XSLT Stylesheets Documentation.................................................................3203
XML Schema Documentation.........................................................................3203
WSDL Documentation....................................................................................3204
XML Instance Generator................................................................................3204
Flatten XML Schema....................................................................................3205
Compare Directories......................................................................................3206
Compare Files................................................................................................3210
Format and Indent Files..................................................................................3215

Chapter 24. Glossary......................................................................................3216
Active Cell.......................................................................................................3216
Alternate CSS Style.......................................................................................3216
Anchor.............................................................................................................3216
Apache Ant.....................................................................................................3216
Block Element...............................................................................................3216
Bookmark......................................................................................................3216
Callout.............................................................................................................3217
Working Set.......................................................................................................................... 3224
XML Catalog...................................................................................................................... 3225
Index..................................................................................................................................... a
Copyright............................................................................................................................ bm
1. Introduction


Oxygen XML Editor is a cross-platform application designed to accommodate all of your XML editing, authoring, developing, and publishing needs. It is the best XML editor available for document development using structured mark-up languages such as XML, XSD, Relax NG, XSL, DTD. It is a comprehensive solution for authors who want to edit XML documents visually, with or without extensive knowledge about XML and XML-related technologies. The WYSIWYG-like editor is driven by CSS stylesheets associated with the XML documents and offers many innovative, user-friendly authoring features that make XML authoring easy and powerful.

It offers developers and authors a powerful Integrated Development Environment and the intuitive Graphical User Interface of Oxygen XML Editor is easy to use and provides robust functionality for content editing, project management, and validation of structured mark-up sources. Coupled with XSLT and FOP transformation technologies, Oxygen XML Editor offers support for generating output to multiple target formats, including: PDF, PS, TXT, HTML, JavaHelp, WebHelp, and XML.

This user guide is focused on describing features, functionality, the application interface, and to help you quickly get started. It also includes a vast amount of advanced technical information and instructional topics that are designed to teach you how to use Oxygen XML Editor to accomplish your tasks. It is assumed that you are familiar with the use of your operating system and the concepts related to XML technologies and structured mark-up.
2. Getting Started

This section provides a variety of resources to help you get the most out of the application. Typically, the first step of getting started with Oxygen XML Editor would be to install the software. For detailed information about that process, see the Installation chapter (on page 85).

After installation, when you launch Oxygen XML Editor for the first time, you are greeted with a **Welcome** dialog box. It presents upcoming events, useful video demonstrations, helpful resources, the tip of the day, and also gives you easy access to recently used files and projects and to create new ones.

![Figure 1. Welcome Dialog Box](image)

If you do not want it to be displayed every time you launch Oxygen XML Editor, deselect the **Show at startup** option in the bottom-left corner of the dialog box. To display it any time, go to **Help > Welcome**.

**What is Oxygen XML Editor**

Oxygen XML Editor is the best XML editor available and is a complete XML development and authoring solution. It is designed to accommodate a large number of users, ranging from beginners to XML experts. It is the only XML tool that supports all of the XML schema languages and provides a large variety of powerful tools for editing and publishing XML documents.
You can use Oxygen XML Editor to work with most XML-based standards and technologies. It is a cross-platform application available on all the major operating systems (Windows, macOS, Linux, Solaris) and can be used either as a standalone application or as an Eclipse plugin.

For a list of many of the features and technologies that are included in Oxygen XML Editor, see the Oxygen Website.

**Getting Familiar with the Interface**

Oxygen XML Editor includes several perspectives (on page 3221) and editing modes (on page 353) to help you accomplish a wide range of tasks. Each perspective and editing mode also includes a large variety of helper views, menu actions, toolbars, and contextual menu functions.

There are various ways that you can configure the layout of the views or editors (on page 360), and you can customize the toolbars (on page 365).

Regardless of the perspective (on page 3221) or editing mode (on page 353) that you are working with, the default layout consists of the following areas:

**Menus**

Menu-driven access to all the features and functions available in Oxygen XML Editor. Most of the menus are common for all types of documents, but Oxygen XML Editor also includes some context-sensitive and framework-specific menus and actions that are only available for a specific context or type of document.

**Toolbars**

Easy access to common and frequently used functions. Each icon is a button that acts as a shortcut to a related function. Some of the toolbars are common for all perspectives, editing modes, and types of documents, while others are specific to the particular perspective or mode. Some toolbars are also framework-specific, depending on the type of document that is being edited. All the toolbars can be configured (on page 365) to suit your specific needs.

**Helper Views**

Oxygen XML Editor includes a large variety of dockable (on page 3218) views to assist you with editing, viewing, searching, validating, transforming, and organizing your documents. Many of the views also contain useful contextual menu actions, toolbar buttons, or menus. The most commonly used views for each perspective and editing mode are displayed by default and you can choose to display others to suit your specific needs. The layout of the views can also be configured (on page 360) according to your preferences.

**Editor Pane**

The main editing area in the center of the application. Each editing mode (on page 353) provides a main editor pane where you spend most of your time reading, editing, applying markup, and validating your documents. The editor pane in each editing mode (on page 353) also includes various contextual menu actions and other features to help streamline your editing...
tasks. Each file that has been opened has a tab at the top of the editing pane and there are several ways to switch between tabs or move them (on page 393).

**Perspectives**

Oxygen XML Editor includes several different perspectives (on page 344) that you can use to work with your documents. The Editor perspective is the most commonly used perspective used for displaying and editing the content of your XML documents, and it is the default perspective when you start Oxygen XML Editor for the first time. Oxygen XML Editor also includes a Database perspective that allows you to manage databases and their connections and a few debugging perspectives that allow you to detect problems in XSLT or XQuery transformations.

**Status Bar**

The status bar at the bottom of the application contains some useful information when you are working with documents. It includes the following information, in the order it is displayed from left to right:

- The path of the current document.
- The Unicode value (on page 463) for the character directly to the right of the current cursor position.
- The status of the current document. The status of Modified is displayed for documents that have not yet been saved. Otherwise, this section is left blank.
- In Text editing mode (on page 353), the current line and character position is displayed.
- If the Check for notifications option (on page 126) is selected, this section will show you when new messages have been received. The types of messages include the addition of new videos on the Oxygen XML Editor website, the announcement of upcoming webinars and conferences where the Oxygen XML Editor team will participate, and more.
- The memory consumption, including the memory used by the application and the maximum amount that is allocated to the application.
- If the Show memory status option (on page 128) is selected, a Free unused memory icon is displayed in the bottom-right corner and you can use this icon to free up unused memory.

**Supported Document Types**

You can use the main editing pane in Oxygen XML Editor to edit a large variety of document types.

The supported document types include the following:

- XML documents
- XSLT stylesheets
- XML Schema
- DTD (Document Type Definition) schemas
- RELAX NG full syntax schemas
- RELAX NG compact syntax schemas
Resources to Help You Get Started Using Oxygen XML Editor

Configuring Oxygen XML Editor

There are numerous ways that you can configure Oxygen XML Editor to accommodate your specific needs.

See the Configuring Oxygen section (on page 124) for details on the various ways that you can configure the application and its features.

Video Tutorials and Webinars

The Oxygen XML Editor website includes numerous video demonstrations and webinars that present many of the features that are available in Oxygen XML Editor and show you how to complete specific tasks or how to use the various features.

Go to the Oxygen Videos page to see the list of video tutorials.

Go to the Oxygen Events page to see all the upcoming and past webinars, conferences, and other events.

Oxygen XML Editor Documentation

The Oxygen XML Editor documentation includes a plethora of sections and topics to provide you with a variety of information, ranging from basic authoring tasks to advanced developer techniques. You can, of course, search through the documentation using standard search mechanisms, but you can also place the cursor in any particular position in the interface and use the F1 key to open a dialog box that presents a section in the documentation that is appropriate for the context of the current cursor position. Aside from the other topics in this Getting Started section, the following are links to other sections of the documentation that might be helpful for your specific needs:
• **Text Editing Mode Section (on page 353)** - Provides information about the **Text** editor.

• **Author Editing Mode Section (on page 354)** - Provides information about the visual WYSIWYG-like **Author** editing mode.

• **XML Schema Diagram Editor (on page 355)** - Provides information about the schema design mode.

• **Editing Specific Document Types Chapter (on page 516)** - Includes information about editing numerous different types of documents.

• **DITA Authoring Chapter (on page 2886)** - Provides information about using DITA to edit and structure your content.

• **Publishing Chapter (on page 1432)** - Provides information about the various ways that you can publish content.

• **Importing Data Chapter (on page 2091)** - Provides information about importing data from text files, MS Excel files, database data, and HTML files.

• **Tools Chapter (on page 2580)** - Details about the various built-in tools that are available in Oxygen XML Editor.

• **Add-ons Chapter (on page 2514)** - Information about how to extend the functionality of Oxygen XML Editor through add-ons.

### Sample Documents

Your installation of Oxygen XML Editor includes a large variety of sample documents and projects that you can use as templates to get started and to experiment with the various features and technologies. They are located in the **samples** folder that is located in the installation directory of Oxygen XML Editor. You will find files and folders for various types of documents, including the following:

• **Sample project file (**sample.xpr**)** - A sample project file that will allow you to experiment with how projects can be structured and used. When you open this project file, you will be able to see all the sample files and folders in the **Project view (on page 402)**.

• **Sample files (**personal.xml, etc.**)** - A collection of interrelated sample files that will allow you to experiment with the structure and relationship between XML files, stylesheets, and schemas.

• **Various document type folders** - The various folders contain sample files for numerous document types, such as CSS, DITA, DocBook, ePub, TEI, XHTML, and many others.

### Other Resources

The following list includes links to various other resources that will help you get started using the features of Oxygen XML Editor:

• See the **Oxygen XML Editor Blog Site** for a large variety of current and archived blogs regarding numerous features, requests, and instructional topics.

• Take advantage of the **Oxygen XML Editor Forum** to see various announcements and learn more about specific issues that other users have experienced.

• If you are using DITA, see the incredibly helpful **DITA Style Guide Best Practices for Authors**.

• To learn about the WebHelp features in Oxygen XML Editor, see the **Publishing DITA and DocBook to WebHelp** section of the website.
• For more information about various additional tools that are integrated into Oxygen XML Editor, see the Tools section (on page 2580).
• See the External Resource Page for links to various other helpful resources, such as discussion lists, external tutorials, and more.
• See the Oxygen SDK section for details about the SDK that allows you to extend and develop Oxygen XML Editor frameworks (on page 3219) and plugins (on page 3222), and to integrate Eclipse plugins.
• For a list of new features that were implemented in the latest version of Oxygen XML Editor, see the What's New Section of the Website.
• You can select the Tip of the Day (on page 53) action in the Help menu (on page 51) to display a dialog box that includes a variety of tips for using Oxygen XML Editor.
• You can select Show Dynamic Help view (on page 52) from the Help menu (on page 51) to dynamically opens a topic that is relevant to the focused editor, view, or dialog box.

Your First Document or Project

This section includes several topics that will help you get started with your first document or project.

Your First XML Document

To create your first XML document, select File > New or click the New button on the toolbar. The New document wizard (on page 368) is displayed:

![New Document Wizard](image)
You can either create a new XML document from scratch by choosing one of the available types in the wizard. You can also create one from a template by choosing a template from the Global templates or Framework templates folders. If you are looking for a common document type, such as DITA or DocBook, you can find templates for these document types in the Framework templates folder. If your company has created its own templates, you can also find them there. After you use this dialog box to create a few documents, those document types will appear in the Recently used folder, which allows you to easily create other new documents of those types.

For some document types, you may find a lot of different templates. For example, there are numerous templates for DocBook documents, and DITA topic types and maps. Choose the template that best meets your needs.

**Writing Your First Document**

Depending on the type of document you choose, the Oxygen XML Editor interface changes to support editing that document type. This may include new menus, toolbar buttons, and items in the contextual menus.

Also, depending on the type of document you choose, Oxygen XML Editor may open your document in Text (on page 353) or Author (on page 354) mode. Text mode shows the raw XML source file, while Author mode shows a graphical view of the document.

The availability of Author mode for your document type depends on the type you choose and if there is a CSS stylesheet available to create the Author mode. Oxygen XML Editor includes default Author mode views for most of the document types it supports. If your company has created its own document types, Author mode stylesheets may have also been created for that type. However, if you create a plain XML file, or one based on a schema that is not included in the Oxygen XML Editor built-in support, you need to edit it in Text mode or create your own Author mode CSS (on page 2307) for it.

You can switch back and forth between Author mode and Text mode at any time by clicking the buttons at the bottom left of the editor window. You do not lose any formatting when switching from Author to Text mode. Text and Author modes are just different views for the same XML document.

There is also a Grid mode (on page 354) available that displays all content in an XML document as a structured grid of nested tables. This is useful for certain kinds of documents, particularly those that are structured like databases. You can also use it when you want to display XML content in a table-like manner (for example, if you need to extract XML content to a spreadsheet (on page 587)).

If you use Author mode, you might find that it is similar to word processors that you are used to. Likewise, the Text mode is similar to many other typical text editors. If you are new to XML, the biggest difference is that XML documents have a particular structure that you have to follow. Oxygen XML Editor assists you with a continuous validation of the XML markup.

**Structuring Your First Document**

Each XML document type has a particular structure that you have to follow as you write and edit the document. Some document types give you a lot of choices, while others give you very few. In either case, you
need to make sure that your document follows the particular structure for the document type you are creating. This means:

- At any given location in the document, there are only certain XML elements allowed. Oxygen XML Editor helps you determine which elements are allowed. In Author mode, when you press Enter, Oxygen XML Editor assumes that you want to enter a new element and shows you a list of elements that can be created in this location. Keep typing until the element you want is highlighted and press Enter to insert the element. If you want to view the overall structure of a document and see what is allowed (and where), you can use the Model view (on page 545) (Window > Show View > Model).
- When you create certain elements, you may find that your text gets a jagged red underline and you get a warning that your content is invalid. This is usually because the element you have just created requires certain other elements inside of it. Your document will be invalid until you create those elements. Oxygen XML Editor helps you with this. If there is only one possible element that can go inside the element you just created, Oxygen XML Editor creates it for you. However, if there is more than one possibility, you have to create the appropriate elements yourself. In many cases, Oxygen XML Editor presents XML Quick Fixes (on page 813) that help you resolve errors by offering proposals to quickly fix problems such as missing required attributes or invalid elements.

**Editing Your First Document**

Once you have completed the first draft of your document, you may need to edit it. As with any editor, Oxygen XML Editor provides the normal cut, copy, and paste options as well as drag and drop editing. However, when you are editing an XML document, you have to make sure that your edits respect the structure of the XML document type. In fact, you are often editing the structure as well as the content of your document.

Oxygen XML Editor provides many tools to help you edit your structure and to keep your structure valid while editing text.

**The Document Breadcrumbs**

Across the top of the editor window, there is a set of breadcrumbs that shows you exactly where the insertion point is in the structure of the document. You can click any element in the breadcrumbs to select that entire element in the document.

```
book chapter sect1 sect2 sect3 para figure title
```

**Showing Tags**

To see exactly where you are in the structure of the document, you can show the tags graphically in the Author view. There are several levels of tag visibility that you can choose using the Tags Display Mode drop-down menu (on page 594) on the toolbar (the button may look a little different than this, as it changes to reflect the level of tags currently displayed).

**Outline View**
The **Outline view (on page 539)** shows you the structure of your document in outline format. You can use it to select elements, or to move elements around in the document.

![Outline View](image)

You can configure the **Outline** view to determine what is shown, such as element names, attributes, and comments. Certain choices may work better for particular document types. You can also filter the **Outline** view to show only elements with a certain name.

![Outline View Filtered to only Show Element Names](image)

**Cut and Paste, Drag and Drop**

You can cut and paste or drag and drop text, just as you would in any other editor. However, when you do this in **Author** view, it is important to remember that you are actually moving blocks of XML. When you cut and paste or drag and drop a block of XML, the result has to be valid both where the content is inserted, and where it is removed from.
A big part of doing this correctly is to make sure that you pick up the right block of text in the first place. Using the breadcrumbs or Outline view, or showing tags and using them to select content, can help ensure that you are selecting the right chunk of XML.

If you do try to paste or drop a chunk of XML somewhere that is not valid, Oxygen XML Editor warns you and tries to suggest actions that make it valid (such as by removing surrounding elements from the chunk you are moving, by creating a new element at the destination, or by inserting it in a nearby location).

If you are using Author mode, you can also switch to Text mode to see exactly which bits of XML you are selecting and moving.

Refactoring actions

You can perform many common structure edits, such as renaming an element or wrapping text in an element, using the actions in the Refactoring menu of the contextual menu (or the Document > Markup menu). More advanced refactoring operations are also available using the XML Refactoring tool (on page 840) that is available in the Tools menu.

Validating Your First Document

Validation is the process of making sure that an XML document abides by the rules of its schema. If Oxygen XML Editor knows how to find the schema, it validates the document for you as you type. Oxygen XML Editor finds the schema automatically for most of the document types created from templates. However, in some cases, you may have to tell it how to find the schema (on page 775).

When Oxygen XML Editor validates as you type, there is a small bar at the right edge of the editor that shows you if the document is invalid and where errors are found. If the indicator at the top of that bar is green, your document is valid. If the document is invalid, the indicator turns red and a red flag shows you where the errors are found. Click that flag to jump to the error. Remember that sometimes your document is invalid simply because the structure you are creating is not yet complete.

In addition to problems with the validity of the XML document itself, Oxygen XML Editor also reports warnings for a number of conditions, such as if your document contains a cross reference that cannot be resolved, or if Oxygen XML Editor cannot find the schema specified by the document. The location of these warnings is marked in yellow on the validation bar. If the document contains warnings, but no errors, the validity indicator turns yellow.

You can also validate your document at any time by selecting the Validate action from the Validation toolbar drop-down menu or the Document > Validate menu. When you validate in this manner, if errors are found, the validation result opens in a new pane at the bottom of the editor that shows each validation error on a separate line. Clicking the error takes you to the location in your document where the error was detected.
Proofing Your First Document

Oxygen XML Editor includes an automatic (as-you-type) spell checking feature (on page 456), as well as a manual spell checking action. To check the spelling of your document manually, use the Check Spelling action on the toolbar or from the Edit menu.

Transforming Your First Document

An XML document must be transformed to be published. Transformations are specific to the particular type of document you have created. For example, a DITA transformation cannot be used on a DocBook file, or vice versa. A single document type may have many multiple transformations that produce different kinds of outputs. For some document types, such as a DITA, many different content files may be combined together by a transformation. You need to locate and launch a transformation that is appropriate for your document type and the kind of output you want to generate.

Oxygen XML Editor uses transformation scenarios (on page 1432) to control the transformation process. Depending on the document type you have created, there may be several transformation scenarios already configured for your use. This may include the default transformation scenarios supplied by Oxygen XML Editor or ones created by your organization.

To see the list of transformations available for your document, select the Apply Transformation Scenario(s) action from the toolbar or the Document > Transformation menu. A list of available transformation scenarios is displayed. Choose one or more scenarios to apply, and click Apply associated. Exactly how your transformed content appears depends on how the transformation scenario is configured.

Getting Started with DITA

The information in this topic is meant to be a very basic starting point for those who are just getting started using DITA in Oxygen XML Editor. Oxygen XML Editor makes it easy to create, edit, manage, and publish DITA content, but it requires at least some basic DITA knowledge. To truly get the most out of Oxygen XML Editor and all of its DITA-related features, you should explore resources in the online DITA community to acquire knowledge of its concepts and uses.

Understanding DITA Topics

It is important to understand the role that a DITA topic plays in a DITA project. A DITA topic is not associated with a single published document. It is a separate entity that can potentially be included in many different books, help systems, or websites. Therefore, when you write a DITA topic you are not writing a book, a help system, or a website. You are writing an individual piece of content. This affects how you approach the writing task and how Oxygen XML Editor works to support you as you write.
Most of your topics are actually related to other topics, and those relationships can affect how you write and handle things such as links and content reuse. Oxygen XML Editor helps you manage those relationships. Depending on how your topics are related, you can use the tools provided in Oxygen XML Editor, along with the features of DITA, in a variety of ways.

**Creating a DITA Topic in Oxygen XML Editor**

To create a DITA topic *(on page 2961)*:

1. Select **File > New** or click the **New** button on the toolbar.

   **Step Result:** The **New Document Wizard** *(on page 368)* is displayed:

   *Figure 5. New DITA Document Wizard*

   ![](image)

   2. Go to **Framework templates > DITA > topic** and select the type of topic that you want to create.

   **Note:**

   If your organization has created DITA customizations, the appropriate template files may be in another location, and various types of topics may be provided for your use. Check with the
3. Select a file path where it will be saved. You can also optionally specify a title.
4. Click Create.

**Result:** Your document is opened in the editor. Eventually, you will need to add a reference to it in your DITA map (on page 42).

Your DITA topic is an XML document, thus all the editing features that Oxygen XML Editor provides for editing XML documents (on page 34) also apply to DITA topics. Oxygen XML Editor also provides additional specific DITA-related support for working with DITA topics (on page 2959), their associated DITA maps (on page 2895), and for creating DITA output (on page 3080).

**Role of Maps**

The basic method that DITA uses to express the relationship between topics is through a DITA map (on page 3218). Other relationships between topics, such as cross references, generally need to be made between topics in the same root map. DITA uses maps to determine which topics are part of any output that you create. While customized DITA solutions can use other mechanisms, generally DITA is not used as a way to publish individual topics. Output is created from a map and includes all the topics referenced by the map.

A publication is not always represented by a single map. For instance, if you are writing a book, you might use a submap to create each chapter and then organize the chapters in a main root map to create the book. This helps you to manage your content, offers the possibility of reusing submaps, and segregates content to support multiple people working on the same project.

**Creating a Map in Oxygen XML Editor**

To create a map (on page 2913):

1. Select File > New or click the New button on the toolbar.
2. Go to Framework templates > DITA Map > map and select the type of map you want to create.
3. Choose whether you want to open the map in the Editor or in the **DITA Maps Manager (on page 2897)**. Usually, opening it in the DITA Maps Manager is the best choice. The DITA Maps Manager presents a view of the DITA map that is similar to a table of contents.

![Figure 6. DITA Maps Manager View](image)

### Adding Existing Topics to a Map in Oxygen XML Editor

There are several ways to **add a topic reference to a map (on page 2917)**. Perhaps the easiest method is to add a reference to a topic that is already open in the editor:

1. Open the DITA topic in the main editing window.

2. Right-click the **DITA map** in the DITA Maps Manager view (on page 2897) and choose **Reference to the currently edited file** from the **Append Child, Insert Before, or Insert After** submenu.

   **Step Result:** This opens the Insert Reference dialog box (on page 2923) with all of the required fields already filled in for you.
3. You can fill in additional information in the various tabs in this dialog box or add it to the map later.
4. Select Insert and close to add a reference to your topic in the map.
5. Save the DITA map.

Adding New Topics to a Map in Oxygen XML Editor

As you add topics to your map, you may want to create a new topic as a child or sibling of another topic. This is usually done at the map level.

To add a new topic to a map (on page 2917), follow these steps:

1. In the DITA Maps Manager (on page 2897), right-click the node in the current map where you want to add the new topic.

2. Select one of the following actions:
   - Append Child > New - Select this action to insert the new topic as a child of the selected node. This action opens a New file dialog box (on page 2962) that allows you to select the type of document and assists you with naming it. After you have configured your new topic, click Create.
   - Insert Before > New - Select this action to insert the new topic as a sibling to the current node, before it. This action opens a New file dialog box (on page 2962) that allows you to select the type of document and assists you with naming it. After you have configured your new topic, click Create.
- **Insert After** > **New** - Select this action to insert the new topic as a sibling to the current node, after it. This action opens a New file dialog box *(on page 2962)* that allows you to select the type of document and assists you with naming it. After you have configured your new topic, click Create.

- **Duplicate** - Select this action to create a copy of the selected topic and insert it as a sibling. This action opens a dialog box that allows you to choose the file name and location for the newly created copy of the topic. After you have selected the name and path for your new topic, click OK.

> Note:
The value of the root ID is generated taking the Use the file name as the value of the root ID attribute option from the **DITA > Topics** preferences page *(on page 273)* into account. When the option is deselected, a unique ID is generated.

**Step Result:** The new topic is now referenced (as a `<topicref>`) in the **DITA map** at the location where you inserted it and the new topic is opened in the editor.

3. Save the **DITA map**.

You can also change the order and nesting of topics in the **DITA Maps Manager** view by doing either of the following:

- Select the topic to move while holding down the **Alt** key and use the arrow keys to move it around.
- Use the mouse to drag and drop the topic to the desired location.

The way your parent and child topics are organized in any particular output depends on both the configuration of those topics in the map and the rules of the output transformation that is applied to them. Do not assume that your topics must have the same organization for all output types. The map defines the organization of the topics, not the topics themselves. It is possible to create a variety of maps, each with different organization and configuration options to produce a variety of outputs.

### Adding Submaps in Oxygen XML Editor

If you have a large set of information, such as a long book or extensive help system, a single map can become long and difficult to manage. To make it easier to manage, you can break up the content into smaller submaps *(on page 2914)*. A submap might represent a chapter of a book, a section of a user manual, or a page on a website. To build a publication out of these smaller maps, you must add them to a map that represents the overall publication.

To add a child map to the current map *(on page 2914)*:
1. Right-click the parent DITA map in the DITA Maps Manager view (on page 2897) and choose Append child > Map reference.

   **Step Result:** This opens the Insert Reference dialog box (on page 2923) with all of the required fields already filled in for you.

2. You can fill in additional information in the various tabs in this dialog box or add it to the map later.
3. Select Insert and close to add a reference to your submap in the main map.
4. Save the main DITA map.

### Validating a Map in Oxygen XML Editor

Just as it is with your individual topics, it is important to validate your maps (on page 2941). Oxygen XML Editor provides a validation function for DITA maps that does more than simply validating that the XML is well-formed. It also does the following:

- Validates all of the relationships defined in the maps.
- Validates all of the files that are included in the map.
- Validates all of the links that are expressed in the files.

Validating the map that describes your entire publication validates all the files that make up the publication and all of the relationships between them.

To validate a map:

1. Click the Validate and Check for Completeness button in the DITA Maps Manager view (on page 2897).

   **Step Result:** This opens the DITA Map Completeness Check dialog box (on page 2942).

2. Select any of the various options you want to check.
3. Click Check to run the validation process.

### Publishing Your Topics in Oxygen XML Editor

As noted previously, in DITA standards you usually do not publish output from an individual topic. Instead, you create published output (on page 3080) by running a DITA transformation on a map. This collects all the topics that are referenced in the map, organizes them, and produces output in a particular format. By default, Oxygen XML Editor uses the transformations provided by the DITA Open Toolkit for publishing to various output formats (such as PDF, WebHelp or EPUB). Your organization may have created various custom transformations or modified the built-in DITA Open Toolkit transformations. In either case, Oxygen XML Editor manages them by using transformation scenarios.

To publish output for a map:
1. Click the Configure Transformation Scenario(s) button in the DITA Maps Manager view (on page 2897).

**Step Result:** This opens the Configure Transformation Scenario(s) dialog box (on page 1548).

![Configure Transformation Scenarios Dialog Box](image)

2. Select the appropriate transformation depending on the type of output you desire.
3. To change or view the configuration or storage options for a transformation scenario, select the transformation and click Edit.
4. Click Apply associated.

**Result:** Depending on the configuration of the transformation scenario, when the transformation is finished, your output may automatically be opened in the appropriate application.

**DITA Projects**

Once you have a basic understanding of DITA and how to work with DITA topics and maps, you probably want to create a DITA project to organize and manage your planned content/resources (on page 399). Oxygen XML Editor includes a Project view (on page 402) that helps you organize your projects and offers a variety of helpful project-related features and makes it easy to share your projects with other members of your team.
Tip:
There are several sample project templates available for DITA users that can be used as a starting point or for inspiration. These sample project templates are found in the Framework templates > DITA folder in the New Project wizard: (on page 399)

- **Sample DITA Project** - This is a basic DITA project meant to help new users see how a DITA project is structured.
- **Startup DITA Project** - This is a startup DITA project that imposes a custom set of options (e.g. spell check settings and custom dictionaries), a custom DITA framework extension (e.g. custom new file templates. custom actions, custom CSS used for visual editing) and a folder structure for a DITA project according to best practices. Once created, the project contains a Readme.html file that explains all customizations and their benefits. If you plan to start your own DITA project using a version control system (such as Git), you can use this startup DITA project template to customize various aspects of DITA editing and share them with your team.

Resources

For more information about getting started with DITA and how to work with DITA in Oxygen XML Editor, see our compiled collection of DITA-related webinars that are meant to help you with your journey into working with DITA: Webinars: Working with DITA in Oxygen.

Related information

- DITA Authoring (on page 2886)
- Editing XML Documents in Author Mode (on page 588)
- https://www.oxygenxml.com/dita/1.3/specs/
- Webinars: Working with DITA in Oxygen

Creating a New Project

Oxygen XML Editor allows you to organize your XML-related files into projects. This helps you manage and organize your files and also allows you to perform batch operations (such as validation and transformation) over multiple files. You can also share your project settings and transformation/validation scenarios (on page 415) with other users. Use the Project view (on page 402) to manage projects, and the files and folders contained within.

Creating a New Project

To create a new project, select 🗂️ New Project from the Project menu, the New menu in the contextual menu, or the drop-down menu at the top-left of the Project view.

This opens a new project wizard:
Tip:
There are several sample project templates available for DITA users that can be used as a starting point or for inspiration. These sample project templates are found in the **Framework templates > DITA** folder in the **New Project wizard: (on page 399)**

- **Sample DITA Project** - This is a basic DITA project meant to help new users see how a DITA project is structured.
- **Startup DITA Project** - This is a startup DITA project that imposes a custom set of options (e.g. spell check settings and custom dictionaries), a custom DITA framework extension (e.g. custom new file templates, custom actions, custom CSS used for visual editing) and a folder structure for a DITA project according to best practices. Once created, the project contains a **Readme.html** file that explains all customizations and their benefits. If you plan to start your own DITA project using a version control system (such as Git), you can use this startup DITA project template to customize various aspects of DITA editing and share them with your team.

With the exception of the **Default project** template, which is a pseudo-template and does not exist on the local disk (it is used only to create a new .xpr file), project templates are actually ZIP archives (with a .zxpr extension) and are stored within the file template directory structure (for example, frameworks\dita\templates\sample-project\Sample DITA Project.zxpr).

Tip:
Archives with a .zxpr extension can be edited in the **Archive Browser view (on page 2008).**
After selecting a project template, you can specify the following:

**Project file name**

Specifies the name of the new project file. Oxygen XML Editor provides a default proposal for the file name based on the following rules:

- If there is an `.xpr` file inside the archive, its name is used.
- Otherwise, the name of the template is used.

**Project directory**

Specifies the directory where the archive content will be extracted.

>Note:
The archive should not contain an extra single folder as the root. For the Project directory path to work properly, the archive must have the `.xpr` file on the first level, along with the other resources (files and folders).

Once you are done, click the Create button to begin the creation process. Oxygen XML Editor extracts the content from the archive inside the path specified in the Project directory field.

**Editor Variables in Project Templates**

By default, the editor variables inside project resources created from a project template are not resolved. To start having them resolved, the project template must be customized (on page 377) by using the expandEditorVariablesIncludeFilter property. This filter determines the resources where the editor variables will be resolved. If you need to exclude a subset of resources from the set specified by the expandEditorVariablesIncludeFilter property, the expandEditorVariablesExcludeFilter property can be used.

>Note:
Usually, project files (*.xpr), framework files (*.framework), and framework extension scripts (*.exf) should be excluded from the editor variable resolving process.

The values of the inclusion and exclusion filters can be file paths relative to the project directory that can use wildcards or simply wildcards. Each filter can have multiple values, separated by spaces.

Possible filter values:

- `./*` - Matches all resources from the first level in the project directory.
- `*` or `./**` - Matches all resources on all levels inside the project directory.
- `dir1/dir2/*.dita` - Matches all `.dita` files from `[PROJECT_DIR]/dir1/dir2`, but not from subdirectories of dir2.
- `dir1/dir2/**/*.dita` - Matches all `.dita` files from `[PROJECT_DIR]/dir1/dir2`, including those from subdirectories of dir2.
- `dir1/**/*` - Matches all resources on all levels inside `[PROJECT_DIR]/dir1.`
• `dir1/article1.xml, dir2/article2.xml` - Matches only the two `.xml` files.

• `./**/_suffix.md, ./**/prefix_*_.html` - Matches all `.md` files with names that end with `_suffix` and all `.html` files with names that start with `prefix_`.

Adding Items to the Project

To add items to the project, select any of the following actions that are available when invoking the contextual menu in the Project view:

- **New > File**
  
  Opens a New file dialog box that helps you create a new file and adds it to the project structure.

- **New > Folder**
  
  Opens a New Folder dialog box that allows you to specify a name for a new folder and adds it to the structure of the project.

The project itself is considered a logical folder. You can add a logical folder, or content to a logical folder, by using one of the following actions that are available in the contextual menu, when invoked from the project root:

- **New > Logical Folder**
  
  Creates a logical folder in the tree structure (the icon is a magenta folder on macOS - 📁).

- **New > Logical Folders from Web**
  
  Replicates the structure of a remote folder accessible over FTP/SFTP/WebDAV, as a structure of logical folders. The newly created logical folders contain the file structure of the folder it points to.

- **Add Folder**
  
  Adds a link to a physical folder, whose name and content mirror a real folder that exists in the local file system (the icon of this action is different on macOS - 📁).

- **Add Files**
  
  Adds links to files on the local file system.

- **Add Edited File**
  
  Adds a link to the currently edited file in the project.

Using Linked Folders (Shortcuts)

Another easy way to organize your XML working files is to place them in a directory and then to create a corresponding linked folder in your project. If you add new files to that folder, you can simply use the **Refresh (F5)** action from the project contextual menu and the Project view (on page 402) will display the existing files and subdirectories. If your files are scattered among several folders, but represent the same class of files, you might find it useful to combine them in a logical folder.
You can create linked folders (shortcuts) by dragging and dropping folders from the Windows Explorer (macOS Finder) to the project tree, or by selecting Add Folder in the contextual menu from the project root. Linked folders are displayed in the Project view (on page 402) with bold text. To create a file inside a linked folder, select the New > File action from the contextual menu. The linked files presented in the Project view (on page 402) are marked with a special icon.

Note:
Files may have multiple instances within the folder system, but cannot appear twice within the same folder.

For more information on managing projects and their content, see Project View (on page 402).

For more details about how you can share projects with other users, see Sharing a Project - Team Collaboration (on page 415).

Getting Help

If you run into specific problems while using Oxygen XML Editor you can take advantage of a variety of support related resources. Those resources include the following:

- The Oxygen XML Editor Support Section of the Website
- The Oxygen XML Editor Forum
- The Oxygen XML Editor Video Tutorials
- The Common Problems and Solutions Section of the User Manual (on page 2855)
- The Online Technical Support Form

The application also includes various specific help-related resources in the Help menu.

Help Menu

The Oxygen XML Editor Help menu provides various resources to assist you with your tasks.

This menu includes the following actions or options:

Welcome
This option opens the Welcome screen that includes some resources to assist you with using Oxygen XML Editor.

Help (F1)
Use this action (or the F1 key) to open a dialog box that presents a section in the User Manual that is appropriate for the context of the current cursor position. If the Use online help option is selected, this action will open the User Manual in an online mode.
Use online help

If this option is selected, the Help (F1) action will open the Oxygen XML Editor User Manual in an online mode.

Show Dynamic Help view

Use this action to open a view that loads the latest online WebHelp version of the Oxygen XML Editor User Manual, and dynamically opens a topic that is relevant to the focused editor, view, or dialog box. It requires Java 1.8 and an online connection. In Windows, if a Java 1.8 version is not detected, you will be advised to upgrade, while in Linux and macOS with Java 1.7 and lower, Oxygen XML Editor will attempt to load an offline version of the documentation. In all three operating systems, with Java 1.8, if an online connection is not detected, you will receive an error message advising you to check your proxy settings.

You can also open the Dynamic Help view by selecting it from the Window > Show View menu.

Install new add-ons

Opens a dialog box that allows you to install new add-ons (on page 3222) to extend the functionality of Oxygen XML Editor.

Check for add-ons updates

Opens a dialog box that allows you to check for updates on installed add-ons (on page 3222).

Manage add-ons

Opens a dialog box that allows you to manage installed add-ons (on page 3222).

Check for a New Version

Use this action to view information about the latest version of Oxygen XML Editor.

Browse Oxygen Website

Opens the Oxygen XML Editor website in your default internet browser.

Register

If you encounter problems with your Oxygen XML Editor license, you can use this option to open a dialog box that provides options for obtaining or using a license key.

Lock/Unlock floating license

If you are using a Floating License, you can lock it so that it does not get released to the pool (on page 105) unless you or the system administrator unlocks it.

Report problem

You can use this option to open a dialog box that allows you to write the description of a problem that was encountered while using the application. You can also select additional information to be sent to the technical support team in the five tabs:
- **General info** - You can edit your contact details in case you want to be contacted for further details or to be notified of a resolution.
- **Class Loader URLs** - You can choose whether or not to include the listed Class Loader URLs with your report.
- **System properties** - You can choose whether or not to include the listed system property details with your report.

**Tip:**
You are able to change the URL where the reported problem is sent by using the com.oxygenxml.report.problems.url System property. The report is sent in XML format through the report parameter of the POST HTTP method.

- **Plugins** - You can choose whether or not to include details about your installed plugins (on page 3222) with your report.
- **Frameworks** - You can choose whether or not to include details about your installed frameworks (on page 3219) with your report.

**Support Center**
Use this option to open the Oxygen XML Editor Support Section of the Website.

**Support Tools > Clipboard Inspector**
Opens a dialog box that displays extensive details of all the transferable objects from the clipboard. This is helpful if you experience problems while copying content from other applications and pasting it into Oxygen XML Editor. You can use the Copy button to copy all of this data and then paste it into an email to be sent to the Oxygen support team.

**Support Tools > Randomize XML text content**
Use this action when you need to send samples to the Oxygen support team and you want to keep the text content confidential. It opens a dialog box that allows you to select the resources that will have the text content randomized. You can then save the resources and send them to the Oxygen support team without fear of compromising sensitive or private data. For more information, see Randomize XML Text Content (on page 54).

**Warning:**
Before using this action, it is highly recommended that you copy the XML resources to be processed, save them in a separate folder, and then process this operation on the copies instead of the original files. Otherwise, you may lose your original content.

**Tip of the Day**
Opens a dialog box that offers tips for using Oxygen XML Editor.

**About**
Use this option to open a dialog box that contains information about Oxygen XML Editor and the installed version. This dialog box includes the following tabs:

- **Copyright** - This tab contains general information about the product and the version of the product you are using, along with contact details and the SGN number. Details regarding the memory usage are also presented at the bottom of the dialog box.

- **Libraries** - This tab presents the list of third-party libraries that Oxygen XML Editor uses. To view the End-User Licence Agreement of each library, double-click it.

- **Frameworks** - This tab contains a list with the frameworks (on page 3219) that are bundled with Oxygen XML Editor.

- **System Properties** - This tab contains a list with system properties and their values. The contextual menu allows you to select and copy the properties.

### Related information

Details to Submit in a Request for Technical Support Using the Online Form (on page 2864)

### Randomize XML Text Content

Oxygen XML Editor includes an action that randomizes the text content of an XML document. This action is available in the Help > Support Tools menu. It is helpful if you need to send XML samples to the Oxygen support team and you want to keep the text content confidential. It opens a dialog box that allows you to select the resources that will have the text content randomized. You can then save the resources and send them to the Oxygen support team without fear of compromising sensitive or private data.

⚠️ **Warning:**

Before using this action, it is highly recommended that you copy the XML resources to be processed, save them in a separate folder, and then perform this operation on the copies instead of the original files. Otherwise, you may lose your original content.
The Randomize XML Text Content dialog box includes the following options:

**Scope**

Allows you to select the set of files whose text content will be randomized by the operation. You can select from predefined resource sets (such as the current file, your whole project, the current DITA map (on page 3218) hierarchy for DITA projects, etc.) or you can define your own set of resources by creating a working set (on page 3224).

**Filters**

This section includes the following options:

- **Include files** - Allows you to filter the selected resources by using a file pattern. For example, to restrict the operation to only analyze build files you could use `build*.xml` for the file pattern.
- **Restrict only to known XML file types** - When selected, only resources with a known XML file type will be affected by the operation.
- **Look inside archives** - When selected, the resources inside archives will also be affected.

**Frequently Used Shortcut Keys**

Oxygen XML Editor includes numerous shortcut keys that are assigned to actions to help you edit content. All the shortcuts that are assigned to actions are displayed in the table in the Menu Shortcut Keys preference page (on page 294).
For information about how to assign or configure shortcut keys, see How to Assign a Shortcut Key or Edit an Existing Shortcut (on page 296).

<table>
<thead>
<tr>
<th><strong>Table 1. Frequently Used Shortcut Keys in Oxygen XML Editor</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Action</strong></td>
</tr>
<tr>
<td>Attribute Editor</td>
</tr>
<tr>
<td>Beginning</td>
</tr>
<tr>
<td>Check Spelling</td>
</tr>
<tr>
<td>Check Well-Formedness</td>
</tr>
<tr>
<td>Configure Transformation</td>
</tr>
<tr>
<td>Content Completion / New Line</td>
</tr>
<tr>
<td>Create Bookmark #</td>
</tr>
<tr>
<td>Content Completion (Text Mode)</td>
</tr>
<tr>
<td>Create Next Bookmark</td>
</tr>
<tr>
<td>Delete Next Word</td>
</tr>
<tr>
<td>Delete Previous Word</td>
</tr>
<tr>
<td>Action</td>
</tr>
<tr>
<td>-----------------------------</td>
</tr>
<tr>
<td>Delete Tags</td>
</tr>
<tr>
<td>Duplicate Lines Up (Text Mode)</td>
</tr>
<tr>
<td>Duplicate Lines Down (Text Mode)</td>
</tr>
<tr>
<td>End</td>
</tr>
<tr>
<td>Exit</td>
</tr>
<tr>
<td>Find</td>
</tr>
<tr>
<td>Find Next</td>
</tr>
<tr>
<td>Find Previous</td>
</tr>
<tr>
<td>Go To Bookmark</td>
</tr>
<tr>
<td>Go To Definition</td>
</tr>
<tr>
<td>Help</td>
</tr>
</tbody>
</table>
| Insert Para / Format Indent | `Ctrl + Shift + P`          | `Command + Shift + P` | - **Author** mode - Inserts a paragraph at the cursor position  
|                             |                             |                    | - **Text** mode - Formats and indents current document                               |
| Move Tab Left               | `Ctrl + Alt + Comma`        | `Ctrl + Alt + Comma` | Moves the current file tab one position to the left                                    |
| Move Tab Right              | `Ctrl + Alt + Period`       | `Ctrl + Alt + Period` | Moves the current file tab one position to the right                                   |
### Table 1. Frequently Used Shortcut Keys in Oxygen XML Editor (continued)

<table>
<thead>
<tr>
<th>Action</th>
<th>Windows/Linux Shortcut Keys</th>
<th>macOS Shortcut Keys</th>
<th>Description of Default Assigned Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Move Node Down (Author)</td>
<td>Alt + DownArrow</td>
<td>Alt + DownArrow</td>
<td>Moves the selected XML node down in <strong>Author</strong> mode</td>
</tr>
<tr>
<td>Move Node Down (Text)</td>
<td>Ctrl + Alt + DownArrow</td>
<td>Command + Alt + DownArrow</td>
<td>Moves the selected XML node down in <strong>Text</strong> mode</td>
</tr>
<tr>
<td>Move Node Up (Author)</td>
<td>Alt + UpArrow</td>
<td>Alt + UpArrow</td>
<td>Moves the selected XML node up in <strong>Author</strong> mode</td>
</tr>
<tr>
<td>Move Node Up (Text)</td>
<td>Ctrl + Alt + UpArrow</td>
<td>Command + Alt + UpArrow</td>
<td>Moves the selected XML node up in <strong>Text</strong> mode</td>
</tr>
<tr>
<td>New File</td>
<td>Ctrl + N</td>
<td>Command + N</td>
<td>Opens wizard for creating new documents</td>
</tr>
<tr>
<td>Next Word</td>
<td>Ctrl + RightArrow</td>
<td>Command + RightArrow</td>
<td>Navigates to next word</td>
</tr>
<tr>
<td>Open/Find Resource</td>
<td>Ctrl + Shift + R</td>
<td>Command + Shift + R</td>
<td>Opens the Open/Find Resource dialog box</td>
</tr>
<tr>
<td>Previous Word</td>
<td>Ctrl + LeftArrow</td>
<td>Command + LeftArrow</td>
<td>Navigates to previous word</td>
</tr>
<tr>
<td>Print Preview</td>
<td>Ctrl + P</td>
<td>Command + P</td>
<td>Opens the print preview (page setup) dialog box</td>
</tr>
<tr>
<td>Quick Assist</td>
<td>Alt + 1</td>
<td>Command + Alt + 1</td>
<td>Opens Quick Assist menu if actions are available in the current context (usually indicated with a bulb 🌟 icon in the left stripe)</td>
</tr>
<tr>
<td>Quick Find</td>
<td>Alt + Shift + F</td>
<td>Alt + Shift + F</td>
<td>Opens the Quick Find mechanism at the bottom of the editor</td>
</tr>
<tr>
<td>Redo</td>
<td>Ctrl + Y (Windows) - Ctrl + Shift + Z (Linux)</td>
<td>Command + Shift + Z</td>
<td>Redo last editing action</td>
</tr>
<tr>
<td>Refresh</td>
<td>F5</td>
<td>F5</td>
<td>Refresh</td>
</tr>
<tr>
<td>Remove Bookmarks</td>
<td>Ctrl + F7</td>
<td>Command + F7</td>
<td>Removes all bookmarks</td>
</tr>
<tr>
<td>Reopen Last</td>
<td>Ctrl + Alt + T</td>
<td>Command + Alt + T</td>
<td>Reopens the editor tab that was closed most recently</td>
</tr>
</tbody>
</table>
Table 1. Frequently Used Shortcut Keys in Oxygen XML Editor (continued)

<table>
<thead>
<tr>
<th>Action</th>
<th>Windows/Linux Shortcut Keys</th>
<th>macOS Shortcut Keys</th>
<th>Description of Default Assigned Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Closed Editor</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Reset Zoom</td>
<td>Ctrl + NumPad0</td>
<td>Command + NumPad0</td>
<td>Resets zoom (default font size)</td>
</tr>
<tr>
<td>Save</td>
<td>Ctrl + S</td>
<td>Command + S</td>
<td>Saves current document</td>
</tr>
<tr>
<td>Save All</td>
<td>Ctrl + Shift + S</td>
<td>Command + Shift + S</td>
<td>Saves all open files</td>
</tr>
<tr>
<td>Scroll Down</td>
<td>Ctrl + DownArrow</td>
<td>Command + DownArrow</td>
<td>Scrolls the editor down</td>
</tr>
<tr>
<td>Scroll Up</td>
<td>Ctrl + UpArrow</td>
<td>Command + Up Arrow</td>
<td>Scrolls the editor up</td>
</tr>
<tr>
<td>Select Content of Element</td>
<td>Alt + [Mouse Triple Click]</td>
<td>Alt + [Mouse Triple Click]</td>
<td>Selects the content of an element in <strong>Author</strong> mode.</td>
</tr>
<tr>
<td>Shift Left</td>
<td>Shift + Tab</td>
<td>Shift + Tab</td>
<td>• <strong>Author</strong> mode - Moves the cursor to the previous XML node</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• <strong>Text</strong> mode - Shifts content to the left</td>
</tr>
<tr>
<td>Shift Right</td>
<td>Tab</td>
<td>Tab</td>
<td>• <strong>Author</strong> mode - Moves cursor to the next XML node</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• <strong>Text</strong> mode - Shifts content to the right</td>
</tr>
<tr>
<td>Split Element</td>
<td>Alt + Shift + D</td>
<td>Ctrl + Alt + D</td>
<td>Splits the element the cursor position</td>
</tr>
<tr>
<td>Surround With</td>
<td>Ctrl + E</td>
<td>Command + E</td>
<td>Surrounds selected content with specified tag</td>
</tr>
<tr>
<td>Switch Tabs</td>
<td>Ctrl + Tab / Ctrl + Shift + Tab</td>
<td>Command + Tab / Command + Shift + Tab</td>
<td>Switches between open tabs</td>
</tr>
<tr>
<td>Transform</td>
<td>Ctrl + Shift + T</td>
<td>Command + Shift + T</td>
<td>Opens a dialog box for selecting a transformation scenario</td>
</tr>
<tr>
<td>Underline / Open URL</td>
<td>Ctrl + U</td>
<td>Command + U</td>
<td>• Underlines selected content (in the main editor)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Opens the URL (when focus is outside the main editor)</td>
</tr>
</tbody>
</table>
Table 1. Frequently Used Shortcut Keys in Oxygen XML Editor (continued)

<table>
<thead>
<tr>
<th>Action</th>
<th>Windows/Linux Shortcut Keys</th>
<th>macOS Shortcut Keys</th>
<th>Description of Default Assigned Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Undo</td>
<td>Ctrl + Z</td>
<td>Command + Z</td>
<td>Undo last editing action</td>
</tr>
<tr>
<td>Validate</td>
<td>Ctrl + Shift + V</td>
<td>Command + Shift + V</td>
<td>Validates current document</td>
</tr>
<tr>
<td>Zoom In</td>
<td>Ctrl + NumPad+</td>
<td>Command + NumPad+</td>
<td>Zooms in (increase font size)</td>
</tr>
<tr>
<td>Zoom Out</td>
<td>Ctrl + NumPad-</td>
<td>Command + NumPad-</td>
<td>Zooms out (decrease font size)</td>
</tr>
</tbody>
</table>

Troubleshooting:
If you encounter problems with keyboard shortcuts not working as expected, see Keyboard Shortcuts Result in Unexpected Behavior (on page 2876) or Keyboard Shortcuts Do Not Work At All (on page 2876).

Accessibility Support in Oxygen

The Oxygen team is dedicated to developing software products that are usable for everyone, including those with physical challenges and disabilities. Oxygen XML Editor is designed to adhere to the U.S. Government Section 508 accessibility standards: https://www.oxygenxml.com/xml_editor/section508.html.

Adjusting Fonts and Colors

If you have low vision, go to Options > Preferences > Appearance > Fonts where you can adjust the font styles and sizes used in the entire application, both for the editing areas and UI labels. If you have color blindness, you can also adjust most of the colors used in Oxygen XML Editor by going to Options > Preferences > Appearance and changing the current color theme. You can also search for other color-related settings in the Preferences dialog box.

Installing Oxygen XML Editor

Installation kits for Windows and Linux are made using the Install4j product. If you have problems navigating the Install4j installation wizard, you can run the installation from a command-prompt application using the -c flag (on page 88) like this:

C:\Users\your_user_name\Downloads\oxygenAuthor-64bit.exe -c

Screen Reader Software

If you are using a text to speech narrator, Oxygen XML Editor supports this since it is a Java application and it is periodically tested on Windows using both the NVDA and JAWS screen readers.
Using the JAWS (Job Access With Speech) Screen Reader with Oxygen XML Editor

The JAWS screen reader can be downloaded from: http://www.freedomscientific.com/Products/Blindness/JAWS.

For JAWS to work, you need to enable the Java access bridge in Oxygen XML Editor: http://docs.oracle.com/javase/7/docs/technotes/guides/access/enable_and_test.html.

To enable the Java access bridge:

1. Since Oxygen XML Editor comes bundled with its own Java VM, you need to open a command-prompt application and use the cd command to go to the Oxygen XML Editor installation directory (for example, in Windows, it would be something like this:  

   ```bash
   cd C:\Program Files\Oxygen XML Editor 21.1
   ```

2. Then run the following command:

   ```bash
   jre\bin\jabswitch -enable
   ```

3. Press Enter and you should receive a notification that the access bridge has been enabled.

Once the Java access bridge is enabled and as long as the JAWS narrator is active, when Oxygen XML Editor starts, the narrator will start reading content from Oxygen XML Editor and you can interact with the application and read menus, content from open XML documents, and UI components from dialog boxes and side views.

Using the NVDA Screen Reader with Oxygen XML Editor

The NVDA screen reader can be downloaded for free from: https://www.nvaccess.org/.

For NVDA to work, you need to enable the Java access bridge in Oxygen XML Editor: http://docs.oracle.com/javase/7/docs/technotes/guides/access/enable_and_test.html.

To enable the Java access bridge:

1. Since Oxygen XML Editor comes bundled with its own Java VM, you need to open a command-prompt application and use the cd command to go to the Oxygen XML Editor installation directory (for example, in Windows, it would be something like this:

   ```bash
   cd C:\Program Files\Oxygen XML Editor 21.1
   ```

2. Then run the following command:

   ```bash
   jre\bin\jabswitch -enable
   ```

3. Press Enter and you should receive a notification that the access bridge has been enabled.

Once the Java access bridge is enabled and as long as the NVDA narrator is started, when Oxygen XML Editor starts, the narrator will start reading content from Oxygen XML Editor and you can interact with the application and read menus, content from open XML documents, and UI components from dialog boxes and side views.
Important:

If after these steps the narrator still does not read anything from a started Oxygen XML Editor application, please go to the folder `C:\Windows\SysWOW64\` and make sure the library `WindowsAccessBridge-32.dll` is present there. If it is not present, try to search online, download the library file and copy it to the folder. Then restart Oxygen XML Editor.

Besides the main editing area, Oxygen XML Editor also has side views (for example, the Attributes, Outline, Elements views) that help with editing the XML content. NVDA versions 2020.1 and older have a registered bug that makes the narrator read content from the side views when editing in the main editing area. Because of this problem, when using NVDA versions 2020.1 or older, the following workflow is suggested:

1. Start Oxygen XML Editor.
2. Go to the Window menu and select Maximize Editing Area (or hold Alt, then W, then M). This action will hide all side views and allow you to properly edit in the main editing area.
3. Whenever you want to open a side view, go to Window > Show View (or hold Alt, then W, then S) and choose the view you want to open. For example, to show the Elements view, you can hold Alt, then W, then S, then E.
4. When you are done using the side view, go to the Window menu and select Hide current view (or hold Alt, then W, then H) to hide the side view and return the focus to the main editing area.

Hints for the Visually Impaired

Here are a few hints for using Oxygen XML Editor if you are visually impaired:

- The top main menu contains actions to open, save, and close documents, switch between open documents, or switch between the various editing modes for XML documents that are already open. All actions in the main menu bar should have mnemonics making it possible to memorize various shortcuts. For example, using the `alt-w-s-e` shortcut should open the Window menu, open the Show view submenu from it and show the Elements view,
  - The File menu contains actions to open, save, or close the currently edited XML document.
  - The Edit menu contains actions to undo/redo or cut/copy/paste content. They also have the usual shortcuts that can be used instead of directly invoking the actions from the menu.
  - The Find menu contains an action to show the Find/Replace dialog box. Sometimes the JAWS narrator overloads the `CTRL+F` shortcut and presents its own find/replace window but the Oxygen XML Editor Find/Replace dialog box provides the ability to perform complex find/replace operations in the open file.
  - In the Options menu, you have access to the Preferences dialog box that contains global application settings and access to the Menu Shortcut Keys table where you can configure shortcuts for the most commonly used actions.
  - The Window menu includes actions to switch between open XML documents. Also, you can use the Show view submenu to open a particular side view and move the focus to that view.
- An open XML document can be edited with accessibility support either in the Text editing mode (where the XML tags are accessible in the edited content) or in the visual Author editing mode (where the XML
tags are hidden and only the text content is shown). You can switch between these editing modes by using the Document > Edit Mode menu.

- **Text** mode provides access to the entire source document with all of its XML content, just like you have in any text editing application.

Pressing the < key will present a list of available XML elements (on page 3217). If you do not want to choose from the list whenever you want to insert an XML element, you have two choices:
  - After the list of available XML elements is shown, you can press the ESC key to close it and continue to manually insert the XML tag.
  - You can disable the content completion list from the Options > Preferences > Editor / Content Completion page by deselecting Enable Content Completion. After the content completion is disabled, you can force it to be displayed by using the Ctrl+Space keyboard shortcut.

In addition, using the Window > Show view submenu, you can change focus to the Attributes, Elements, or Outline view. The Attributes view presents the existing and possible attributes that can be inserted in an XML tag. The Elements view shows you the list of XML elements that can be inserted at the cursor position (also, pressing F2 on a selected element presents its annotation). The Outline view shows the current path in the XML structure.

- **Author** mode is useful for reviewing written XML content because it has support for change tracking and for adding comments. Editing in the Author visual editing mode, you have access to only the text content in the XML document.

Pressing Shift+F2 will read the current element context where the cursor is located. Pressing Ctrl+Shift+F3 will read the current element context and the entire path in the XML structure where the cursor is located. You can also use the Outline view to better understand the XML structure.

In the Author editing mode, you can also use the Attributes and Elements views similar to using them in the Text editing mode. Pressing Enter in the Author visual editing mode can also be used to present a list of allowed elements at the current position.

Oxygen XML Editor VPAT Accessibility Conformance Report

**International Edition**

VPAT® Version 2.3 – April 2019

Name of Product/Version

Oxygen XML Editor 24.1

Product Description

Oxygen XML Editor is a cross-platform application designed to accommodate all of your XML editing, authoring, developing, and publishing needs.
Notes

Oxygen XML Editor has been designed and enhanced to adhere to the U.S. Government Section 508 accessibility standards and the Web Content Accessibility Guidelines (WCAG). For details, see Oxygen XML Editor Accessibility (on page 60).

Evaluation Methods Used:

The following applications were used for testing Oxygen XML Editor:

- NVDA assistive technology
- JAWS assistive technology

Applicable Standards/Guidelines

This report covers the degree of conformance for the following accessibility standards/guidelines:

<table>
<thead>
<tr>
<th>Standard/Guideline</th>
<th>Included In Report</th>
</tr>
</thead>
<tbody>
<tr>
<td>Web Content Accessibility Guidelines 2.0</td>
<td>Level A - Yes</td>
</tr>
<tr>
<td></td>
<td>Level AA - Yes</td>
</tr>
<tr>
<td></td>
<td>Level AAA - No</td>
</tr>
<tr>
<td>Web Content Accessibility Guidelines 2.1</td>
<td>Level A - Yes</td>
</tr>
<tr>
<td></td>
<td>Level AA - Yes</td>
</tr>
<tr>
<td></td>
<td>Level AAA - No</td>
</tr>
<tr>
<td>Revised Section 508 standards published January 18, 2017 and corrected January 22, 2018</td>
<td>Yes</td>
</tr>
<tr>
<td>EN 301 549 Accessibility requirements suitable for public procurement of ICT products and services in Europe - V2.1.2 (2018-08)</td>
<td>No</td>
</tr>
</tbody>
</table>

Terms

The terms used in the Conformance Level information are defined as follows:

- **Supports**: The functionality of the product has at least one method that meets the criterion without known defects or meets with equivalent facilitation.
- **Partially Supports**: Some functionality of the product does not meet the criterion.
- **Does Not Support**: The majority of product functionality does not meet the criterion.
• **Not Applicable**: The criterion is not relevant to the product.
• **Not Evaluated**: The product has not been evaluated against the criterion. This can be used only in WCAG 2.0 Level AAA.

**WCAG 2.x Report**

Tables 1 and 2 also document conformance with:

Revised Section 508: Chapter 5 – 501.1 Scope, 504.2 Content Creation or Editing, and Chapter 6 – 602.3 Electronic Support Documentation.

**Note:**
When reporting on conformance with the WCAG 2.x Success Criteria, they are scoped for full pages, complete processes, and accessibility-supported ways of using technology as documented in the [WCAG 2.0 Conformance Requirements](#).

### Table 1: Success Criteria, Level A

<table>
<thead>
<tr>
<th>Criteria</th>
<th>Conformance Level</th>
<th>Remarks and Explanations</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>1.1.1 Non-text Content</strong> (Level A)</td>
<td>Supports</td>
<td>Text alternatives are provided for non-text content.</td>
</tr>
<tr>
<td>Also applies to: Revised Section 508</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>501 (Web)(Software)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>504.2 (Authoring Tool)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>602.3 (Support Docs)</td>
<td></td>
</tr>
<tr>
<td><strong>1.2.1 Audio-only and Video-only (Prerecorded)</strong> (Level A)</td>
<td>Not Applicable</td>
<td>The product does not play audio and video content to end users.</td>
</tr>
<tr>
<td>Also applies to: Revised Section 508</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>501 (Web)(Software)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>504.2 (Authoring Tool)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>602.3 (Support Docs)</td>
<td></td>
</tr>
<tr>
<td><strong>1.2.2 Captions (Prerecorded)</strong> (Level A)</td>
<td>Not Applicable</td>
<td>The product does not provide prerecorded media that requires captions.</td>
</tr>
<tr>
<td>Also applies to: Revised Section 508</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Criteria</td>
<td>Conformance Level</td>
<td>Remarks and Explanations</td>
</tr>
<tr>
<td>-------------------------------------------------------------------------</td>
<td>-------------------</td>
<td>----------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>1.2.3 Audio Description or Media Alternative (Prerecorded) (Level A)</td>
<td>Not Applicable</td>
<td>The product does not provide pre-recorded media that requires alternate descriptions.</td>
</tr>
<tr>
<td>Also applies to:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Revised Section 508</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.3.1 Info and Relationships (Level A)</td>
<td>Partially Supports</td>
<td>Information, structure, and relationships conveyed through presentation can be programmatically determined or are available in text, with exceptions that include:</td>
</tr>
<tr>
<td>Also applies to:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Revised Section 508</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.3.2 Meaningful Sequence (Level A)</td>
<td>Supports</td>
<td>The product presents content in a meaningful sequence.</td>
</tr>
<tr>
<td>Also applies to:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Revised Section 508</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.3.3 Sensory Characteristics (Level A)</td>
<td>Supports</td>
<td>The product provides textual identification for understanding and operating content.</td>
</tr>
<tr>
<td>Also applies to:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Revised Section 508</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.4.1 Use of Color (Level A)</td>
<td>Supports</td>
<td>Color is not used as the only visual means of conveying information, in-</td>
</tr>
<tr>
<td>Also applies to:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Criteria</td>
<td>Conformance Level</td>
<td>Remarks and Explanations</td>
</tr>
<tr>
<td>----------------------------------------------</td>
<td>-------------------</td>
<td>--------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Revised Section 508</td>
<td></td>
<td>dicating an action, prompting a response, or distinguishing a visual element.</td>
</tr>
<tr>
<td>• 501 (Web)(Software)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• 504.2 (Authoring Tool)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• 602.3 (Support Docs)</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>1.4.2 Audio Control</strong> (Level A)</td>
<td>Not Applicable</td>
<td>There is no sound that plays automatically by default.</td>
</tr>
<tr>
<td>Also applies to:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Revised Section 508</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• 501 (Web)(Software)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• 504.2 (Authoring Tool)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• 602.3 (Support Docs)</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>2.1.1 Keyboard</strong> (Level A)</td>
<td>Partially Supports</td>
<td>Most of the content is operable through a keyboard interface, with exceptions that include:</td>
</tr>
<tr>
<td>Also applies to:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Revised Section 508</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• 501 (Web)(Software)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• 504.2 (Authoring Tool)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• 602.3 (Support Docs)</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>2.1.2 No Keyboard Trap</strong> (Level A)</td>
<td>Partially Supports</td>
<td>The product does not usually have user interface elements that trap the keyboard focus. Exceptions include some trees located in side views and dialog boxes.</td>
</tr>
<tr>
<td>Also applies to:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Revised Section 508</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• 501 (Web)(Software)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• 504.2 (Authoring Tool)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• 602.3 (Support Docs)</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>2.1.4 Character Key Shortcuts</strong> (Level A</td>
<td>Not Applicable</td>
<td>The product does not include character key shortcuts.</td>
</tr>
<tr>
<td>2.1 only)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Also applies to:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Revised Section 508 – Does not apply</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>2.2.1 Timing Adjustable</strong> (Level A)</td>
<td>Not Applicable</td>
<td>The product does not include time limits.</td>
</tr>
<tr>
<td>Also applies to:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Criteria</td>
<td>Conformance Level</td>
<td>Remarks and Explanations</td>
</tr>
<tr>
<td>----------------------------------------------</td>
<td>-------------------</td>
<td>--------------------------</td>
</tr>
<tr>
<td>Revised Section 508</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• 501 (Web)(Software)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• 504.2 (Authoring Tool)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• 602.3 (Support Docs)</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>2.2.2 Pause, Stop, Hide</strong> (Level A)</td>
<td>Supports</td>
<td>Side views update their content automatically as a result of the user interaction with the open documents. They can be hidden or this functionality can be inhibited. The product does not include other elements that automatically move, blink, or scroll.</td>
</tr>
<tr>
<td>Also applies to:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Revised Section 508</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• 501 (Web)(Software)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• 504.2 (Authoring Tool)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• 602.3 (Support Docs)</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>2.3.1 Three Flashes or Below Threshold</strong>  (Level A)</td>
<td>Supports</td>
<td>The product does not have content that flashes more than three times in any one second.</td>
</tr>
<tr>
<td>Also applies to:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Revised Section 508</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• 501 (Web)(Software)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• 504.2 (Authoring Tool)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• 602.3 (Support Docs)</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>2.4.1 Bypass Blocks</strong> (Level A)</td>
<td>Not Applicable</td>
<td>The application does not contain blocks of repeated content.</td>
</tr>
<tr>
<td>Also applies to:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Revised Section 508</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• 501 (Web)(Software) – Does not apply to non-web software</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• 504.2 (Authoring Tool)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• 602.3 (Support Docs) – Does not apply to non-web docs</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>2.4.2 Page Titled</strong> (Level A)</td>
<td>Supports</td>
<td>Each side view and dialog box in the application has a title.</td>
</tr>
<tr>
<td>Also applies to:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Revised Section 508</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Criteria</td>
<td>Conformance Level</td>
<td>Remarks and Explanations</td>
</tr>
<tr>
<td>----------------------------------------------</td>
<td>-------------------</td>
<td>--------------------------------------------------</td>
</tr>
<tr>
<td>• 501 (Web)(Software)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• 504.2 (Authoring Tool)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• 602.3 (Support Docs)</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>2.4.3 Focus Order</strong> (Level A)</td>
<td>Supports</td>
<td>Focusable components receive focus in an order that preserves meaning and operability.</td>
</tr>
<tr>
<td>Also applies to:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Revised Section 508</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• 501 (Web)(Software)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• 504.2 (Authoring Tool)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• 602.3 (Support Docs)</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>2.4.4 Link Purpose (In Context)</strong> (Level A)</td>
<td>Supports</td>
<td>The purpose of each link can be determined from the link text alone or from the link text together with its programmatically-determined link context.</td>
</tr>
<tr>
<td>Also applies to:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Revised Section 508</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• 501 (Web)(Software)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• 504.2 (Authoring Tool)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• 602.3 (Support Docs)</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>2.5.1 Pointer Gestures</strong> (Level A 2.1 only)</td>
<td>Not Applicable</td>
<td>The product does not have functionality that requires multi-point or path-based gestures.</td>
</tr>
<tr>
<td>Also applies to:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Revised Section 508 – Does not apply</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>2.5.2 Pointer Cancellation</strong> (Level A 2.1 only)</td>
<td>Partially Supports</td>
<td>Almost all pointer operations in the product are activated on Up events. Exceptions may include selection changes in the dialog box for new files and in the side views.</td>
</tr>
<tr>
<td>Also applies to:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Revised Section 508 – Does not apply</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>2.5.3 Label in Name</strong> (Level A 2.1 only)</td>
<td>Supports</td>
<td>The names of the user interface components contain the text that is presented visually.</td>
</tr>
<tr>
<td>Also applies to:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Revised Section 508 – Does not apply</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>2.5.4 Motion Actuation</strong> (Level A 2.1 only)</td>
<td>Not Applicable</td>
<td>The product does not contain functionality that can be operated by device or user motion.</td>
</tr>
<tr>
<td>Also applies to:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Revised Section 508 – Does not apply</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Criteria</td>
<td>Conformance Level</td>
<td>Remarks and Explanations</td>
</tr>
<tr>
<td>----------</td>
<td>-------------------</td>
<td>--------------------------</td>
</tr>
<tr>
<td><strong>3.1.1 Language of Page</strong> (Level A)</td>
<td>Does Not Support</td>
<td>The product does not report the default language for each open document.</td>
</tr>
<tr>
<td>Also applies to: Revised Section 508</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• 501 (Web)(Software)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• 504.2 (Authoring Tool)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• 602.3 (Support Docs)</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>3.2.1 On Focus</strong> (Level A)</td>
<td>Supports</td>
<td>No changes of context occur when any component receives focus.</td>
</tr>
<tr>
<td>Also applies to: Revised Section 508</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• 501 (Web)(Software)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• 504.2 (Authoring Tool)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• 602.3 (Support Docs)</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>3.2.2 On Input</strong> (Level A)</td>
<td>Supports</td>
<td>Changing the setting of any user interface component does not automatically cause a change of context.</td>
</tr>
<tr>
<td>Also applies to: Revised Section 508</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• 501 (Web)(Software)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• 504.2 (Authoring Tool)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• 602.3 (Support Docs)</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>3.3.1 Error Identification</strong> (Level A)</td>
<td>Supports</td>
<td>Product shows error messages when user input is invalid.</td>
</tr>
<tr>
<td>Also applies to: Revised Section 508</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• 501 (Web)(Software)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• 504.2 (Authoring Tool)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• 602.3 (Support Docs)</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>3.3.2 Labels or Instructions</strong> (Level A)</td>
<td>Partially Supports</td>
<td>Most input areas in the product provide labels and instructions. Exceptions include the content completion windows in the Text, Grid, Author, and schema Design modes.</td>
</tr>
<tr>
<td>Also applies to: Revised Section 508</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### 4.1.1 Parsing (Level A)

**Also applies to:**
- Revised Section 508

<table>
<thead>
<tr>
<th>Criteria</th>
<th>Conformance Level</th>
<th>Remarks and Explanations</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.1.1 Parsing (Level A)</td>
<td>Supports</td>
<td>All labels presenting HTML content in the product have valid HTML content.</td>
</tr>
</tbody>
</table>

### 4.1.2 Name, Role, Value (Level A)

**Also applies to:**
- Revised Section 508

<table>
<thead>
<tr>
<th>Criteria</th>
<th>Conformance Level</th>
<th>Remarks and Explanations</th>
</tr>
</thead>
</table>
| 4.1.2 Name, Role, Value (Level A) | Partially Supports | Almost all visual components in the product have identifiable names and roles with exceptions that include:
  - Grid editing mode.
  - Schema **Design** editing mode. |

### Table 2: Success Criteria, Level AA

<table>
<thead>
<tr>
<th>Criteria</th>
<th>Conformance Level</th>
<th>Remarks and Explanations</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>1.2.4 Captions (Live)</strong> (Level AA)</td>
<td>Not Applicable</td>
<td>The product does not contain live audio content.</td>
</tr>
<tr>
<td><strong>1.2.5 Audio Description (Prerecorded)</strong> (Level AA)</td>
<td>Not Applicable</td>
<td>The product does not provide prerecorded video content that requires audio description.</td>
</tr>
<tr>
<td>Criteria</td>
<td>Conformance Level</td>
<td>Remarks and Explanations</td>
</tr>
<tr>
<td>----------------------------------------------------</td>
<td>-------------------</td>
<td>----------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>1.3.4 Orientation</strong> (Level AA 2.1 only)</td>
<td>Supports</td>
<td>Content does not restrict its view and operation to a single display orientation.</td>
</tr>
<tr>
<td>Also applies to:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Revised Section 508 – Does not apply</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>1.3.5 Identify Input Purpose</strong> (Level AA 2.1 only)</td>
<td>Not Applicable</td>
<td>The content does not contain input fields that collect information about the user.</td>
</tr>
<tr>
<td>Also applies to:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Revised Section 508 – Does not apply</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>1.4.3 Contrast (Minimum)</strong> (Level AA)</td>
<td>Partially Supports</td>
<td>The product has sufficient contrast between the foreground and background colors, with few exceptions, including:</td>
</tr>
<tr>
<td>Also applies to:</td>
<td></td>
<td>• Change tracking content in the document for some of the author colors.</td>
</tr>
<tr>
<td>Revised Section 508</td>
<td></td>
<td>• Change tracking content in some review panel items, when the item is selected.</td>
</tr>
<tr>
<td>• 501 (Web)(Software)</td>
<td></td>
<td>• Placeholders shown in empty elements.</td>
</tr>
<tr>
<td>• 504.2 (Authoring Tool)</td>
<td></td>
<td>• Comments marked as done.</td>
</tr>
<tr>
<td>• 602.3 (Support Docs)</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>1.4.4 Resize text</strong> (Level AA)</td>
<td>Partially Supports</td>
<td>In most situations, text in the main editing area, side views, and dialog boxes can be resized to reasonable dimensions by increasing the font without loss of content or functionality and without using assistive technology.</td>
</tr>
<tr>
<td>Also applies to:</td>
<td></td>
<td>Sizes of certain text components cannot be increased.</td>
</tr>
<tr>
<td>Revised Section 508</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• 501 (Web)(Software)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• 504.2 (Authoring Tool)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• 602.3 (Support Docs)</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>1.4.5 Images of Text</strong> (Level AA)</td>
<td>Supports</td>
<td>The few buttons with icons containing text characters also provide text alternatives.</td>
</tr>
<tr>
<td>Also applies to:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Revised Section 508</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Criteria</td>
<td>Conformance Level</td>
<td>Remarks and Explanations</td>
</tr>
<tr>
<td>-------------------------------------------------------------------------</td>
<td>-------------------</td>
<td>--------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>• 501 (Web)(Software)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• 504.2 (Authoring Tool)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• 602.3 (Support Docs)</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>1.4.10 Reflow</strong> (Level AA 2.1 only)</td>
<td>Partially Supports</td>
<td>The majority of the user interface controls can be presented without loss of information or functionality, and without requiring scrolling in two dimensions. In the editor area, the text will re-flow depending on the editor mode (Text/Grid/Author/Design) and both horizontal and vertical scrolls may be needed.</td>
</tr>
<tr>
<td>Also applies to:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Revised Section 508 – Does not apply</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>1.4.11 Non-text Contrast</strong> (Level AA 2.1 only)</td>
<td>Supports</td>
<td>User interface components and states have sufficient contrast against adjacent colors.</td>
</tr>
<tr>
<td>Also applies to:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Revised Section 508 – Does not apply</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>1.4.12 Text Spacing</strong> (Level AA 2.1 only)</td>
<td>Supports</td>
<td>There is no loss of content or functionality occurs by setting line height (line spacing), spacing following paragraphs, letter spacing, and word spacing.</td>
</tr>
<tr>
<td>Also applies to:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Revised Section 508 – Does not apply</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>1.4.13 Content on Hover or Focus</strong> (Level AA 2.1 only)</td>
<td>Partially Supports</td>
<td>The tooltips for most user interface simple components (buttons) are not hoverable and persistent.</td>
</tr>
<tr>
<td>Also applies to:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Revised Section 508 – Does not apply</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>2.4.5 Multiple Ways</strong> (Level AA)</td>
<td>Supports</td>
<td>There are multiple ways to navigate between the open documents.</td>
</tr>
<tr>
<td>Also applies to:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Revised Section 508</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• 501 (Web)(Software) – Does not apply to non-web software</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• 504.2 (Authoring Tool)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• 602.3 (Support Docs) – Does not apply to non-web docs</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Criteria</td>
<td>Conformance Level</td>
<td>Remarks and Explanations</td>
</tr>
<tr>
<td>----------------------------------</td>
<td>-------------------</td>
<td>-----------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>2.4.6 Headings and Labels</strong> (Level AA)</td>
<td>Supports</td>
<td>Headings and labels describe the topic or purpose.</td>
</tr>
<tr>
<td>Also applies to:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Revised Section 508</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• 501 (Web)(Software)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• 504.2 (Authoring Tool)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• 602.3 (Support Docs)</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>2.4.7 Focus Visible</strong> (Level AA)</td>
<td>Partially Supports</td>
<td>The product has a visible indication of focus for almost all user interface controls (buttons, text fields, combo boxes, etc.) Exceptions include some combo boxes located in the DITA Maps Manager and SharePoint Browser view, and some filtering buttons located in the Templates tab from the New/Edit scenario and the DITA Reusable Components view.</td>
</tr>
<tr>
<td>Also applies to:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Revised Section 508</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• 501 (Web)(Software)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• 504.2 (Authoring Tool)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• 602.3 (Support Docs)</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>3.1.2 Language of Parts</strong> (Level AA)</td>
<td>Does Not Support</td>
<td>The language of parts is not specified.</td>
</tr>
<tr>
<td>Also applies to:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Revised Section 508</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• 501 (Web)(Software)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• 504.2 (Authoring Tool)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• 602.3 (Support Docs)</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>3.2.3 Consistent Navigation</strong> (Level AA)</td>
<td>Supports</td>
<td>The product has a consistent navigation mechanism.</td>
</tr>
<tr>
<td>Also applies to:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Revised Section 508</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• 501 (Web)(Software)</td>
<td>501 (Web)(Software) – Does not apply to non-web software</td>
<td></td>
</tr>
<tr>
<td>• 504.2 (Authoring Tool)</td>
<td>504.2 (Authoring Tool)</td>
<td></td>
</tr>
<tr>
<td>• 602.3 (Support Docs)</td>
<td>602.3 (Support Docs) – Does not apply to non-web docs</td>
<td></td>
</tr>
<tr>
<td><strong>3.2.4 Consistent Identification</strong> (Level AA)</td>
<td>Supports</td>
<td>The product components are identified consistently.</td>
</tr>
<tr>
<td>Also applies to:</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Criteria

<table>
<thead>
<tr>
<th>Criteria</th>
<th>Conformance Level</th>
<th>Remarks and Explanations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Revised Section 508</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• 501 (Web)(Software) – Does not apply to non-web software</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• 504.2 (Authoring Tool)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• 602.3 (Support Docs) – Does not apply to non-web docs</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>3.3.3 Error Suggestion</strong> (Level AA)</td>
<td>Supports</td>
<td>The product provides suggestions for the input error if there are any available.</td>
</tr>
<tr>
<td>Also applies to: Revised Section 508</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• 501 (Web)(Software)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• 504.2 (Authoring Tool)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• 602.3 (Support Docs)</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>3.3.4 Error Prevention (Legal, Financial, Data)</strong> (Level AA)</td>
<td>Not Applicable</td>
<td>The product does not process legal commitments or financial transactions or modify user-controllable data.</td>
</tr>
<tr>
<td>Also applies to: Revised Section 508</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• 501 (Web)(Software)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• 504.2 (Authoring Tool)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• 602.3 (Support Docs)</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>4.1.3 Status Messages</strong> (Level AA 2.1 only)</td>
<td>Not Applicable</td>
<td>The product does not contain status messages as defined by this criterion.</td>
</tr>
<tr>
<td>Also applies to: Revised Section 508 – Does not apply</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Table 3: Success Criteria, Level AAA

<table>
<thead>
<tr>
<th>Criteria</th>
<th>Conformance Level</th>
<th>Remarks and Explanations</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>1.2.6 Sign Language (Prerecorded)</strong> (Level AAA)</td>
<td>Not Evaluated</td>
<td></td>
</tr>
<tr>
<td>Revised Section 508 – Does not apply</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>1.2.7 Extended Audio Description (Prerecorded)</strong> (Level AAA)</td>
<td>Not Evaluated</td>
<td></td>
</tr>
<tr>
<td>Revised Section 508 – Does not apply</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Criteria</td>
<td>Conformance Level</td>
<td>Remarks and Explanations</td>
</tr>
<tr>
<td>----------</td>
<td>-------------------</td>
<td>--------------------------</td>
</tr>
<tr>
<td><strong>1.2.8 Media Alternative (Prerecorded)</strong> (Level AAA)</td>
<td></td>
<td>Not Evaluated</td>
</tr>
<tr>
<td>Revised Section 508 – Does not apply</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>1.2.9 Audio-only (Live)</strong> (Level AAA)</td>
<td></td>
<td>Not Evaluated</td>
</tr>
<tr>
<td>Revised Section 508 – Does not apply</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>1.3.6 Identify Purpose</strong> (Level AAA 2.1 only)</td>
<td></td>
<td>Not Evaluated</td>
</tr>
<tr>
<td>Revised Section 508 – Does not apply</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>1.4.6 Contrast Enhanced</strong> (Level AAA)</td>
<td></td>
<td>Not Evaluated</td>
</tr>
<tr>
<td>Revised Section 508 – Does not apply</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>1.4.7 Low or No Background Audio</strong> (Level AAA)</td>
<td></td>
<td>Not Evaluated</td>
</tr>
<tr>
<td>Revised Section 508 – Does not apply</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>1.4.8 Visual Presentation</strong> (Level AAA)</td>
<td></td>
<td>Not Evaluated</td>
</tr>
<tr>
<td>Revised Section 508 – Does not apply</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>1.4.9 Images of Text (No Exception) Control</strong> (Level AAA)</td>
<td></td>
<td>Not Evaluated</td>
</tr>
<tr>
<td>Revised Section 508 – Does not apply</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>2.1.3 Keyboard (No Exception)</strong> (Level AAA)</td>
<td></td>
<td>Not Evaluated</td>
</tr>
<tr>
<td>Revised Section 508 – Does not apply</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>2.2.3 No Timing</strong> (Level AAA)</td>
<td></td>
<td>Not Evaluated</td>
</tr>
<tr>
<td>Revised Section 508 – Does not apply</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>2.2.4 Interruptions</strong> (Level AAA)</td>
<td></td>
<td>Not Evaluated</td>
</tr>
<tr>
<td>Revised Section 508 – Does not apply</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>2.2.5 Re-authenticating</strong> (Level AAA)</td>
<td></td>
<td>Not Evaluated</td>
</tr>
<tr>
<td>Revised Section 508 – Does not apply</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>2.2.6 Timeouts</strong> (Level AAA 2.1 only)</td>
<td></td>
<td>Not Evaluated</td>
</tr>
<tr>
<td>Revised Section 508 – Does not apply</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Criteria</td>
<td>Conformance Level</td>
<td>Remarks and Explanations</td>
</tr>
<tr>
<td>---------------------------------------------------</td>
<td>-------------------</td>
<td>--------------------------</td>
</tr>
<tr>
<td><strong>2.3.2 Three Flashes</strong> (Level AAA)</td>
<td></td>
<td>Not Evaluated</td>
</tr>
<tr>
<td>Revised Section 508 – Does not apply</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>2.3.3 Animation from Interactions</strong> (Level AAA 2.1 only)</td>
<td></td>
<td>Not Evaluated</td>
</tr>
<tr>
<td>Revised Section 508 – Does not apply</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>2.4.8 Location</strong> (Level AAA)</td>
<td></td>
<td>Not Evaluated</td>
</tr>
<tr>
<td>Revised Section 508 – Does not apply</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>2.4.9 Link Purpose (Link Only)</strong> (Level AAA)</td>
<td></td>
<td>Not Evaluated</td>
</tr>
<tr>
<td>Revised Section 508 – Does not apply</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>2.4.10 Section Headings</strong> (Level AAA)</td>
<td></td>
<td>Not Evaluated</td>
</tr>
<tr>
<td>Revised Section 508 – Does not apply</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>2.5.5 Target Size</strong> (Level AAA 2.1 only)</td>
<td></td>
<td>Not Evaluated</td>
</tr>
<tr>
<td>Revised Section 508 – Does not apply</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>2.5.6 Concurrent Input Mechanisms</strong> (Level AAA 2.1 only)</td>
<td></td>
<td>Not Evaluated</td>
</tr>
<tr>
<td>Revised Section 508 – Does not apply</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>3.1.3 Unusual Words</strong> (Level AAA)</td>
<td></td>
<td>Not Evaluated</td>
</tr>
<tr>
<td>Revised Section 508 – Does not apply</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>3.1.4 Abbreviations</strong> (Level AAA)</td>
<td></td>
<td>Not Evaluated</td>
</tr>
<tr>
<td>Revised Section 508 – Does not apply</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>3.1.5 Reading Level</strong> (Level AAA)</td>
<td></td>
<td>Not Evaluated</td>
</tr>
<tr>
<td>Revised Section 508 – Does not apply</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>3.1.6 Pronunciation</strong> (Level AAA)</td>
<td></td>
<td>Not Evaluated</td>
</tr>
<tr>
<td>Revised Section 508 – Does not apply</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>3.2.5 Change on Request</strong> (Level AAA)</td>
<td></td>
<td>Not Evaluated</td>
</tr>
<tr>
<td>Revised Section 508 – Does not apply</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>3.3.5 Help</strong> (Level AAA)</td>
<td></td>
<td>Not Evaluated</td>
</tr>
</tbody>
</table>
### Revised Section 508 Report

N/A

### Chapter 3: Functional Performance Criteria (FPC)

<table>
<thead>
<tr>
<th>Criteria</th>
<th>Conformance Level</th>
<th>Remarks and Explanations</th>
</tr>
</thead>
<tbody>
<tr>
<td>302.1 Without Vision</td>
<td>Partially Supports</td>
<td>Much of the product is operable without vision. As noted in 1.3.1 Info and Relationships, some structural and hierarchical information is not communicated to screen readers. Also, as noted in 2.1.1 Keyboard, some components are not accessible via keyboard.</td>
</tr>
<tr>
<td>302.2 With Limited Vision</td>
<td>Supports</td>
<td>The product is operable with limited vision.</td>
</tr>
<tr>
<td>302.3 Without Perception of Color</td>
<td>Supports</td>
<td>Color is not used as the only visual means of conveying information, indicating an action, prompting a response, or distinguishing a visual element.</td>
</tr>
<tr>
<td>302.4 Without Hearing</td>
<td>Supports</td>
<td>The product does not require hearing for use.</td>
</tr>
<tr>
<td>302.5 With Limited Hearing</td>
<td>Supports</td>
<td>The product does not require hearing for use.</td>
</tr>
<tr>
<td>302.6 Without Speech</td>
<td>Supports</td>
<td>The product does not require speech for use.</td>
</tr>
<tr>
<td>302.7 With Limited Manipulation</td>
<td>Partially Supports</td>
<td>Most content of the product is operable for users with limited manipulation who rely on keyboard access. As noted in 2.1.1 Keyboard, some components are not accessible via keyboard.</td>
</tr>
</tbody>
</table>
### Chapter 4: Hardware

Notes: Not Applicable - **Oxygen XML Editor** is not a hardware product.

### Chapter 5: Software

#### 501 General

<table>
<thead>
<tr>
<th>Criteria</th>
<th>Conformance Level</th>
<th>Remarks and Explanations</th>
</tr>
</thead>
<tbody>
<tr>
<td>501.1 Scope – Incorporation of WCAG 2.0 AA</td>
<td>See [WCAG 2.x section](on page 65) in WCAG section</td>
<td>See information in WCAG section</td>
</tr>
</tbody>
</table>

#### 502 Interoperability with Assistive Technology

<table>
<thead>
<tr>
<th>Criteria</th>
<th>Conformance Level</th>
<th>Remarks and Explanations</th>
</tr>
</thead>
<tbody>
<tr>
<td>502.2.1 User Control of Accessibility Features</td>
<td>Not Applicable</td>
<td>The product is not platform software.</td>
</tr>
<tr>
<td>502.2.2 No Disruption of Accessibility Features</td>
<td>Supports</td>
<td>The product does not disrupt platform features that are defined in the platform documentation as accessibility features.</td>
</tr>
</tbody>
</table>

#### 502.3 Accessibility Services

<table>
<thead>
<tr>
<th>Criteria</th>
<th>Conformance Level</th>
<th>Remarks and Explanations</th>
</tr>
</thead>
<tbody>
<tr>
<td>502.3.1 Object Information</td>
<td>Partially Supports</td>
<td>Most of the object information can be programmatically determined. As noted in [1.3.1 Info and Relationships](on page 65)</td>
</tr>
<tr>
<td>Criteria</td>
<td>Conformance Level</td>
<td>Remarks and Explanations</td>
</tr>
<tr>
<td>----------------------------------------------</td>
<td>-------------------</td>
<td>--------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>502.3.2 Modification of Object Information</td>
<td>Partially Supports</td>
<td>States and properties that can be set by the user can be set programmatically. As noted in 1.3.1 <em>Info and Relationships</em>, there are few exceptions.</td>
</tr>
<tr>
<td>502.3.3 Row, Column, and Headers</td>
<td>Partially Supports</td>
<td>The insert table feature in the editing area of the product does not communicate information about the headers. Row and column information is presented in a way that can be used by assistive technology.</td>
</tr>
<tr>
<td>502.3.4 Values</td>
<td>Partially Supports</td>
<td>The current values of an object can be programmatically determined. As noted in 1.3.1 <em>Info and Relationships</em>, there are few exceptions. The content from the Grid and schema Design editing modes is not exposed programmatically.</td>
</tr>
<tr>
<td>502.3.5 Modification of Values</td>
<td>Partially Supports</td>
<td>Values that can be set by the user are capable of being set programmatically. As noted in 1.3.1 <em>Info and Relationships</em>, there are few exceptions. The content from the Grid and schema Design editing modes cannot be set programmatically.</td>
</tr>
<tr>
<td>502.3.6 Label Relationships</td>
<td>Partially Supports</td>
<td>Information, structure, and relationships conveyed through presentation can be programmatically determined or are available in text. As noted in 1.3.1 <em>Info and Relationships</em>, there are few exceptions.</td>
</tr>
<tr>
<td>502.3.7 Hierarchical Relationships</td>
<td>Partially Supports</td>
<td>The hierarchical relationships are, in general, accessible programmatically, but as noted in 1.3.1 <em>Info and Relationships</em>, there are few exceptions.</td>
</tr>
<tr>
<td>Criteria</td>
<td>Conformance Level</td>
<td>Remarks and Explanations</td>
</tr>
<tr>
<td>--------------------------------</td>
<td>-------------------</td>
<td>----------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>502.3.8 Text</td>
<td>Partially Supports</td>
<td>The content of text objects, text attributes, and the boundary of text rendered to the screen is programmatically determinable. As noted in 1.3.1 Info and Relationships, there are few exceptions, including the text from the Grid and schema Design modes.</td>
</tr>
<tr>
<td>502.3.9 Modification of Text</td>
<td>Partially Supports</td>
<td>Text can be set programmatically, including through assistive technology. Text in the editor can be modified using the keyboard. As noted in 1.3.1 Info and Relationships, the text content from the Grid and schema Design editing modes cannot be accessed programmatically.</td>
</tr>
<tr>
<td>502.3.10 List of Actions</td>
<td>Supports</td>
<td>Actions that can be executed on an object can be determined programmatically from the context menu or content completion menu.</td>
</tr>
<tr>
<td>502.3.11 Actions on Objects</td>
<td>Supports</td>
<td>Actions on objects can performed by users, including those using assistive technologies.</td>
</tr>
<tr>
<td>502.3.12 Focus Cursor</td>
<td>Partially Supports</td>
<td>The focus location, selection state, and text insertion point information can be determined programmatically. As noted in 1.3.1 Info and Relationships, the content from the Grid and schema Design editing modes is not exposed programmatically.</td>
</tr>
<tr>
<td>502.3.13 Modification of Focus Cursor</td>
<td>Partially Supports</td>
<td>The focus location, selection state and text insertion can be controlled programmatically or through the keyboard. As noted in 1.3.1 Info and Relationships, the content from the Grid and schema Design editing modes cannot be accessed programmatically.</td>
</tr>
</tbody>
</table>
### 502.3.14 Event Notification

- **Supports**
- **Remarks and Explanations**
  > The changes in states and other properties are communicated through notifications that are communicated to assistive technology.

### 502.4 Platform Accessibility Features

- **Not Applicable**
- **Remarks and Explanations**
  > This product is not platform software.

### 503 Applications

#### 503.2 User Preferences

- **Partially Supports**
- **Remarks and Explanations**
  > The product permits some of the user preferences from platform settings. Exceptions include:
    - Cursor thickness is not modified in **Text, Author, and Grid** editing modes.
    - Some elements may not use the platform font size.
  > The product provides custom preferences for changing the color, font type, and font size.

- **Remarks and Explanations**
  > Exceptions include:
    - Cursor thickness is not modified in **Text, Author, and Grid** editing modes.
    - Some elements may not use the platform font size.

#### 503.3 Alternative User Interfaces

- **Not Applicable**
- **Remarks and Explanations**
  > The application does not provide an alternative user interface that functions as assistive technology.

### 503.4 User Controls for Captions and Audio Description

#### 503.4.1 Caption Controls

- **Not Applicable**
- **Remarks and Explanations**
  > The product does not provide controls for volume adjustment.

#### 503.4.2 Audio Description Controls

- **Not Applicable**
- **Remarks and Explanations**
  > The product does not provide controls for program selection.

### 504 Authoring Tools

#### 504.2 Content Creation or Editing (if not authoring tool, enter "not applicable")

- **See the WCAG 2.x section (on page 65)**
- **Remarks and Explanations**
  > See information in WCAG section
<table>
<thead>
<tr>
<th>Criteria</th>
<th>Conformance Level</th>
<th>Remarks and Explanations</th>
</tr>
</thead>
<tbody>
<tr>
<td>504.2.1 Preservation of Information Provided for Accessibility in Format Conversion</td>
<td>Partially Supports</td>
<td>For the main XML vocabulary supported in the application (DITA), the accessibility information is preserved in the generated main formats (WebHelp, PDF).</td>
</tr>
<tr>
<td>504.2.2 PDF Export</td>
<td>Supports</td>
<td>The product is capable of publishing PDF files that conform to PDF/UA-1.</td>
</tr>
<tr>
<td>504.3 Prompts</td>
<td>Partially Supports</td>
<td>For DITA documents, there is an optional Schematron that performs accessibility checks on the content and prompts the authors whenever it detects errors.</td>
</tr>
<tr>
<td>504.4 Templates</td>
<td>Does Not Support</td>
<td>The product does provide several templates. However, these templates offer only minimal structure and do not mark content in ways that promote following the WCAG success criteria. The existing templates can be customized.</td>
</tr>
</tbody>
</table>

Chapter 6: Support Documentation and Services

601.1 Scope

602 Support Documentation

<table>
<thead>
<tr>
<th>Criteria</th>
<th>Conformance Level</th>
<th>Remarks and Explanations</th>
</tr>
</thead>
<tbody>
<tr>
<td>602.2 Accessibility and Compatibility Features</td>
<td>Partially Supports</td>
<td>The documentation of the product lists and explains the accessibility and compatibility features of the product.</td>
</tr>
<tr>
<td>602.3 Electronic Support Documentation</td>
<td>Partially Supports</td>
<td>The self-service documentation is generated with Oxygen XML WebHelp. You can find its VPAT statement here.</td>
</tr>
<tr>
<td>602.4 Alternate Formats for Non-Electronic Support Documentation</td>
<td>Not Applicable</td>
<td>Documentation is not provided in non-electronic formats.</td>
</tr>
</tbody>
</table>

603 Support Services
<table>
<thead>
<tr>
<th>Criteria</th>
<th>Conformance Level</th>
<th>Remarks and Explanations</th>
</tr>
</thead>
<tbody>
<tr>
<td>603.2 Information on Accessibility and Compatibility Features</td>
<td>Supports</td>
<td>The support services cover the accessibility features.</td>
</tr>
<tr>
<td>603.3 Accommodation of Communication Needs</td>
<td>Supports</td>
<td>Support is provided over a variety of channels including email and phone.</td>
</tr>
</tbody>
</table>

**Legal Disclaimer**

This report describes Oxygen XML Editor ability to support the stated VPAT Standards/Guidelines, subject to Syncro Soft's interpretation of the same. This accessibility report is provided for informational purposes only, and the contents hereof are subject to change without notice. SYNCRO SOFT MAKES NO WARRANTIES, EXPRESS OR IMPLIED, IN THIS DOCUMENT. For more information regarding the accessibility status, please contact us at sales@oxygenxml.com.

© 2020 Syncro Soft SRL. All rights reserved.
3. Installation

Oxygen XML Editor is available on Windows, Linux, and macOS and there are a variety of methods and options for installing and running Oxygen XML Editor on your system or server. This section also includes information about registering, transferring, or releasing licenses, upgrading, installing add-ons, and uninstalling.

Choosing How Oxygen XML Editor Runs

You can install Oxygen XML Editor to run in several ways:

• As a desktop application (running standalone or as an Eclipse plugin) on Windows, Linux, or macOS.
• As a desktop application (running standalone or as an Eclipse plugin) on a Unix or Linux server or on Windows Terminal Server.

Choosing an Installer

You also have a choice of several different installers:

• The native installer for your platform (Windows, Linux, or macOS).
• On Windows and Linux, the native installer can also run in unattended mode.

Choosing a License Option

You must obtain and register a license key (on page 101) to run Oxygen XML Editor.

You can choose from two types of licenses:

• A named-user license, which can be used by a single person on multiple computers.
• A floating license, which can be used by different people at different times. Only one person can use a floating license at a time.

Upgrading, transferring, and uninstalling.

You can also upgrade (on page 118) Oxygen XML Editor, transfer a license (on page 106), or uninstall (on page 122) Oxygen XML Editor.

Getting help with installation

If you need help, email support at: support@oxygenxml.com.
Installing Oxygen XML Editor on Windows

System Requirements

Operating Systems

The product has been fully tested on Windows 10. The latest version of Oxygen XML Editor might work on other versions of Windows, but they have not been officially tested.

CPU

- Minimum - Intel/AMD Dual-core class CPU, 2 GHz
- Recommended - Quad-core class processor

Memory

- Minimum - 3 GB of RAM
- Recommended - 8 GB of RAM

Storage

- Minimum - 1 GB free disk space
- Recommended - 2 GB free disk space

Java

Oxygen XML Editor only supports official and stable Java Virtual Machines with version 1.8 from Oracle. If you use the native Windows installer, Oxygen XML Editor will be installed with its own copy of Java with the specific update version that has been thoroughly tested.

All Platforms Package

If you use the all platforms package, your system must have a compatible Java 1.8 virtual machine installed. To see the exact Java update version that is supported, go to www.oxygenxml.com, navigate to the Download page for the particular product you are installing, and click on the tab for your particular platform.

Note:

Oxygen XML Editor may work with other versions of Java, but since Oxygen XML Editor has only been thoroughly tested with specific versions, there is no guarantee that it will be stable with any other Java version.

Oxygen XML Editor uses the following rules to determine which installed version of Java to use:
1. If you install using the native Windows installer, which installs a version of Java as part of the Oxygen XML Editor installation, the version in the jre subdirectory of the installation directory is used.

2. Otherwise, if the Windows environment variable \texttt{JAVA_HOME} is set, Oxygen XML Editor uses the Java version pointed to by this variable.

3. Otherwise, the version of Java pointed to by your \texttt{PATH} environment variable is used.

If you run Oxygen XML Editor using the batch file, \texttt{oxygen.bat}, you can edit the batch file to specify a particular version to use.

**Windows Installer**

To install Oxygen XML Editor using the Windows installer, follow these steps:

1. Make sure that your system meets the system requirements (on page 86).
2. Download the Windows installer.
3. [Optional] Validate the integrity of the downloaded file by checking it against the MD5 sum published on the download page.
4. Run the installer and follow the instructions in the installation program.
5. Start Oxygen XML Editor using one of the following methods:
   - Use one of the shortcuts created by the installer.
   - Run \texttt{oxygen.bat}, which is located in the installation directory.
6. To license your copy of Oxygen XML Editor, go to \texttt{Help > Register} and enter your license information (on page 101).

**Windows Unattended Installation**

You can run the installation in unattended mode by running the installer from the command line with the \texttt{-q} parameter. By default, running the installer in unattended mode installs Oxygen XML Editor with the default options and does not overwrite existing files. You can change various options for the unattended installer using the installer command-line parameters.

**Windows Installer Command-Line Reference**

The Oxygen XML Editor installer for Windows supports a variety of command-line parameters.

**Commonly Used Command-Line Parameters**

The Oxygen XML Editor installer supports the following commonly used command-line parameters:

\texttt{-q}

Instructs the installer to run in unattended mode. The installer will not prompt the user for input during the install. Default settings will be used for all options unless a \texttt{response.varfile (on page 90)} is specified using the \texttt{-varfile} option.
-overwrite

In unattended mode, the installer does not overwrite files with the same name if a previous version of the Oxygen XML Editor is installed in the same folder. The -overwrite parameter added after the -q parameter forces the overwriting of these files.

-console

Displays a console during an unattended installation.

Note:

If you want the installer to run in the foreground, you need to use the start/wait command (for example, start/wait oxygen.exe -q -console). Otherwise, it will run in the background.

-varfile

Specifies the location of a response.varfile (on page 90), normally to be used during an unattended installation.

-c

Allows users to configure the installation by inputting answers to installation questions in the command line.

Tip:

Using this parameter is the best way to use the installer for people who are visually impaired.

-V[variable name]=[variable value]

This command-line parameter can be used to define any of the variables listed below to be used by an installation.

EXAMPLE:

```
oxygen.exe -q -overwrite -console -VautoVersionChecking=false
```

Command-Line Variables for Preconfiguring License Server Details

The Oxygen XML Editor installer also supports the following command-line variables used for preconfiguring license server details:

autoVersionChecking

Used for automatic version checking. Possible values are true (default) or false.

backup.license.servlet.url

Specifies the URL of the backup HTTP license server.
backup.license.servlet.user.name

Specifies the user name for the backup HTTP license server.

backup.license.servlet.password

Specifies the password for the backup HTTP license server, in clear form (will be stored encrypted).

backup.license.servlet.password.encrypted

Specifies the password for the HTTP license server, in encrypted form. Can be obtained from an entry with the same name in an existing license.xml file (found in: \[user_home_directory]\AppData\Roaming\com.oxygenxml).

downloadResources

Used to download resources (links to video demonstrations, webinars, and upcoming events) from https://www.oxygenxml.com to populate the application welcome screen. Possible values are true (default) or false.

license.servlet.url

Specifies the URL of the HTTP license server.

license.servlet.user.name

Specifies the user name for the HTTP license server.

license.servlet.password

Specifies the password for the HTTP license server, in clear form (will be stored encrypted).

license.servlet.password.encrypted

Specifies the password for the HTTP license server, in encrypted form. Can be obtained from an entry with the same name in an existing license.xml file (found in: \[user_home_directory]\AppData\Roaming\com.oxygenxml).

reportProblem

Used to report a problem encountered while using Oxygen XML Editor. Possible values are true (default) or false.

EXAMPLE:

```
oxygen.exe "-Vlicense.servlet.url=http://main.licenseserver:8080/oXygenLicenseServlet/license-servlet"
   "-Vlicense.servlet.user.name=user" "-Vlicense.servlet.password=mypass"
   "-Vbackup.license.servlet.url=http://backup.licenseserver:8080/oXygenLicenseServlet/license-servlet"
   "-Vbackup.license.servlet.user.name=user" "-Vbackup.license.servlet.password=mypass"
```
Windows Installer *response.varfile*

The Oxygen XML Editor installer for Windows also creates a file called *response.varfile*, which records the choices that the user made when running the installer interactively. The generated response file is found in the `[OXYGEN_INSTALL_DIR]/.install4j` folder. You can use the *response.varfile* to set the options for an unintended install *(on page 87)*. For more information about the *response.varfile* format, see install4j site.

**Variables (can be used in the *response.varfile* or from the command line)**

The following variables are supported in the *response.varfile* (or from the command line):

- **autoVersionChecking**
  Used for automatic version checking. Possible values are *true* (default) or *false*.

- **downloadResources**
  Used to download resources (links to video demonstrations, webinars, and upcoming events) from [https://www.oxygenxml.com](https://www.oxygenxml.com) to populate the application welcome screen. Possible values are *true* (default) or *false*.

- **reportProblem**
  Used to report a problem encountered while using Oxygen XML Editor. Possible values are *true* (default) or *false*.

---

**Installing Oxygen XML Editor on macOS**

**System Requirements**

**Operating system**

The product has been fully tested on macOS 10.15 (Catalina), 11 (Big Sur), and 12 (Monterey). The latest version of Oxygen XML Editor might work on older versions of macOS, but they have not been officially tested.

---

**Note:**

The OpenJDK JRE bundled with Oxygen XML Editor is created using *jlink* from Oracle's OpenJDK distribution and *jmods* of the corresponding version of JavaFX by Gluon.

**CPU**

- Minimum - Intel-based Dual-core Mac, 2 GHz
- Recommended - Quad-core class processor

**Memory**
• Minimum - 3 GB of RAM  
• Recommended - 8 GB of RAM

Storage

• Minimum - 1 GB free disk space  
• Recommended - 2 GB free disk space

macOS Installation

To install Oxygen XML Editor on macOS, follow these steps:

1. **Download** the macOS installation package (oxygen.dmg).
2. [Optional] Validate the integrity of the downloaded file by checking it against the MD5 sum published on the download page.
3. Double-click the oxygen.dmg disk image file to mount it.
4. Drag/Copy the Oxygen XML Editor folder to your /Applications folder (or another location if you wish).

![Warning]

If you receive a warning that an Oxygen XML Editor installation folder already exists in the Applications folder, do not attempt to merge the two installations. Instead, move the old installation folder to the trash bin before installing the application. If you are prompted to Replace the old folder, cancel the installation, move the old folder to the trash bin, and restart the installation process.

![Important]

Do not copy the files/folders from within the Oxygen XML Editor folder (always copy the folder itself), otherwise you will omit invisible files/folders and the application may no longer start.

5. Start Oxygen XML Editor, using one of the following methods:
   - Double-click Oxygen XML Editor.app.
   - Run `sh oxygen.sh` in the command-line interface.
6. To license your copy of Oxygen XML Editor, go to Help > Register to enter your license key (on page 101).

macOS Unattended Installation

To install Oxygen XML Editor on macOS in unattended mode, follow these steps:

1. **Download** the macOS installation package (oxygen.dmg).
2. Mount the oxygen.dmg file in the command line.
3. Copy the oxygen folder for the particular version from the mounted volume to the Applications folder (or another folder where you want to install it), as in the following example:

```bash
cp -aR "/Volumes/Oxygen XML Editor 21.0/Oxygen XML Editor" /Applications/
```

4. Eject the mounted disc image:

```bash
hdiutil detach "/Volumes/Oxygen XML Editor 21.0"
```

### Installing Oxygen XML Editor on Linux

#### System Requirements

**Operating System**

The product has been fully tested on Ubuntu 20.04. The latest version of Oxygen XML Editor might work on other flavors/versions of Linux, but they have not been officially tested.

**CPU**

- Minimum - Intel/AMD Dual-core class CPU, 2 GHz
- Recommended - Quad-core class processor

**Memory**

- Minimum - 3 GB of RAM
- Recommended - 8 GB of RAM

**Storage**

- Minimum - 1 GB free disk space
- Recommended - 2 GB free disk space

**Java**

Oxygen XML Editor only supports official and stable Java Virtual Machines with version 1.8 from Oracle. If you use the Linux installer, Oxygen XML Editor will be installed with its own copy of Java with the specific update version that has been thoroughly tested.

**All Platforms Package**

If you use the all platforms package, your system must have a compatible Java 1.8 virtual machine installed. To see the exact Java update version that is supported, go to [www.oxygenxml.com](http://www.oxygenxml.com), navigate to the [Download](http://www.oxygenxml.com) page for the particular product you are installing, and click on the tab for your particular platform.
Note:
Oxygen XML Editor may work with other versions of Java, but since Oxygen XML Editor has only been thoroughly tested with specific versions, there is no guarantee that it will be stable with any other Java version.

Attention:
Oxygen XML Editor does not work with the GNU libgcj Java Virtual Machine.

Oxygen XML Editor uses the following rules to determine which installed version of Java to use:

1. If you used the Linux installer, which installs a version of Java as part of the Oxygen XML Editor installation, the version in the jre subdirectory of the installation directory is used.
2. Otherwise, if the Linux environment variable JAVA_HOME is set, Oxygen XML Editor uses the Java version pointed to by this variable.
3. Otherwise, the version of Java pointed to by your PATH environment variable is used.

You can also change the version of the Java Virtual Machine that runs Oxygen XML Editor by editing the script file, oxygen.sh. Go to the Java command at the end of the script file and specify the full path to the Java executable of the desired JVM version. For example:

```
/usr/bin/jre1.8.0_121/bin/java -Xmx1024m ...
```

**X.org**

The version of Java bundled with Oxygen XML Editor requires X.org (Wayland is not supported).

**Linux Installer**

To install Oxygen XML Editor using the Linux installer, follow these steps:

1. Make sure that your system meets the system requirements (on page 92).
2. Download the Linux installer.
3. [Optional] Validate the integrity of the downloaded file by checking it against the MD5 sum published on the download page.
4. Run the installer and follow the instructions in the installation program.
5. Start Oxygen XML Editor using one of the following methods:
   - Use the `oxygen` shortcut created by the installer.
   - From a command line, type `sh oxygen.sh`. This file is located in the installation folder.

6. To license your copy of Oxygen XML Editor go to Help > Register and enter your license information (on page 101).

**Linux Unattended Installation**

You can run the installation in unattended mode by running the installer from the command line with the `-q` parameter. By default, running the installer in unattended mode installs Oxygen XML Editor with the default options and does not overwrite existing files. You can change various options for the unattended installer using the installer command-line parameters.

**Linux Installer Command-Line Reference**

The Oxygen XML Editor installer for Linux supports a variety of command-line parameters.

**Commonly Used Command-Line Parameters**

The Oxygen XML Editor installer supports the following commonly used command-line parameters:

- `-q`
Instructs the installer to run in unattended mode. The installer will not prompt the user for input during the install. Default settings will be used for all options unless a response.varfile (on page 96) is specified using the -varfile option.

-**overwrite**

In unattended mode, the installer does not overwrite files with the same name if a previous version of the Oxygen XML Editor is installed in the same folder. The -**overwrite** parameter added after the -q parameter forces the overwriting of these files.

-**console**

Displays a console during the installation.

-**varfile**

Specifies the location of a response.varfile (on page 96), normally to be used during an unattended installation.

-**V**

Used to define a variable parameter (on page 96) to be used by an installation.

**EXAMPLE:**

```
oxygen.sh -q -overwrite -console -VautoVersionChecking=false
```

**Command-Line Parameters for Preconfiguring License Server Details**

The Oxygen XML Editor installer also supports the following command-line parameters used for preconfiguring license server details:

- **license.servlet.url**
  
  Specifies the URL of the HTTP license server.

- **license.servlet.user.name**
  
  Specifies the user name for the HTTP license server.

- **license.servlet.password**
  
  Specifies the password for the HTTP license server, in clear form (will be stored encrypted).

- **license.servlet.password.encrypted**
  
  Specifies the password for the HTTP license server, in encrypted form. Can be obtained from an entry with the same name in an existing license.xml file (found in: [user_home_directory]/.com.oxygenxml).

- **backup/license.servlet.url**
  
  Specifies the URL of the backup HTTP license server.

- **backup/license.servlet.user.name**
Specifies the user name for the backup HTTP license server.

**backup.license.servlet.password**

Specifies the password for the backup HTTP license server, in clear form (will be stored encrypted).

**backup.license.servlet.password.encrypted**

Specifies the password for the HTTP license server, in encrypted form. Can be obtained from an entry with the same name in an existing `license.xml` file (found in: `[user_home_directory]/.com.oxygenxml`).

**EXAMPLE:**

```sh
oxxygen.sh "-Vlicense.servlet.url=http://main.licenseserver:8080/oXygenLicenseServlet/license-servlet"
-"-Vlicense.servlet.user.name=user" 
-"Vlicense.servlet.password=mypass"
-"Vbackup.license.servlet.url=http://backup.licenseserver:8080/oXygenLicenseServlet/license-servlet"
-"Vbackup.license.servlet.user.name=user" 
-"Vbackup.license.servlet.password=mypass"
```

**Linux Installer response.varfile**

The Oxygen XML Editor installer for Linux also creates a file called `response.varfile`, which records the choices that the user made when running the installer interactively. The generated response file is found in the `[OXYGEN_INSTALL_DIR]/.install4j` folder. You can use the `response.varfile` to set the options for an unintended install ([on page 94](#)). For more information about the `response.varfile` format, see the [install4j site](#).

**Variable Parameters (can be used in the `response.varfile` or from the command line)**

The following variable parameters are supported in the `response.varfile` (or from the command line):

**autoVersionChecking**

Used for automatic version checking. Possible values are `true` (default) or `false`.

**reportProblem**

Used to report a problem encountered while using Oxygen XML Editor. Possible values are `true` (default) or `false`.

**downloadResources**

Used to download resources (links to video demonstrations, webinars, and upcoming events) from [https://www.oxygenxml.com](https://www.oxygenxml.com) to populate the application welcome screen. Possible values are `true` (default) or `false`. 
Installing Oxygen XML Editor on Windows Server

System Requirements

Operating systems

Windows Server 2012 or Windows Server 2012 R2

CPU

- Minimum - Intel/AMD Dual-core class CPU, 2 GHz
- Recommended - Quad-core class processor

Memory

- Minimum values per user - 1 GB of RAM
- Recommended values per concurrent user - 2 GB of RAM

Storage

- Minimum - 1 GB free disk space
- Recommended - 2 GB free disk space

Java

Oxygen XML Editor only supports official and stable Java Virtual Machines with version 1.8 from Oracle. If you use the native Windows installer, Oxygen XML Editor will be installed with its own copy of Java with the specific update version that has been thoroughly tested.

All Platforms Package

If you use the all platforms package, your system must have a compatible Java 1.8 virtual machine installed. To see the exact Java update version that is supported, go to www.oxygenxml.com, navigate to the Download page for the particular product you are installing, and click on the tab for your particular platform.

Note:

Oxygen XML Editor may work with other versions of Java, but since Oxygen XML Editor has only been thoroughly tested with specific versions, there is no guarantee that it will be stable with any other Java version.

Oxygen XML Editor uses the following rules to determine which installed version of Java to use:
1. If you install using the native Windows installer, which installs a version of Java as part of the Oxygen XML Editor installation, the version in the jre subdirectory of the installation directory is used.

2. Otherwise, if the Windows environment variable JAVA_HOME is set, Oxygen XML Editor uses the Java version pointed to by this variable.

3. Otherwise, the version of Java pointed to by your PATH environment variable is used.

If you run Oxygen XML Editor using the batch file, oxygen.bat, you can edit the batch file to specify a particular version to use.

Windows Installer

To install Oxygen XML Editor using the Windows installer, follow these steps:

1. Make sure that your system meets the system requirements (on page 97).
2. Download the Windows installer.
3. [Optional] Validate the integrity of the downloaded file by checking it against the MD5 sum published on the download page.
4. Run the installer and follow the instructions in the installation program.
5. Start Oxygen XML Editor using one of the following methods:
   ◦ Use one of the shortcuts created by the installer.
   ◦ Run oxygen.bat, which is located in the installation directory.
6. To license your copy of Oxygen XML Editor go to Help > Register and enter your license information (on page 101).

Configuring Windows Terminal Server

1. Install Oxygen XML Editor on the server and make its shortcuts available to all users.
2. Make sure you allocate sufficient memory to Oxygen XML Editor by adding the -Xmx parameter either in the .bat startup script (on page 339), or in the .vmoptions configuration file (on page 342) (if you start it from an executable launcher).

Installing Oxygen XML Editor on a Linux / UNIX Server

System Requirements

Operating system

The product has been fully tested on Ubuntu 20.04. The latest version of Oxygen XML Editor might work on other flavors/versions of Linux, but they have not been officially tested.

CPU

- Minimum - Intel/AMD Dual-core class CPU, 2 GHz
- Recommended - Quad-core class processor
Memory

- Minimum - 3 GB of RAM
- Recommended - 8 GB of RAM

Storage

- Minimum - 1 GB free disk space
- Recommended - 2 GB free disk space

Java

Oxygen XML Editor only supports official and stable Java Virtual Machines with version 1.8 from Oracle. If you use the Linux installer, Oxygen XML Editor will be installed with its own copy of Java with the specific update version that has been thoroughly tested.

**All Platforms Package**

If you use the all platforms package, your system must have a compatible Java 1.8 virtual machine installed. To see the exact Java update version that is supported, go to [www.oxygenxml.com](http://www.oxygenxml.com), navigate to the Download page for the particular product you are installing, and click on the tab for your particular platform.

**Note:**

Oxygen XML Editor may work with other versions of Java, but since Oxygen XML Editor has only been thoroughly tested with specific versions, there is no guarantee that it will be stable with any other Java version.

**Attention:**

Oxygen XML Editor does not work with the GNU libgcj Java Virtual Machine.

Oxygen XML Editor uses the following rules to determine which installed version of Java to use:

1. If you used the Linux installer, which installs a version of Java as part of the Oxygen XML Editor installation, the version in the jre subdirectory of the installation directory is used.
2. Otherwise, if the Linux environment variable JAVA_HOME is set, Oxygen XML Editor uses the Java version pointed to by this variable.
3. Otherwise, the version of Java pointed to by your PATH environment variable is used.

You can also change the version of the Java Virtual Machine that runs Oxygen XML Editor by editing the script file, oxygen.sh. Go to the Java command at the end of
the script file and specify the full path to the Java executable of the desired JVM version. For example:

```
/usr/bin/jre1.8.0_121/bin/java -Xmx1024m ...
```

**Linux Installer**

To install Oxygen XML Editor using the Linux installer, follow these steps:

1. Make sure that your system meets the system requirements (on page 98).
2. Download the Linux installer.
3. [Optional] Validate the integrity of the downloaded file by checking it against the MD5 sum published on the download page.
4. Run the installer and follow the instructions in the installation program.

**Note:**

For example, open a shell, cd to the installation directory, and at the prompt type `sh ./oxygen-32bit.sh` or `sh ./oxygen-64bit.sh`, depending on which installer you downloaded.

5. Start Oxygen XML Editor using one of the following methods:
   - Use the oxygen shortcut created by the installer.
   - From a command line, type `sh oxygen.sh`. This file is located in the installation folder.
6. To license your copy of Oxygen XML Editor go to Help > Register and enter your license information (on page 101).

**Unix/Linux Server Configuration**

1. Install Oxygen XML Editor on the server and make sure the oxygen.sh script is executable and the installation directory is in the PATH of the users that need to use the application.
2. Make sure you allocate sufficient memory to Oxygen XML Editor by setting an appropriate value for the `-Xmx` parameter in the .sh startup script.

**Note:**

The default value of the `-Xmx` parameter is 1 GB. To avoid performance issues with large documents (on page 2856), you may need to adjust it.

3. Make sure the X server processes located on the workstations allow connections from the server host.
   - For this, use the xhost command.
4. Start telnet (or ssh) on the server host.
5. Start an xterm process with the display parameter set on the current workstation. For example: `xterm -display workstationip:0.0`.
6. Start Oxygen XML Editor by typing `sh oxygen.sh` from the command line. This file is located in the installation folder.
Site-Wide Deployment

If you are deploying Oxygen XML Editor for a group, there are various things you can do to customize Oxygen XML Editor for your users and to make the deployment more efficient.

Creating custom default options

You can create a custom set of default options (on page 309) for Oxygen XML Editor. These will become the default options for each of your users, replacing the normal default settings. Users can still set options to suit themselves in their own copies of Oxygen XML Editor, but if they choose to reset their options to defaults, the custom defaults that you set will be used.

Creating default project files

Oxygen XML Editor project files (on page 398) are used to configure a project. You can create and deploy default project files (on page 399) for your projects so that your users will have a preconfigured project file to begin work with.

Shared project files

Rather than each user having their own project file, you can create and deploy shared project files (on page 415) so that all users share the same project configuration and settings and automatically inherit all project changes.

Using the unattended installer

You can speed up the installation process by using the unattended installer for Windows (on page 87) or Linux (on page 94) installs.

Using floating licenses

If you have a number of people using Oxygen XML Editor on a part-time basis or in different time zones, you can use a floating license (on page 103) so that multiple people can share a license.

Licensing

Oxygen XML Editor is not free software. To activate and use Oxygen XML Editor, you need a license.

For demonstration and evaluation purposes, a time limited license is available upon request at https://www.oxygenxml.com/register.html. This license is supplied at no cost for a period of 30 days from the date of issue. During this period, the software is fully functional, enabling you to test all its functionality. To continue using the software after the trial period, you must purchase a permanent license.

Choosing a License Type

You can use one of the following license types with Oxygen XML Editor:
1. A **Named-User License (on page 102)** may be used by a single **Named User (on page 3221)** on one or more computers. Named-user licenses are not transferable to a new **Named User**. If you order multiple named-user licenses, you will receive a single license key good for a specified number of **named users**. It is your responsibility to keep track of the **named users** that each license is assigned to.

2. A **Floating License (on page 103)** may be used by any user on any machine. However, the total number of copies of Oxygen XML Editor in use at one time must not be more than the number of floating licenses available. A user who runs two different distributions of Oxygen XML Editor (for example, Standalone and Eclipse Plugin) at the same time on the same computer, consumes a single floating license.

3. A **Subscription license (on page 102)** that allows you to use the application for a specific period of time (either 6 months or 1 year). This type of license is user-based and is covered by a Support and Maintenance Pack, which means that during the subscription period you will get free upgrades to all major and minor releases and priority technical support.

4. A special **Academic Group License** (*Classroom, Department, or Site* license) may be used by students and teachers in academic institutions. These licenses provide a cost effective way of getting access to Oxygen XML Editor for learning purposes.

For definitions and legal details of the license types, consult the End-User License Agreement available at https://www.oxygenxml.com/eula.html.

### Obtaining a License Key

You can obtain a license key for Oxygen XML Editor in one of the following ways:

- You can purchase one or more licenses from the Oxygen XML Editor website at https://www.oxygenxml.com/buy.html or through one of the authorized resellers. A license key will be sent to you by email.
- If your company or organization has already purchased licenses, contact your license administrator to obtain a license key or configuration details to connect to a license server.
- If you purchased a subscription and you received a registration code, you can use it to obtain a license key from https://www.oxygenxml.com/registerCode.html. A license key will be sent to you by email.
- If you want to evaluate the product, you can obtain a trial license key for 30 days from the Oxygen XML Editor website at https://www.oxygenxml.com/register.html.

### Register a Named-User or Subscription License

To register a **Named-User License** or **Subscription License** on a machine owned by the **Named User (on page 3221)**, follow these steps:

1. Purchase a license from the **Oxygen XML Editor website**. You will receive an email that contains your license key.
2. Save a backup copy of your email message that contains the new license key.
3. Start Oxygen XML Editor.
If this is a new installation of Oxygen XML Editor, the registration dialog box is displayed. If the registration dialog box is not displayed, go to Help > Register.

**Figure 11. License Registration Dialog Box**

4. Select **Use a license key** as the licensing method.

Note: If your license key has 20 or more licenses, you must use a license server *(on page 106)* instead.

5. Paste your license key into the registration dialog box. The license key is composed of nine lines of text between two text markers.
6. Click **OK**.

Related information

Oxygen XML Editor End-User License Agreement

**Floating Licenses**

The floating license type is commonly used by organizations that have a large number of infrequent users who do not need to use Oxygen XML Editor all at the same time. Instead of each user having their own individual license key to use at their discretion, there are a pool of licenses that are used one at a time by various users upon request.
To use floating licenses, a license server is required and the license key needs to be activated. Your system administrator will most likely be responsible for setting up the license server (on page 106). Then you will need to request a floating license from the server (on page 104). This process is designed to help you comply with the Oxygen End-User License Agreement (EULA). This means that the license key will be locked to a particular license server deployment, and the same license key cannot be used with any other license server.

For information about releasing and returning a floating license to the pool for other users, see Releasing a Floating License (on page 105).

For information about reserving (or locking) a floating license so that it does not get returned to the pool, see Reserving a Floating License (on page 106).

**Requesting a Floating License from an HTTP License Server**

**How to Request a Floating License**

To request a floating license from an HTTP license server, follow this procedure:

1. Contact your server administrator to make sure the license server has already been set up and get network address and login details for the license server.
2. Start Oxygen XML Editor.
3. Go to Help > Register.
   
   **Step Result:** The license registration dialog box is displayed.
4. Choose Use a license server as licensing method.
5. Select HTTP/HTTPS Server as server type.
6. In the URL field, enter the address of the license server. The URL address has the following format: 

   http://hostName:port/oXygenLicenseServlet/license-servlet.
7. Complete the User and Password fields.
8. Click the OK button.

**Result:** If a floating license is available, it is registered in Oxygen XML Editor. To display the license details, open the About dialog box from the Help menu. If a floating license is not available, you will get a message listing the users currently using floating licenses.

**How to Register Floating Licenses for Additional Users**

If you are an administrator and you want to register floating licenses for multiple users without having to open Oxygen XML Editor on each machine to manually configure the registration details one by one, you can use the following procedure:
1. Reset the registration details in Oxygen XML Editor:
   a. Go to Help > Register.
   b. Click OK without entering any information in this dialog box.
   c. Click Reset and restart the application.

2. Register the license using one of the floating license registration procedures (on page 103).

   **Step Result:** A license.xml file is created.

3. Copy the license.xml file from the preferences directory (on page 125) and place it in the installation folder on each machine to be registered.

Related information

Setting up an HTTP License Server (Floating or Named-User Licenses) (on page 107)

### Releasing a Floating License

The floating license you are using will be released and returned to the pool if any of the following occur:

- The connection with the license server is lost.
- You exit the application running on your machine, and no other copies of Oxygen XML Editor running on your machine are using your floating license.
- You register a Named User (on page 3221) license with your copy of Oxygen XML Editor, and no other copies of Oxygen XML Editor running on your machine are using your floating license.
- Your computer idles for more than 2 hours.
- Your system administrator manually revokes the license (on page 113).

**Tip:**

To prevent your floating license from being released, you can use the Lock floating license action available in the Help menu. You can use the same action to unlock the license. Note that your system administrator can also unlock your license (on page 113).

To release a floating license on demand, follow these steps:

1. Go to Help > Register.
   - The license registration dialog box is displayed.
2. The license key field should be empty (this is normal). If it is not empty, delete any text in the field.
3. Make sure the Use a license key option is selected.
4. Click OK.
   - A dialog box is displayed asking if you want to reset your license key.
5. Select between:
- **Use the last one** - Falls back to your previous license key. Use this option if you want to release a floating license and revert to a Named User (on page 3221) license.
- **Reset** - Removes your license key from your user account on the current computer.

The Reset button erases all the licensing information. To complete the reset operation, close and restart Oxygen XML Editor.

**Reserving a Floating License**

There are instances where you might want to reserve or lock a floating license. For example, you could lock a floating license if you want to use your floating license offline while traveling.

To reserve/lock a floating license, follow these steps:

1. Select Lock floating license from the Help menu.
2. Click OK.

Your floating license is now locked. You can use the same action to unlock the license or you can contact your system administrator to unlock it.

**Transferring a License Key**

If you want to transfer your Oxygen XML Editor license key to another computer (for example, if you are disposing of your old computer or transferring it to another person), you must first unregister your license. You can then register your license (on page 101) on the new computer in the normal way.

To unregister a license, prior to transferring it, follow this procedure:

1. Go to Help > Register.
   - The license registration dialog box is displayed.
2. Make sure the Use a license key option is selected.
3. The license key field should be empty (this is normal). If it is not empty, delete any text in the field.
4. Click the Remove button at the bottom-right corner of the dialog box.
   - A confirmation message is displayed asking if you want to remove your license key.
5. Select between:
   - **Yes** - Removes your license key from your user account on the current computer.
   - **No** - Falls back to your previous license key, if applicable.

**License Servers**

**Installing a License Server to Manage Licenses**

If you are using floating licenses or a large number of user-based licenses (20 or more) for Oxygen XML Editor, you must set up an Oxygen XML Editor HTTP server (on page 107).
Note:
Oxygen XML Editor version 17 or higher requires a license server version 17 or higher. License servers version 20.1 or higher can be used with any version of a floating or named-user license key.

Activating License Keys

To help you comply with the Oxygen XML Editor EULA (terms of licensing), all floating or named-user licenses require activation. This means that the license key will be locked to a particular license server deployment and no multiple uses of the same license key are possible.

During the activation process, a code that uniquely identifies your license server deployment is sent to the Oxygen XML Editor servers, which in turn will sign the license key.

Splitting or Combining License Keys to Work with Your License Servers

A license server can only manage one license key. If you have multiple license keys for the same Oxygen XML Editor version and you want to have all of them managed by the same server, or if you have a multiple-user floating license and you want to split it between two or more license servers, contact the Oxygen support team and ask for a new license key.

Setting up an HTTP License Server (Floating or Named-User Licenses)

Restriction:
The floating license server does not work with Docker containers.

The HTTP License Server is available in several distributions, tailored for covering various deployment configurations:

- **Windows installer** ([on page 108](#)) - Easy-to-use Windows installation wizard. Requires elevated permissions to run it.
- **All-platform distribution** ([on page 108](#)) - Script-based deployment that does not require elevated permissions to run it. Provides scripts for Windows, macOS, and Linux.
- **Web Archive (WAR) distribution** ([on page 109](#)) - Provides more flexibility in your deployment configuration, but it requires an existing HTTP server (such as Apache Tomcat).

HTTP License Server System Requirements

### Table 2. Requirements

<table>
<thead>
<tr>
<th>Hardware</th>
<th>Specification</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>1 core</td>
</tr>
<tr>
<td>RAM</td>
<td>512 MB/Linux OS, 1 GB/Windows OS (256 MB available memory)</td>
</tr>
</tbody>
</table>
### Table 2. Requirements (continued)

<table>
<thead>
<tr>
<th>Hardware</th>
<th>Specification</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hard Disk Space</td>
<td>500 MB</td>
</tr>
<tr>
<td>Network Requirements</td>
<td><strong>Network interfaces stay unchanged (static MAC addresses) after activation</strong></td>
</tr>
<tr>
<td>Server OS Requirements</td>
<td>Linux or Windows</td>
</tr>
<tr>
<td>Antivirus and Firewall Requirements</td>
<td>Allow access to the configured TCP port (default 8080)</td>
</tr>
</tbody>
</table>

### Installing the HTTP License Server Installer Distribution for Windows

1. Download the HTTP license server installer from the [HTTP License Server website](http://www.httplicenser.com).
2. Run the installer and follow the on-screen instructions.
3. You must configure two sets of credentials:
   a. **Administrator credentials** - Used for accessing the Oxygen XML Editor license server administrative interface.
   b. **Standard user credentials** - Used by an Oxygen XML Editor application to connect to the license server.
4. You can choose to change the default 8080 port the server runs on. If you need to change the port after the installation, you can do it by editing the following `vmoptions` file: `"oxygenHTTPLicenseServer.vmoptions`.
5. Optionally, you can choose to install the server as a Windows service. In this case, you can choose the name of the Windows service.

#### Tip:
In case you run into issues, the license server log file is located in:

```
[Installation_Directory]\work\logs\oxygenLicenseServlet.log
```

### Installing the HTTP License Server All-Platform Distribution

1. **Prerequisite** Java 8 or later must be installed.
2. Download the HTTP license server all-platform archive from the [HTTP License Server website](http://www.httplicenser.com).
3. Unpack the archive.
4. Run the license server scripts suitable for your operating system (`licenseServer.bat` for Windows or `licenseServer.sh` for Linux and macOS).

#### Note:
To specify a different port (other than the default 8080), you can pass the new port number as an argument to the scripts (for example, `licenseServer.bat 8082`). You can also change
5. On the first run, you are prompted to set two sets of credentials:
   a. **Administrator credentials** - Used for accessing the Oxygen XML Editor license server administrative interface.
   b. **Standard user credentials** - Used by an Oxygen XML Editor application to connect to the license server.

**Tip:**

If you want to manually install, start, stop, or uninstall the server as a Windows service, run the following scripts from a command line as an Administrator:

- `installWindowsService.bat [serviceName]` - Installs the server as a Windows service with the name `serviceName`. The parameters for the license key folder and the server port can be set in the `oXygenLicenseServer.vmoptions` file.
- `startWindowsService.bat [serviceName]` - Starts the Windows service.
- `stopWindowsService.bat [serviceName]` - Stops the Windows service.
- `uninstallWindowsService.bat [serviceName]` - Uninstalls the Windows service.

If you do not provide the `serviceName` argument, the default name `oXygenLicenseServer` is used.

If the license server is installed as a Windows service, the output and error messages are automatically redirected to the following log files that are created in the install folder:

- `outLicenseServer.log` - Standard output stream of the server.
- `errLicenseServer.log` - Standard error stream of the server.

---

**Installing the HTTP License Server WAR Distribution**

1. Make sure that you have Java Servlet Container installed on the server you have selected to be the license server. Apache Tomcat 5.5 or higher is recommended (available at [http://tomcat.apache.org](http://tomcat.apache.org)).

**Important:**

By default, the license server stores the statistics database and other data in the Java Servlet Container's temporary directory. If you are not using Apache Tomcat, this directory may be deleted when the server is stopped or restarted. However, you can set the `oxygen.license.server.work.dir` system property to specify a different path for the directory where the database is stored.

2. Download the HTTP license server Web Archive (.war) from the [HTTP License Server website](http://http-license-server.Oxygencorp.com).
3. Configure three user roles in your installation of the Java Servlet Container (such as Apache Tomcat):
   a. One user with the role user, used by an Oxygen XML Editor application to connect to the license server. In the subsequent example, this user name is John.
   b. Another user with the role admin, used for accessing the HTTP License Server administrative interface and the management interface. In the subsequent example, this user name is Mary.

For example, in Apache Tomcat, a typical way to achieve this is to edit the tomcat-users.xml file from your Tomcat installation (if using a Tomcat zip/tar.gz distribution, by default this configuration file is found in the /TomcatInstallFolder/conf/ directory). After adding the three users, the configuration file might look like this:

```xml
<tomcat-users xmlns="http://tomcat.apache.org/xml"
               xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
               xsi:schemaLocation="http://tomcat.apache.org/xml tomcat-users.xsd"
               version="1.0">
  <!-- ... other user and role definitions ... -->
  <role rolename="user"/>
  <role rolename="admin"/>
  <user username="John" password="user_pass" roles="user"/>
  <user username="Mary" password="admin_pass" roles="admin"/>
</tomcat-users>
```

4. Deploy the WAR file.

   For example, in Apache Tomcat, go to the Web Application Manager page and log in with the user you configured with the admin role (Mary in the example above). In the WAR file to deploy section, choose the WAR file and click the Deploy button. The oXygenLicenseServlet application is now up and running, but the license key is not yet registered.

5. Go to the HTTP License Server administration page. By default, the address of this page is http://<server-address>/oXygenLicenseServlet. In Apache Tomcat, you can also open this page by clicking the oXygenLicenseServlet link in the manager page.

   You need to authenticate with the user configured with the admin role (Mary in the example above).

6. Activate the license key. This process involves binding your license key to your license server deployment. The browser used in the activation process needs to have Internet access.

   ![Note:
   If you cannot access the internet during the deployment, you can manually activate the license key (on page 111).](image)

Once the process is completed you cannot activate the license on another license server. Follow these steps to activate the license:
a. Paste your license key into the form and click **Register/Activate**.

**Step Result:** You will be redirected to an online form hosted on the Oxygen XML Editor website. This form is pre-filled with an activation code that uniquely identifies your license server deployment, and your license key.

b. Click **Register/Activate**.

If the activation process is successfully completed, your license server is running. Follow the on-screen instructions to configure the Oxygen XML Editor client applications.

7. The application's log file location is specified by the `log4j.appender.R2.File` property from the `WEB-INF/lib/log4j.properties` configuration file.

For example, in Apache Tomcat, the configuration file is located at: `TomcatInstallDir/webapps/oXygenLicenseServlet/WEB-INF/lib/log4j.properties` and the default log file location is `TomcatInstallDir/logs/oxygenLicenseServlet.log`.

### Installing Multiple Instances of the Oxygen License Servlet on a Tomcat Web Server

For cases where your organization has multiple sets of licenses (for example, an integrator with multiple clients might host a different license server for each client), follow this procedure to install multiple instances of the Oxygen License Servlet on a Tomcat web server:

1. Rename the license server WAR file according to your needs. For example, you could use the customer name and a number (e.g. `client23415`).
2. Go to your Tomcat license server manager (e.g. [http://my.tomcatserver.com:port/manager/](http://my.tomcatserver.com:port/manager/)) and enter your credentials.
3. Scroll to **WAR file to deploy** and press **Browse** button.
4. Locate the WAR file from step 1 and press the **Open** button.
5. Press the **Deploy** button.
6. Check that the newly deployed license server is running (it must be in the **Applications** table).

### Manual License Activation Procedure (For Users Whose License Server is Offline)

If your license server is not connected to the internet (therefore, the license cannot be activated automatically during the installation), you can manually activate the license by following these steps:

1. Access the HTTP license server management page in a web browser.
2. Copy the machine signature code.

---

**Note:**

The machine signature is displayed on the page as long as the license key was never activated. If you are trying to update/replace an already activated license key, the machine signature can be found by clicking on **Remove/Replace License**, then **Replace** on the next page.

4. Enter or paste the machine signature code and the license key, then click **Activate**.

   **Step Result:** The activated license key is displayed on-screen.

5. Copy the activated license key and paste it in the license registration page of the HTTP server.

**Preconfiguring License Server Details When Installing Oxygen XML Editor**

It is possible to install Oxygen XML Editor with the license server details preconfigured. For more information, see:

- **Windows:** Windows Installation: Command-Line Parameters for Preconfiguring License Server Details *(on page 88)*
- **Linux:** Linux Installation: Command-Line Parameters for Preconfiguring License Server Details *(on page 95)*

**Backup License Server Information**

If you want to use a backup license server, the setup instructions are the same as the procedures for a main license server, but it requires its own separate license key. Contact the Oxygen support team to find out more details about the backup license pricing and availability.

**Related information**

[Troubleshooting: Server Signature Mismatch Errors *(on page 117)*]

**License Server Management and Statistics Pages**

A system administrator can manage and access information about the license server at: http://hostName:port/oXygenLicenseServlet.

This page provides access to several statistics reports and management tasks. It also shows the current status of the server and provides additional instructions for using the license server with Oxygen XML Editor.

This page includes the following links for accessing statistics or managing tasks:

- **Current Allocated Licenses** - Opens the Allocated License Report page *(on page 113)*.
- **Usage Statistics** (Available only for floating licenses) - Opens the License Usage Statistics page *(on page 113)*.
- **View License Key** - Use this link to open a page where you can see details about the license key.
- **Replace/Remove License Key** - Use this link if you need to replace or remove the current license key *(on page 115)*.
- **Configuration** - Opens a page where you can configure notification settings and specify whether or not users are allowed to lock licenses. This page can be used for setting up the mail server used for sending emails whenever license requests from users are rejected.
• **Users management** (Available only for named-user licenses) - Opens a page where you can manage the list of users who are entitled to use the license key.

• **Allowed users list** (Available only for named-user licenses) - Opens a page where you can see the allowed users list (if one has already been configured), along with instructions for configuring one.

### Allocated License Report Page

This report page provides a system administrator the ability to revoke or unlock current running instances of licenses and includes the following information:

- **License load** - A graphical indicator that shows how many licenses are available.
- **License server status** - General information about the license server status, such as start time, license counts, rejected and acknowledged requests, average usage time, license refresh and timeout intervals, location of the license key, and the server version.
- **Current running instances** - Lists all currently acknowledged users, including user name, date and time when the license was granted, IP and MAC address of the computer where Oxygen XML Editor runs, and lock status.
  - **Revoke** - A system administrator can click on the *Revoke* icon next to a user name to release that particular license and return it to the pool.
  - **Unlock** - If a user has locked their license, the system administrator can also unlock it from this page.

---

**Note:**

This report is also available in XML format at: http://hostName:port/oXygenLicenseServlet/license-servlet/report-xml.

### License Usage Statistics Page (Floating License Only)

This report page provides some usage statistics for the floating licenses. It is helpful for determining the number of licenses that are needed and monitoring times when licenses are consumed. It includes the following information:

- **Maximum number of concurrent licenses** - Shows the maximum number of floating licenses that can be consumed at any given time.

- **Concurrent license consumption per day** - A chart that shows the peak number of licenses that were consumed and the total number of users that were rejected, on a daily basis. This chart can be used to detect the amount of concurrent licenses that are needed to avoid having rejected users.

**Tip:**

You can click on any bar to see the license consumption per hour for that particular day.
• **Concurrent license consumption per hour** - A chart that shows the peak number of licenses that were consumed per hour throughout that particular month. This is useful for identifying the time of day when the most licenses were consumed.

**Users Management Page (Named-User License Only)**

When using a named-user license key, the license server allocates available licenses on a *first come first served* basis until the maximum number is reached. Further users attempting to get a license key will be rejected.

This page provides access to the list of registered users and allows the server admin to:

- Revoke a user's right to use a license.
- Reactivate a previously deactivated user.
Replacing or Removing a License Key in an HTTP License Server

The following procedure assumes that your Oxygen XML Editor HTTP license server contains a previously activated license key (on page 110) and provides instructions for replacing it with another one or removing it completely.

This is useful if, for instance, you want to upgrade your existing license to the latest version or if you receive a new license key that accommodates a different number of users (on page 107).

Replacing a License Key

To replace a license key that is activated on your HTTP license server with a new one, follow these steps:

1. Access the license server by following the link provided by the Tomcat Web Application Manager page and log in using your admin credentials.
2. Click the Replace/Remove license key link. This will open a page that contains details about the license currently in use.
3. Click the Replace button.
4. Paste the new license key in the displayed form.
5. Click Register/Activate. The browser used in the process needs to have Internet access.
**Step Result:** You will be redirected to an online form hosted on the Oxygen XML Editor website. This form is pre-filled with an activation code that uniquely identifies your license server deployment and your license key.

**Note:**
If you cannot access the online activation form, you can manually activate the license key (on page 111).

**Result:** If the activation process is completed successfully, your license server is now running using the new license key. You can click View license key to inspect the key currently used by the license server.

### Removing a License Key

To remove a license key that is activated on your HTTP license server, follow these steps:

1. Access the license server by following the link provided by the Tomcat Web Application Manager page and log in using your admin credentials.
2. Click the Replace/Remove license key link. This will open a page that contains details about the license currently in use.
3. Click the Remove button to begin the license deletion procedure.
4. Click the Remove button in the confirmation page.

**Important:**
The removal process is irreversible. Once the process is complete, you cannot restore the license key.

### Upgrading Your HTTP License Server

The goal of the following procedure is to help you minimize the downtime when you upgrade the HTTP License Server to its latest version:

1. Access the license server by following the link provided by the Tomcat Web Application Manager page. If prompted for authentication, use the admin credentials.
2. Click the View license key link and copy the displayed license key to a file for later use.
3. Go to the Tomcat Web Application Manager page, log in with the user you configured with the admin role, and Undeploy the license server.
4. Download the Web Archive (WAR) distribution of HTTP license server.
5. Deploy the downloaded license server.
6. Access the license server by following the link provided by the Tomcat Web Application Manager page. If prompted for authentication, use the credentials configured for the admin user.
7. Paste your license key into the form and register it.
Configuring a License Server to Only Allow Certain Users

A system administrator can configure the license server to only allow specific users to request a license. This is available only for named-user licenses (not floating licenses) and is managed using an Allowed Users List.

To configure an Allowed Users List:

1. Create a text file named **allowed-users.txt**.
2. Enter the user name for each allowed user on a separate line.
3. Save the file in the license server work directory. You can go to the license server management page, and under Management Tasks, click the Allowed users list link to open a page where you can see the exact directory where it needs to be stored.

**Note:**

If the **allowed-users.txt** file is present but it is empty, all users are allowed to request a license.

In the license server management page, there is a link to the Allowed users list under Management Tasks. Also, in the Current Allocated Licenses page, there is a Show allowed users button. Both of them direct you to a page where you can see the Allowed Users List (if one has already been configured), along with instructions for configuring one. The list updates automatically between license requests every 30 seconds if the file is changed.

**Common Problems: License Server Errors**

This section includes some common problems that may appear when setting up a floating license server.

**Server Signature Mismatch Error**

**Problem**

I receive an error indicating that the current license was already activated on a License Server or that the License Server's Signature does not match.

During the license activation process, the license key becomes bound to a particular license server deployment. This means that a code that uniquely identifies your license server deployment (called Server Signature) is sent to the Oxygen XML Editor servers, which in turn will sign the license key. The Server Signature is computed from the list of network interfaces of the server where you deployed the license.

When starting the license server, if you receive an error stating that your Server Signature does not match, there are several possible causes:

**Possible Cause 1**

The license key was moved to a new server that hosts your license server.
Solution

Revert to your previous configuration.

Possible Cause 2

A new network interface was changed, added, or activated in the server that hosts your license server.

Note:
A specific example of when this could happen is if the Bluetooth or the WiFi module is activated/deactivated.

Solution

If reverting is not possible, contact the Oxygen support team.

Possible Cause 3

The license server was restarted from a different location as the previous restart. For example, some server configurations will have the Apache Tomcat server installed in a versioned folder (/usr/local/apache-tomcat-V.V.V) with a symbolic link to the typical folder (/usr/local/tomcat). The server can be restarted from either location, but it is recommended to always restart from the typical folder (/usr/local/tomcat) and always restart from the same location.

Solution

The server simply needs to always be restarted from the same location.

Upgrading

From time to time, upgrades and patch versions of Oxygen XML Editor are released to provide enhancements that fix problems and add new features.

By default, Oxygen XML Editor automatically checks for new versions at startup. If a newer version is detected, a dialog box will automatically be displayed that provides information about the type of upgrade or update that is available. If the newer version includes a critical bug fix, this dialog box will continue to be displayed every time Oxygen XML Editor is launched. To disable this check, open the Preferences dialog box (Options > Preferences) (on page 124), go to Global, and deselect Automatic Version Checking.

To check for new versions manually, go to Help > Check for New Versions. This opens a dialog box that displays information about whether or not a newer version is available.
Upgrading Oxygen XML Editor on Windows/Linux

What is Preserved During an Upgrade?

When you install a new version of Oxygen XML Editor, some data is preserved and some is overwritten. If there is a previous version of Oxygen XML Editor already installed on your computer, it can coexist with the new one, which means you do not have to uninstall it.

If you install over a previously installed version:

- All the files from its install directory will be removed, including any modification in framework (on page 3219) files, XSLT stylesheets, XML Catalogs (on page 3225), and templates.
- All global user preferences are preserved in the new version.
- All project preferences will be preserved in their project files.
- Any custom frameworks (on page 3219) that were stored outside the installation directory (as configured in Document type associations > Locations (on page 139)) will be preserved and will be found by the new installation.

If you install in a new directory:

- All the files from the old install directory will be preserved, including any modification in framework (on page 3219) files, XSLT stylesheets, XML Catalogs (on page 3225), and templates. However, these modifications will not be automatically imported into the new installation.
- All global user preferences are preserved in the new version.
- All project preferences will be preserved in their project files.
- Any custom frameworks (on page 3219) that were stored outside the installation directory (as configured in Document type associations > Locations (on page 139)) will be preserved and will be found by the new installation.

How to Upgrade Oxygen XML Editor on Windows or Linux

1. Upgrading to a new version might require a new license key. To check if your license key is compatible with the new version, select Help > Check for New Version. Note that the application needs an Internet connection to check the license compatibility.
2. Download and install the new version according to the instructions for your platform and the type of installer you selected.
3. If you installed from an archive (as opposed to an executable installer) you may have to update any shortcuts you have created or modify the system PATH to point to the new installation folder.
4. Restart Oxygen XML Editor.
5. If you require a new license for your upgrade, install it now according to the procedure for your platform and the type of installer you selected.
Upgrading Oxygen XML Editor on macOS

What is Preserved During an Upgrade?

When you install a new version of Oxygen XML Editor, first you need to remove or rename the old installation directory. By renaming the directory, it can coexist with the new installation and the following data will be preserved:

- All the files from the old install directory will be preserved, including any modification in framework (on page 3219) files, XSLT stylesheets, XML Catalogs (on page 3225), and templates. However, these modifications will not be automatically imported into the new installation.
- All global user preferences are preserved in the new version.
- All project preferences will be preserved in their project files.
- Any custom frameworks (on page 3219) that were stored outside the installation directory (as configured in Document type associations > Locations (on page 139)) will be preserved and will be found by the new installation.

How to Upgrade Oxygen XML Editor on macOS

1. Uninstall the current version of Oxygen XML Editor (on page 122) or rename the installation directory (for example, Oxygen XML Editor.old).
2. Upgrading to a new version might require a new license key. To check if your license key is compatible with the new version, select Help > Check for New Version. Note that the application needs an Internet connection to check the license compatibility.
3. Download and install the new version in an empty folder according to the instructions for your platform and the type of installer you selected.
4. If you installed from an archive (as opposed to an executable installer) you may have to update any shortcuts you have created or modify the system PATH to point to the new installation folder.
5. Restart Oxygen XML Editor.
6. If you require a new license for your upgrade, install it now according to the procedure for your platform and the type of installer you selected.

Installing and Updating Add-ons

Oxygen XML Editor provides an add-on (on page 3222) mechanism that can automatically discover and install frameworks (on page 2289) and plugins (on page 2445) from a remote location.

Note:
Frameworks that you install through the add-ons system are read-only.

Installing Add-ons

To install an add-on that is hosted on a remote update site, follow these steps:
1. Go to Help > Install new add-ons.
2. In the displayed dialog box, enter or paste the update site that hosts the add-on in the Show add-ons from field (or select it from the drop-down menu, if applicable). The default add-ons are hosted on https://www.oxygenxml.com/InstData/Addons/default/updateSite.xml. If you want to see a list of all the default and sample add-ons that are available on the Oxygen remote update sites, choose **ALL AVAILABLE SITES** from the drop-down menu. The add-ons list contains the name, status, update version, Oxygen XML Editor version, and the type of the add-on (either framework, or plugin). A short description of each add-on is presented under the add-ons list.

```
Note:
To see all the versions of the add-ons, deselect Show only compatible add-ons and Show only the latest version of the add-ons. Incompatible add-ons are shown only to acknowledge their presence on the remote update site, but you cannot install an incompatible add-on.
```

3. Choose the add-ons you want to install, click the **Next** button, then follow the on-screen instructions.

```
Note:
Accepting the license agreement of the add-on is a mandatory step in the installation process.
```

```
Note:
All add-ons are installed in the extensions directory inside the Oxygen XML Editor preferences directory (on page 125).
```

```
Tip:
As an alternate approach, you can add an **Install** button to a web page that links to a URL that has the syntax https://host/path/to/updateSite.xml?oxygenAddonId=addOnIDValue and drop the button into the application's main editing area.
```

**Managing Installed Add-ons**

To manage the installed add-ons, follow these steps:

1. Go to Help > Manage add-ons
2. The displayed dialog box presents a list of your installed add-ons along with various information (such as the installed version, the compatible Oxygen version, and more). The **Status** column will indicate if an update is available for a particular add-on. Also, you can click on the row for any particular add-on that has an update available to see details for the update (displayed in the preview pane below the list of add-ons).
3. To update an add-on, select the checkbox for the specific add-on, then click **Update** to update it (or **Uninstall** to remove it). If there is a newer version of the add-on available, Oxygen XML Editor will
download the package and install it. Follow the on-screen instructions to complete the installation process.

Note:
Accepting the license agreement of the add-on is a mandatory step in the installation process.

Checking for Add-on Updates

To check if there are available updates for the installed add-ons, go to Help > Check for add-ons updates. This action displays updates that are compatible with the current Oxygen XML Editor version.

Preserving Installed Add-ons After Upgrading Oxygen to a New Version

After installing or upgrading to a new version of Oxygen XML Editor, when you re-start the application, a dialog box will be displayed where you can decide which previously installed versions of add-ons should be imported and used for the new version of the product.

After you make the decision to import previous add-ons to the new application version, add-ons installed for previous versions remain present in the application settings folder and can still be used with those older application versions. The application may advise you to cleanup your previously installed add-on by displaying a dialog box where you can decide which previously installed versions of add-ons should be preserved or removed. To remove all add-ons for a particular application version (to free up disk space), select it in the main pane of the dialog box and click the Remove add-ons button.

Related information

Packing and Deploying Plugins as Add-ons (on page 2445)

Uninstalling

How to Uninstall Oxygen XML Editor

CAUTION:
The following procedure will remove Oxygen XML Editor from your system. All data stored in the installation directory will be removed, including any customizations or any other data you have stored within that directory. Make a backup of any data you want to keep before proceeding.

1. Back up any data you want to keep from the Oxygen XML Editor installation folder.
2. Remove the application according to your operating system:
   ◦ Windows or Linux - Use the appropriate uninstaller shortcut provided with your OS.
   ◦ macOS - Manually delete the installation folder and all its contents.
3. If you want to remove the user preferences:
• **Windows** - Remove the directory: `%APPDATA%\Roaming\com.oxygenxml` (usually `%APPDATA%` has the value: `[user-home-dir]\Application Data`). Note that this directory is hidden.
• **macOS** - Remove the directory: `Library/Preferences/com.oxygenxml` of the user home folder.
• **On Linux**, remove the directory: `.com.oxygenxml` from the user home directory.

**Unattended Uninstall**

The unattended uninstall procedure is available only on Windows and Linux.

Run the uninstaller executable from a command line with the `-q` parameter.

• **Windows** - The uninstaller executable is called `uninstall.exe` and is located in the `Oxygen` installation directory.
• **Linux** - The uninstaller executable is called `uninstall` and is located in the `Oxygen` installation directory.
4.

Configuring Oxygen XML Editor

This chapter presents all the user preferences and options that allow you to configure various features and aspects of the application itself. It also includes information about storing and sharing options, importing and exporting options or scenarios, customizing system properties, setting startup parameters, and the editor variables (on page 323) that are available for customizing user-defined commands.

Preferences

You can configure Oxygen XML Editor options using the Preferences dialog box.

To open the preferences dialog box, go to Options > Preferences.

You can select the preference page you are interested in from the tree on the left of the Preferences dialog box. You can filter the tree by using the filter text box and the following buttons are available to the right of the text box:

- Expand All - Expands the structure of the tree to show all preference pages.
- Collapse All - Collapses the structure of the tree to show only the 1st level preference pages.
- Project-Level Options Only - If toggled on, it filters the tree to only show the preference pages that are saved at project level (on page 312).
Click the icon or press F1 for help on any preferences page.

Some preference pages include an option to control how the options are stored, either as Global Options (on page 312) or Project Options (on page 312).

You can restore options to their default values by pressing the Restore Defaults button, available in each preferences page.

Preferences Directory Location

A variety of resources (such as global options, license information, and history files) are stored in a preferences directory (com.oxygenxml) that is in the following locations:
Global Preferences

The global options cover various aspects of the overall operation of Oxygen XML Editor. To configure the global options, open the Preferences dialog box (Options > Preferences) (on page 124) and go to Global.

The following options are available in the Global preferences page:

**Automatic Version Checking**

If this option is selected, Oxygen XML Editor will check for a new version on startup.

**Check for Oxygen-related events at startup**

If this option is selected, Oxygen XML Editor will check for various new event updates on the Oxygen XML Editor website and if any new events are found, they will be presented at startup.

**Check for notifications**

If selected (default value), the application will check for various types of messages from the Oxygen XML Editor website and they will be displayed in the status bar. The types of messages include the addition of new videos on the website, the announcement of upcoming webinars and conferences where the Oxygen XML Editor team will participate, and more.

**Language**

This option specifies the language used in the user interface. You can choose between English, French, German, Dutch, Japanese, or Chinese. You must restart Oxygen XML Editor for the change to take effect.

**Other language**

This option sets the language used in the user interface using an interface localization file. For details about creating this file, see Localizing the User Interface (on page 338). You can use this option to set the language of the user interface to a language that is not shipped with Oxygen XML Editor.

> **Note:**

If some interface labels are not rendered correctly after restarting the application, (for example, Korean characters are not displayed correctly), make sure that your operating system has the appropriate language pack installed (for example, the East-Asian language pack).

**Line separator**

This option specifies the type of line separator to be used when saving files. Use System Default to select the normal line separator for your OS. The other two possible selections are Unix-like and Windows-like.
Notes:

• This option is ignored if the Detect the line separator on file open option (on page 127) is selected AND a line separator is automatically detected.
• When changing the selection in this option, the change does not affect an opened file until you make a modification to the file and save it. At that point, all line separators in the file will change to the type of line separator you chose in this option.

Detect the line separator on file open

When this option is selected, the editor detects the line separator when a file is loaded and it uses it when the file is saved. If this option is not selected, you can use the Line separator option (on page 126) to choose the type of line separator to be used when saving files.

Tip:
To see the line separator type for the current file, you can use the Properties view (Window > Show View > Properties).

Default Internet browser

This option sets the Web browser that Oxygen XML Editor will use to do the following:

• Open (X)HTML or PDF transformation results.
• Open a web page.

If you leave this setting blank, the system default browser will be used.

Open last edited files from project

When this option is selected, Oxygen XML Editor opens the files you had open the last time you used a project whenever you open the application or switch to that project.

Load file content only when switching to its corresponding editor tab

When selected (default), files that were left open in the previous editing session remain as placeholder tabs but the file content is loaded only when switching to the corresponding editor tab. This helps to improve performance. If the option is deselected, the previously open files are all re-loaded at startup.

Check opened files for file system changes

When this option is selected, Oxygen XML Editor checks the content of the all open editors to see if they have been updated by another application. If the file has changed, Oxygen XML Editor will ask you if you want to reload the file.

Auto update unmodified editors on file system changes
If this option is selected, Oxygen XML Editor automatically updates unmodified editors if the edited file changes externally.

**Beep on operation finished**

When this option is selected, Oxygen XML Editor beeps when a validation or transform action ends. Different tones are used for success and failure. The tones used may depend on the sound settings in your operating system.

**Show memory status**

When this option is selected, the memory that Oxygen XML Editor uses is displayed in the status bar. To free memory, click the **Free unused memory** button located at the right side of the status bar. The memory status bar turns yellow or red when Oxygen XML Editor uses too much memory. You can change the amount of memory available to Oxygen XML Editor by changing the parameters of the application launcher (on page 339).

**Order of switching between editor tabs**

This option specifies the order for switching between open file tabs when using **Ctrl + Tab** (Command + Tab on macOS) or **Ctrl + Shift + Tab** (Command + Shift + Tab on macOS) (on page 393). You can choose between:

- **Recently used order** - Switches to the most recently used tab.
- **Visual order** - Switches to the next tab in visual order.

**File Chooser Dialog section**

**Use platform file chooser (Windows and macOS)**

This option is selected by default and it specifies that the native file chooser is used. You can deselect this option if you want the Java Swing file chooser to be used instead. If Oxygen XML Editor encounters a problem while using the native file manager, it will avoid using it again in the current session, even if this option is selected.

**Consider application bundles to be directories when browsing (macOS only)**

This option is available only on the macOS platform. When selected, the file browser dialog box allows you to browse inside an application bundle, as in a regular folder. Otherwise, it is not allowed (the same as the Finder application on macOS).

**Show hidden files and directories**

If this option is selected, Oxygen XML Editor shows system hidden files and folders in the file browser dialog box and the folder browser dialog box.
Tip:
On macOS, you need to press `Command + Shift + Period` in the file browser to show hidden files.

File chooser opens
This option specifies the starting directory that the file browser dialog box (on page 381) will open. You can choose between:

- **Directory of the selected file** - The file browser opens the folder where the selected file is stored, depending on the current selection (for example, a file could be selected from the Project view, DITA Maps Manager, main editing pane, or another location within the application).
- **Last visited directory** - The file browser opens the last visited folder.

Appearence Preferences
This preferences page contains various options that allow you to change the appearance of the user interface of Oxygen XML Editor. To configure the Appearance options, open the Preferences dialog box (Options > Preferences) (on page 124) and go to Appearance.

The following options are available in the Appearance preferences page:

**Look and Feel**
This option allows you to change the graphic style (look and feel) of the user interface. Depending on the operating system, you can choose between various predefined style options.

**Theme**
This option allows you to choose predefined color themes that will be applied over the entire user interface. You can select between the following:

- **Light** (default theme in Windows)
- **Classic** (default theme in macOS)

Note:
In Windows, if a high contrast theme is detected and the Theme option is set to Classic and the Look and Feel option (on page 129) is set to Default or Windows, Oxygen XML Editor inherits the high contrast theme colors that are set in the operating system.

- **Graphite**
You can also change various appearance-related options in other preference pages for the selected theme by clicking on the various links in this section.
**Custom Themes**

You can also create custom themes to share with others or use in other installations of Oxygen XML Editor. To create a custom theme, follow these steps:

1. Select a **Theme** to use as a base.
2. Configure the desired options in any of the option pages listed in this preferences page.
3. Click **Export** and specify a name for your custom theme. If you save the theme to the default file path, your custom theme will immediately appear in the **Theme** drop-down list. Otherwise, if you save it to another location, you can use the **Import button** *(on page 130)* to make it appear in the drop-down list.

**Note:**

In macOS (starting with Yosemite), if you choose **Graphite** for the **Theme**, it is recommended that you select the **Use dark menu and Dock** option that is found in **System Preferences > General**.

**Theme preview area**

Displays a preview of the current **Theme selection** *(on page 129)* (available for predefined color themes).

**Theme management section**

**Reset**

Resets the theme to its default values (this option is available when the theme is modified).

**Rename**

Changes the name of the theme (not available for default or predefined themes).

**Delete**

Removes the selected theme (not available for default or predefined themes).

**Import**

Allows you to import a color theme from an XML theme file. You can use this option to load an exported **custom theme**(on page 130).

**Export**

Allows you to export the current color theme into an XML theme file that can then be shared with others or imported into another installation of Oxygen XML Editor.

**Configure icon saturation and brightness link**
This link is available if you are using the Graphite theme (on page 129). It opens a dialog box where you can configure the saturation and brightness for all the icons in Oxygen XML Editor.

![Figure 17. Configure Icon Saturation and Brightness Dialog Box](image)

**Colors Preferences**

Oxygen XML Editor allows you to configure the colors for frames, dialog boxes, controls, and commands. To configure the Colors, open the Preferences dialog box (Options > Preferences) (on page 124) and go to Appearance > Colors.

Clicking the color button for any of the options opens a **Choose color** dialog box. It includes several tabs that allow you to configure the color in numerous ways. This page allows you to select and configure the color for the following:

**Background Colors**

- **Background**
  - Background color for various general user interface items.

- **Components background**
  - Background color for various components (such as text fields, views, tables, and dialog boxes).

- **Components selection background**
  - Background color for the current selections in certain components, such as some views and panes.

- **Components inactive selection background**
  - Background color for a selection in a view that is not the current focus.

**Menus, toolbars and frame background**
Background color for specific components such as menus, toolbars, and the application frame.

**Menus and toolbars selection background**

(This option is not available for macOS) Background color for menu selections and toolbar buttons.

**View titles background**

Background color for the titles of view and tabs.

**Status bar background**

Background color of the status bar at the bottom of the editor.

**Foreground Colors**

**Foreground**

Foreground color for various general user interface items.

**Component selection foreground**

Foreground color for the current selection.

**Disabled foreground**

Foreground color for various components that are not the current focus (such as views other than the currently selected one).

**Link foreground**

Foreground color for links in views and dialog boxes.

**View titles foreground**

Foreground color for the title bar of views.

**Status bar foreground**

Foreground color for the text in the status bar at the bottom of the editor.

**Other Colors**

**Borders and table grids**

Color for certain borders and table grid lines.

**Text component border**

Color for the borders of text fields and drop-down lists.

**View/Editor tabs border**

Color for the borders of views and tabs.

**Scroll bars, chevrons**

Color for scroll bars (navigation bars) and chevrons (button to expand a non-visible area).
Separator

Color for the separators in toolbars, menus, and dialog boxes.

Note:
You must restart the application for your changes to be applied.

Fonts Preferences

Oxygen XML Editor allows you to choose the fonts to be used in the Text, Design, and Grid editor modes, and fonts for the Author mode that are not specified in the associated CSS stylesheet. To configure the font options, open the Preferences dialog box (Options > Preferences) (on page 124) and go to Appearance > Fonts.

The following options are available:

Editor

Specifies the font family, size, and weight to be used in the Text mode editor. To change the current values, double-click the text field or click the Choose button. This opens a dialog box where you can choose the font family, font size, and whether or not to bold the text. You can enter or paste content in the Sample box to see a preview of how it will look in the application. If you select the Show only the fonts that can render the sample text option and paste content in the Sample box, the application detects fonts that can render the particular character set and filters the fonts that can be selected accordingly.

Note:
On macOS, the default font, Monaco, cannot be rendered in bold.

Author default font

Specifies the default font family, size, and weight to be used in Author mode. However, the default font will be overridden by the fonts specified in any CSS file associated with the open document. To change the current values, double-click the text field or click the Choose button. This opens a dialog box where you can choose the font family, font size, and whether or not to bold the text. You can enter or paste content in the Sample box to see a preview of how it will look in the application. If you select the Show only the fonts that can render the sample text option and paste content in the Sample box, the application detects fonts that can render the particular character set and filters the fonts that can be selected accordingly.

Schema default font

This option allows you to choose the font to be used in:

• The Design mode of the XML Schema editor (on page 947).
• Images with schema diagram fragments that are included in the HTML documentation generated from an XML Schema.
To change the current values, double-click the text field or click the **Choose** button. This opens a dialog box where you can choose the font family, font size, and whether or not to bold the text. You can enter or paste content in the **Sample** box to see a preview of how it will look in the application. If you select the **Show only the fonts that can render the sample text** option and paste content in the **Sample** box, the application detects fonts that can render the particular character set and filters the fonts that can be selected accordingly.

**Text antialiasing**

This option allows you to set the text anti-aliasing behavior:

- **Default** - Allows the application to use the setting of the operating system, if available.
- **On** - Sets the text anti-aliasing to pixel level.
- **Off** - Disables text anti-aliasing.
- Sub-pixel anti-aliasing modes, such as GASP, LCD_HRGB, LCD_HBGR, LCD_VRGB, and LCD_VBGR.

**Text components**

Specifies the font family, size, and weight to be used in text boxes within the interface. To change the current values, double-click the text field or click the **Choose** button. This opens a dialog box where you can choose the font family, font size, and whether or not to bold the text. You can enter or paste content in the **Sample** box to see a preview of how it will look in the application. If you select the **Show only the fonts that can render the sample text** option and paste content in the **Sample** box, the application detects fonts that can render the particular character set and filters the fonts that can be selected accordingly.

**GUI**

Specifies the font family, size, and weight to be used for user interface labels. To change the current values, double-click the text field or click the **Choose** button. This opens a dialog box where you can choose the font family, font size, and whether or not to bold the text. You can enter or paste content in the **Sample** box to see a preview of how it will look in the application. If you select the **Show only the fonts that can render the sample text** option and paste content in the **Sample** box, the application detects fonts that can render the particular character set and filters the fonts that can be selected accordingly.

**View titles font**

Specifies the font family, size, and weight to be used in the titles of the various views within the interface. To change the current values, double-click the text field or click the **Choose** button. This opens a dialog box where you can choose the font family, font size, and whether or not to bold the text. You can enter or paste content in the **Sample** box to see a preview of how it will look in the application. If you select the **Show only the fonts that can render the sample text** option and paste content in the **Sample** box, the application detects fonts that can render the particular character set and filters the fonts that can be selected accordingly.
Related information
Changing the Font Size in the Editor (on page 521)

Application Layout Preferences

Oxygen XML Editor offers various perspectives (on page 3221) and views that you can arrange in a variety of layouts to suit your needs.

To configure the application layout options, open the Preferences dialog box (Options > Preferences) (on page 124) and go to Application Layout. The following options are available:

Select application layout

You can choose between the following three layouts:

Default

Uses the default layout for all perspectives (on page 3221). Any modification of this layout (such as closing views, displaying views, or a new view arrangement) is saved on exit and reloaded at start-up.

Predefined

Allows you to choose one of the predefined layouts:

- **Advanced** - All views are displayed.
- **Author** - An authoring-oriented layout that includes the following views:
  - Project (on page 402), Archive Browser (on page 2008), DITA Maps Manager (on page 2897), Outline (on page 539), Attributes (on page 627), Model (on page 545), and Elements (on page 633).
- **Basic** - Only the Project view (on page 402) and Outline view (on page 539) are visible. Recommended when you edit XML content and you need maximum screen space.
- **Schema development** - The Project (on page 402), Component Dependencies (on page 1000), Resource Hierarchy/Dependencies (on page 997), Outline (on page 993), Palette (on page 949), and Attributes (on page 995) views are displayed.
- **XQuery development** - The Project (on page 402), Outline (on page 1037), XSLT/XQuery Input (on page 904), XPath/XQuery Builder (on page 1039), and Transformation Scenarios (on page 1555) views are displayed.
- **XSLT development** - The Project (on page 402), Component Dependencies (on page 909), Resource Hierarchy/Dependencies (on page 906), Outline (on page 899), Attributes (on page 542), Model (on page 545),
Custom

Allows you to specify a custom layout to be used. You can save your preferred layout using Window > Export Layout, then enter the location of the saved layout file in this setting.

Reset layout at startup

When this option is selected, Oxygen XML Editor forgets any changes made to the layout during a session and reloads the default layout the next time it is started. This is useful when you want to keep a fixed layout from one session to another.

Remember layout changes for each project

When this option is selected, Oxygen XML Editor saves layouts individually for each project. When you switch projects, the layout you last used for that project is loaded automatically.

Allow detaching of editors from main window

When this option is selected, you can drag and drop an editor window outside of the main screen. This is useful especially when you are using two monitors and you want to view files side by side.

Note:

If the main screen is maximized, you cannot drag and drop an editor outside of it.

View tab placement

Specifies whether the View tabs are located at the top or bottom of the window.

Editor tab placement

Specifies whether the Editor tabs are located at the top or bottom of the window.

The changes you make to any layout are preserved between working sessions. The predefined layout files are saved in the preferences directory of Oxygen XML Editor.

For more information about configuring the user interface of Oxygen XML Editor, watch our video demonstration:

https://www.youtube.com/embed/anwjepfAdEk

Add-ons Preferences

You can use add-ons to enhance the functionality of Oxygen XML Editor. To configure the Add-ons options, open the Preferences dialog box (Options > Preferences) and go to Add-ons.
The following options are available in this preferences page:

**Enable automatic updates checking**

When this option is selected, Oxygen XML Editor will automatically search for available updates.

**Add-on Sites URLs**

This is a list of the URLs for the add-on sites. You can add, edit, and delete sites in this list by using the buttons below the list.

**Automatically install add-ons**

You can use this section to specify required add-ons for a project. Then, when a user opens the project, the specified add-ons will be automatically installed after prompting the user. You can use Ctrl+Space to open a pop-up window with the list of detected add-ons that you can select to be marked for automatic installation. You can also manually enter the add-on ID and multiple IDs can be entered by separating with either a space or new line.

**Project Level Settings Preferences**

The **Project Level Settings** preference page allows you to decide whether various settings should be saved in the project configuration file or in the global settings. Settings that are saved at project level can easily be shared with others. To configure these options, open the Preferences dialog box (Options > Preferences) (on page 124) and go to **Project Level Settings**.

The following options can be toggled on or off to determine which settings will be saved at project level:

**Allow validation scenario associations to be saved at project level**

When this option is selected, the associations for custom validation scenarios (on page 788) will be stored according to their storage location. If you associate a scenario that is stored at the project level, the association will be saved at project level (in the project configuration file). If you associate a scenario at global level, it will be saved globally. If this option is not selected, the association is not allowed to be saved at project level and will be saved globally (even if the scenario is saved in the project file).

**Allow transformation scenario associations to be saved at project level**

When this option is selected, the associations for custom transformation scenarios (on page 1464) will be stored according to their storage location. If you associate a scenario that is stored at the project level, the association will be saved at project level (in the project configuration file). If you associate a scenario at global level, it will be saved globally. If this option is not selected, the association is not allowed to be saved at project level and will be saved globally (even if the scenario is saved in the project file).

**Save current DITA root map at project level**

When this option is selected, when you change the currently selected DITA context root map (on page 2901), it will be saved in the project configuration file.

**Save DITA media working sets at project level**
When this option is selected, all configured working sets for DITA media resources (on page 3068) will be saved in the project configuration file.

**Save DITA map validate and check for completeness settings at project level**

When this is selected, the options chosen in the DITA Map Validate and Check for Completeness dialog box (on page 2941) will be saved in the project configuration file.

### Document Type Association Preferences

Oxygen XML Editor uses document type associations (on page 3218) to associate a document type (on page 1291) with a set of functionality provided by a framework (on page 3219). To configure the Document Type Association options, open the Preferences dialog box (Options > Preferences) (on page 124) and go to Document Type Association.

The following actions are available in this preferences page:

- **Discover more frameworks by using add-ons update sites**
  
  Click on this link to specify URLs for framework add-on update sites.

- **Document Type Table**
  
  This table presents the currently defined frameworks (on page 3219) (document type associations (on page 3218)), sorted by priority and alphabetically. Each edited document type has a set of association rules (on page 142) (used by the application to detect the proper document type association to use for an open XML document).

- **New**
  
  Opens a Document type configuration dialog box (on page 140) that allows you to add a new framework.

- **Edit**
  
  Opens a Document type configuration dialog box (on page 140) that allows you to edit an existing framework.

  **Note:**
  
  If you try to edit an existing framework when you do not have write permissions to its storage location, a dialog box will be shown asking if you want to extend it.

- **Duplicate**
  
  Opens a Document type configuration dialog box (on page 140) that allows you to duplicate the configuration of an existing framework. This will create a snapshot of the framework in its current form. It is merely a copy of the document type and will not evolve along with the base document type as the Extend action does.

- **Extend**
Opens a Document type configuration dialog box (on page 140) that allows you to extend an existing framework. You can add or remove functionality starting from a base document type. All of these changes will be saved as a patch. When the base document type is modified and evolves (for example, from one application version to another) the extension will evolve along with the base document type, allowing it to use the new actions added in the base document type.

Delete

Deletes the selected framework (document type).

Enable DTD/XML Schema processing in document type detection

When this option is selected (default value), the matching process also examines the DTD/XML Schema associated with the document. For example, the fixed attributes declared in the DTD for the root element are also analyzed, if this is specified in the association rules. This is especially useful if you are writing DITA customizations. DITA topics and maps are also matched by looking for the @DITAArchVersion attribute of the root element. This attribute is specified as default in the DTD and it is detected in the root element, helping Oxygen XML Editor to correctly match the DITA customization.

Only for local DTDs/XML Schemas

When this option is selected (default value), only the local DTDs / XML Schemas will be processed.

Enable DTD/XML Schema caching

When this option is selected (default value), the associated DTDs or XML Schema are cached when parsed for the first time, improving performance when opening new documents with similar schema associations.

Related information

Sharing a Framework (on page 2288)

Locations Preferences

Oxygen XML Editor allows you to change the location where frameworks (on page 3219) (document types) are stored, and to specify additional framework directories. The Locations preferences page allows you to specify the main frameworks folder location. You can choose between the Default directory (\{OXYGEN_INSTALL_DIR\}/frameworks) or a Custom specified directory. You can also change the current frameworks folder location value using the com.oxygenxml.editor.frameworks.url system property set in either the .vmoptions configuration files (on page 339) or in the startup scripts (on page 342).

A list of additional frameworks directories can also be specified. The application will look in each of those folders for additional document type configurations to load. Use the Add, Edit and Delete buttons to manage the list of folders.

A document type configuration (framework) can be loaded from the following locations:
• **Internal preferences** - The document type configuration is stored in the application [Internal preferences](#).

• **Additional framework directories** - The document type configuration is loaded from one of the specified [Additional frameworks directories](#) list.

• **Add-ons** - An [add-on](#) can contribute a framework. You can manage the add-ons locations in the [Add-ons preferences page](#).

• **The frameworks folder** - The main folder containing framework configurations.

All loaded document type configurations are first sorted by priority, then by document type name and then by load location (in the exact order specified above). When an XML document is opened, the application chooses the first document type configuration from the sorted list that matches the specific document.

All loaded document type configurations are first sorted by priority, then by document type.

**Document Type Configuration Dialog Box**

The **Document Type Configuration** dialog box allows you to create or edit a framework ([document type](#)). It is displayed when you use the **New**, **Edit**, **Duplicate**, or **Extend** buttons in the **Document Type Association preferences page** ([on page 138](#)) (open the **Preferences** dialog box (**Options > Preferences**) ([on page 124](#)) and go to **Document Type Association**).

**Figure 18. Document Type Configuration Dialog Box**

![Document Type Configuration Dialog Box](#)

The configuration dialog box includes the following fields and sections:

**Name**

The name of the framework. This will be displayed as its name in the **Document Type** column in the **Document Type Association** preferences page ([on page 138](#)).

**Priority**
Depending on the priority level, Oxygen XML Editor establishes the order that the existing frameworks are evaluated to determine the type of a document you are opening. It can be one of the following: Lowest, Low, Normal, High, or Highest. You can set a higher priority for frameworks you want to be evaluated first.

**Note:**
The built-in document types are set to Low priority by default. Frameworks that have the same priority are sorted alphabetically.

**Description**
The document type description displayed as a tooltip in the Document Type Association preferences page (on page 138).

**Storage**
The location where the framework is saved. If you select the External storage option, the framework is saved in a specified file with a mandatory extension (located in a subdirectory of your current framework directory). If you select the Internal storage option, the framework configuration data is saved in the Oxygen XML Editor internal options file (if Global Options (on page 312) is selected) or in the current Oxygen XML Editor project xpr file (if Project Options (on page 312) is selected).

**Initial edit mode**
Sets the default edit mode when you open a document for the first time: Editor specific, Text, Author, Grid and Design (available only for the W3C XML Schema editor). If the Editor specific option is selected, the initial editing mode is determined based upon the document type. You can find the mapping between editors and edit modes in the Edit modes preferences page. (on page 171) You can impose an initial mode for opening files that match the association rules of the document type. For example, if the files are usually edited in the Author mode, you can set it in the Initial edit mode combo box.

**Note:**
You can also customize the initial mode for a document type in the Edit modes preferences page. Open the Preferences dialog box (Options > Preferences) (on page 124) and go to Editor > Edit modes.

**Configuration Tabs**
The bottom section of the dialog box includes various tabs where you can configure numerous options for the framework.
Association Rules Tab

To open the Association Rules tab of the Document type configuration dialog box, open the Preferences dialog box (Options > Preferences) (on page 124), go to Document Type Association, use the New, Edit, Duplicate, or Extend button (on page 138), and click on the Association Rules tab.

In the Association rules tab, you can perform the following actions:

- **New**
  Opens the Document type rule dialog box allowing you to create association rules.

- **Edit**
  Opens the Document type rule dialog box allowing you to edit the properties of the currently selected association rule.

- **Delete**
  Deletes the currently selected association rules from the list.

- **Move Up**
  Moves the selected association rule up one spot in the list.

- **Move Down**
  Moves the selected association rule down one spot in the list.

By combining multiple association rules you can instruct Oxygen XML Editor to identify the type of a document. Oxygen XML Editor identifies the type of a document when the document matches at least one of the association rules. This tab gives you access to a Document type rule dialog box that you can use to create association rules that activate on any document matching all the criteria defined in the dialog box.

To create a new association rule, click the + New button at the bottom of the Association Rules tab, or to edit an existing rule, click the ⌁ Edit button.
The **Document type rule** dialog box includes the following fields and options:

**Namespace**

Specifies the namespace of the root element from the association rules set ("any" by default). If you want to apply the rule only when the root element has no namespace, leave this field empty (remove the **ANY_VALUE** string).

**Root local name**

Specifies the local name of the root element ("any" by default).

**File name**

Specifies the name of the file ("any" by default).

**Public ID**

Represents the Public ID of the matched document.

**Attribute Local name**

Specifies the local name of the attributes for the root element ("any" by default).

**Attribute Namespace**

Specifies the namespace of the attributes for the root element ("any" by default).

**Attribute Value**

Specifies the value of the attributes for the root element ("any" by default).

**Java class**

Presents the name of the Java class that is used to determine if a document matches the rule. This Java class should implement the `ro.sync.ecss.extensions.api.DocumentTypeCustomRuleMatcher` interface.
Tip: You can use wildcards (? and *) or editor variables (on page 323) in the Document Type Rule dialog box, and you can enter multiple values by separating them with a comma.

Schema Tab

In the Schema tab, you can specify a default schema for Oxygen XML Editor to use if an XML document does not contain a schema declaration and no default validation scenario is associated with it.

To open the Schema tab of the Document type configuration dialog box, open the Preferences dialog box (Options > Preferences) (on page 124), go to Document Type Association, use the New, Edit, Duplicate, or Extend button (on page 138), and click on the Schema tab.

This tab includes the following options for defining a schema to be used if no schema is detected in the XML file:

- **Schema type**
  
  Use this drop-down list to select the type of schema.

- **Schema URI**
  
  You can specify the URI of the schema file. You can specify the path by using the text field, its history drop-down, the Insert Editor Variables (on page 323) button, or the browsing actions in the Browse drop-down list.

Tip: It is a good practice to store all resources in the framework directory and use the ${framework} editor variable (on page 330) to reference them. This is a recommended approach to designing a self-contained document type that can be easily maintained and shared between multiple users.

Classpath Tab

The Classpath tab displays a list of folders and JAR (on page 3220) libraries that hold implementations for API extensions, implementations for custom Author mode operations, various resources (such as stylesheets), and framework (on page 3219) translation files. Oxygen XML Editor loads the resources looking in the folders in the order they appear in the list.

To open the Classpath tab of the Document type configuration dialog box, open the Preferences dialog box (Options > Preferences) (on page 124), go to Document Type Association, use the New, Edit, Duplicate, or Extend button (on page 138), and click on the Classpath tab.

The Classpath tab includes the following actions:

- New
Opens a dialog box that allows you to add a resource to the table in the **Classpath** tab. You can specify the path by using the text field, its history drop-down, the **Insert Editor Variables** button, or the browsing actions in the **Browse** drop-down list.

**Tip:**
The path can also contain wildcards (for example, `${framework}/lib/*.jar`).

**Edit**
Opens a dialog box that allows you to edit a resource in the **Classpath** tab. You can specify the path by using the text field, its history drop-down, the **Insert Editor Variables** button, or the browsing actions in the **Browse** drop-down list.

**Tip:**
The path can also contain wildcards (for example, `${framework}/lib/*.jar`).

**Delete**
Deletes the currently selected resource from the list.

**Move Up**
Moves the selected resource up one spot in the list.

**Move Down**
Moves the selected resource down one spot in the list.

**Use parent classloader from plugin with ID (on page 2444)**
Use this option to specify the ID of a **plugin (on page 3222)**. The current **framework** has access to the classes loaded for the **plugin**.

**Author Tab**
The **Author** tab is a container that holds information regarding the CSS file used to render a document in the **Author** mode, and regarding **framework (on page 3219)**-specific actions, menus, contextual menus, toolbars, and content completion list of proposals.

To open the **Author** tab of the **Document type** configuration dialog box, open the **Preferences** dialog box (Options > Preferences) (on page 124), go to **Document Type Association**, use the New, Edit, Duplicate, or Extend button (on page 138), and click on the **Author** tab.
The options that you configure in the **Author** tab are grouped in subtabs.

### CSS Subtab

The **CSS** subtab contains the CSS files that Oxygen XML Editor uses to render a document in the **Author** mode. In this subtab, you can set **main** and **alternate** CSS files. When you are editing a document in the **Author** mode, you can switch between these CSS files from the **Styles** drop-down menu on the **Author Styles** toolbar.

To open the **CSS** subtab, open the **Preferences** dialog box (**Options > Preferences**) *(on page 124)*, go to **Document Type Association**, use the **New, Edit, Duplicate, or Extend** button *(on page 138)*, click on the **Author** tab, and then the **CSS** subtab.

The following actions are available in the **CSS** subtab:

- **New**

  Opens a dialog box that allows you to add a CSS file. You can specify the path by using the text field, its history drop-down, the ✂️ **Insert Editor Variables** *(on page 323)* button, or the browsing actions in the ⌃ ⌘钥匙 **Browse** drop-down list.

- **Edit**

  Opens a dialog box that allows you to edit the current selection.

- **Delete**

  Deletes the currently selected CSS file.

- **Move Up**

  Moves the selected CSS file up in the list.

- **Move Down**

  Moves the selected CSS file down in the list.

**Enable multiple selection of alternate CSSs**

Allows users to apply multiple alternate styles, as layers, over the main CSS style. This option is selected by default for DITA document types.

**If there are CSSs specified in the document then**

You can choose between the following options for controlling how the CSS files that are set in this subtab will be handled if a CSS is specified in the document itself:

- **Ignore CSSs from the associated document type** - The CSS files set in this CSS subtab are overwritten by the CSS files specified in the document itself.

- **Merge them with CSSs from the associated document type** - The CSS files set in this CSS subtab are merged with the CSS files specified in the document itself.
Actions Subtab

The Actions subtab of the Document Type Configuration dialog box contains a sortable table with all the Author mode actions that are configured for the specific framework (on page 3219). Each action has a unique ID, a name, a description, and a shortcut key.

To open the Actions subtab, open the Preferences dialog box (Options > Preferences) (on page 124), go to Document Type Association, select your framework, use the Duplicate or Extend button to create an extension of the framework (or the Edit button for an already extended framework), click on the Author tab, and then the Actions subtab.

The following features are available in this subtab:

**Export existing actions (↑)***

It is possible to export existing actions to use them in other frameworks. Each exported action is extracted from the framework configuration file and exported as an individual XML file.

To export actions, the Storage option (on page 141) in the top part of the Document Type Configuration dialog box must be set to External and the external location must be a subdirectory of your current framework directory.

The Export action is found by right-clicking an action or a selection of multiple actions (the Export button is also located below the table of actions). If you choose to export a single action, a resulting dialog box will allow you to select the destination path for the new XML file that contains the configuration details of the action. If you export multiple actions, they will automatically be saved as individual XML files inside a newly created folder (it will have _externalAuthorActions at the end of the folder name) inside your current framework directory.

**Result:** Exported actions will display the icon in the first column in the table.

**Important:**

The newly created files for the exported actions will not appear on disk until you click OK several times to confirm your changes and exit the Preferences dialog box.

**Tip:**

If you want to create a new XML file for an action, there is a document template called Author Actions in the New document wizard (on page 368) to help you get started.
Note:
You can add or edit the action files outside of Oxygen XML Editor, but you will need to restart the application each time to reload the changes.

For more information, see Creating or Editing Actions Using an Individual XML File for Each Action (on page 2149).

Open in editor (↵)
For exported actions, there is a Open in editor action in the contextual menu that will open the XML file for that action in the main editor.

Create a new action (+)
Use the New button (located underneath the table of actions) to open the Action dialog box (on page 148) where you can configure a new action.

Duplicate an existing action (⋯)
Use the Duplicate action (found in the contextual menu and underneath the table of actions) to duplicate the selected action.

Edit an existing action (↵)
Use the Edit button (found in the contextual menu and underneath the table of actions) to open the Action dialog box (on page 148) where you can edit the selected action.

Delete an existing action (×)
Use the Delete button (found in the contextual menu and underneath the table of actions) to delete the selected action.

Author Action Dialog Box
To edit an existing document type action or create a new one, open the Preferences dialog box (Options > Preferences) (on page 124), go to Document Type Association, use the New, Edit, Duplicate, or Extend button (on page 138), click on the Author tab, and then the Actions subtab. At the bottom of this subtab, click New to create a new action, or Edit to modify an existing one.
The following options are available in the **Action** dialog box:

**ID**

Specifies a unique action identifier.

**Name**

Specifies the name of the action. This name is displayed as a tooltip or as a menu item.

**Tip:**

You can use the $\text{SI18n(key)}$ editor variable (on page 331) to allow for multiple translations of the name.

**Menu access key**

In Windows, you can access menus by holding down `Alt` and pressing the keyboard key that corresponds to the `letter` that is underlined in the name of the menu. Then, while still holding down `Alt`, you can select submenus and menu action the same way by pressing subsequent corresponding keys. You can use this option to specify the `letter` in the name of the action that can be used to access the action.

**Description**

A description of the action. This description is displayed as a tooltip when hovering over the action.
Tip:
You can use the \( \text{i18n('key')} \) editor variable (on page 331) to allow for multiple translations of the description.

How to translate frameworks link

Use this link to see more information about Localizing Frameworks (on page 2229).

Large icon

Allows you to select an image for the icon that Oxygen XML Editor uses for the toolbar action.

Tip:
A good practice is to store the image files inside the framework directory and use the \( \text{frameworks} \) editor variable (on page 330) to make the image relative to the framework location. If the images are bundled in a jar archive (for instance, along with some Java operations implementation), it is convenient to reference the images by their relative path location in the class-path.

Small icon

Allows you to select an image for the icon that Oxygen XML Editor uses for the contextual menu action.

Note:
If you are using a Retina or HiDPI display, Oxygen XML Editor automatically searches for higher resolution icons in the path specified in both the Large icon and Small icon options. For more information, see Using Retina/HiDPI Icons for the Actions from a Framework (on page 2181).

Shortcut key

This field allows you to configure a shortcut key for the action that you are editing. The + character separates the keys.

Enable platform-independent shortcut keys

If this checkbox is selected, the shortcut that you specify in this field is platform-independent and the following modifiers are used:

- **M1** represents the Command key on macOS, and the Ctrl key on other platforms.
- **M2** represents the Shift key.
- **M3** represents the Option key on macOS, and the Alt key on other platforms.
- **M4** represents the Ctrl key on macOS, and is undefined on other platforms.
Operations section

In this section of the Action dialog box, you configure the functionality of the action that you are editing. An action has one or more operation modes. The evaluation of an XPath expression activates an operation mode. The first selected operation mode is activated when you trigger the action. The scope of the XPath expression must consist only of element nodes and attribute nodes of the edited document. Otherwise, the XPath expression does not return a match and does not fire the action. For more details see: Controlling Which Author Operations Gets Executed Through XPath Expressions (on page 152).

The following options are available in this section:

**Activation XPath**

An XPath 2.0 expression that applies to elements and attributes. For more details see: Controlling Which Author Operations Gets Executed Through XPath Expressions (on page 152).

**Operation**

Specifies the invoked operation that can be a default operation (on page 2152) or a custom operation (on page 2177).

**Arguments**

Specifies the arguments of the invoked operation. The 📀 Edit at the bottom of the table allows you to edit the arguments of the operation.

**Operation priority**

Increases or decreases the priority of an operation. The operations are invoked in the order of their priority. If multiple XPath expressions are true, the operation with the highest priority is invoked.

- ✤ Add - Adds an operation.
- ✗ Remove - Removes an operation.
- 📀 Duplicate - Duplicates an operation.

**Evaluate activation XPath expressions even in read-only contexts**

If this checkbox is selected, the action can be invoked even when the cursor is placed in a read-only location.

Related information

Localizing Frameworks (on page 2229)
Controlling Which Author Operations Gets Executed Through XPath Expressions

An Author mode action can have multiple operation modes, each one invoking an Author operation (on page 2152) with certain configured parameters. Each operation mode has an XPath 2.0 expression for activating it.

For each operation mode of an action, the application will check if the XPath expression is fulfilled (when it returns a non-empty node set or a true result). Only the first operation whose XPath operation is fulfilled will be executed.

The following special XPath extension functions are provided:

- **oxy:allows-child-element()** (on page 152) - Use this function to check whether or not an element is valid child element in the current context, according to the associated schema.
- **oxy:allows-global-element()** (on page 154) - Use this function to check whether or not an element is a valid global element for the current framework (on page 3219), according to the associated schema.
- **oxy:current-selected-element()** (on page 155) - Use this function to get the currently selected element.
- **oxy:selected-elements()** (on page 156) - Use this function to get the selected elements.
- **oxy:is-required-element()** (on page 156) - Use this function to check if the element returned by the given XPath expression is required (based on the rules declared in the schema).
- **oxy:platform()** (on page 156) - Use this function to get the current platform in cases where you want to enable or disable an action depending on the platform. Possible values include: eclipse, standalone and webapp.

**oxy:allows-child-element() Function**

The **oxy:allows-child-element()** function allows you to check whether or not an element that matches the arguments of the function is valid as a child of the element at the current cursor position, according to the associated schema. It is evaluated at the cursor position and has the following signature:

```
oxy:allows-child-element($childName, ($attributeName, $defaultAttributeValue, $contains?)?)
```

The following parameters are supported:

- **childName**
  
  The name of the element that you want to check if it is valid in the current context. Its value is a string that supports the following forms:

  - The child element with the specified local name that belongs to the default namespace.
    ```
    oxy:allows-child-element("para")
    ```
    
    The above example verifies if the `<para>` element (of the default namespace) is allowed in the current context.

  - The child element with the local name specified by any namespace.
The above example verifies if the `<para>` element (of any namespace) is allowed in the current context.

• A prefix-qualified name of an element.

```xml
oxy:allows-child-element("prefix:para")
```

The prefix is resolved in the context of the element where the cursor is located. The function matches on the element with the `para` local name from the previously resolved namespace. If the prefix is not resolved to a namespace, the function returns a value of `false`.

• A specified namespace-URI-qualified name of an element.

```xml
oxy:allows-child-element("{namespaceURI}para")
```

The `namespaceURI` is the namespace of the element. The above example verifies if the `<para>` element (of the specified namespace) is allowed in the current context.

• Any element.

```xml
oxy:allows-child-element("*")
```

The above function verifies if any element is allowed in the current context.

**Note:**
A common use case of `oxy:allows-child-element("*")` is in combination with the `attributeName` parameter.

### attributeName
The attribute of an element that you want to check if it is valid in the current context. Its value is a string that supports the following forms:

• The attribute with the specified name from no namespace.

```xml
oxy:allows-child-element("*", "class", " topic/topic ")
```

The above example verifies if an element with the `class` attribute and the default value of this attribute (that contains the `topic/topic` string) is allowed in the current context.

• The attribute with the local name specified by any namespace.

```xml
oxy:allows-child-element("*", "*:localname", " topic/topic ")
```

• A qualified name of an attribute.
oxy:allows-child-element("*", "prefix:localname", " topic/topic ")

The prefix is resolved in the context of the element where the cursor is located. If the prefix is not resolved to a namespace, the function returns a value of false.

defaultAttributeValue

A string that represents the default value of the attribute. Depending on the value of the next parameter, the default value of the attribute must either contain this value or be equal to it.

contains

An optional boolean. The default value is true. For the true value, the default value of the attribute must contain the defaultAttributeValue parameter. If the value is false, the two values must be the same.

oxy:allows-global-element() Function

The oxy:allows-global-element() function allows you to check whether or not an element that matches the arguments of the function is valid for the current framework (on page 3219), according to the associated schema. It has the following signature:

oxy:allows-global-element($elementName, ($attributeName, $defaultAttributeValue, $contains?)?)

The following parameters are supported:

elementName

The name of the element that you want to check if it is valid in the current framework. Its value is a string that supports the following forms:

• The element with the specified local name that belongs to the default namespace.

oxy:allows-global-element("para")

The above example verifies if the <para> element (of the default namespace) is allowed in the current framework.

• The element with the local name specified by any namespace.

oxy:allows-global-element("*:para")

The above example verifies if the <para> element (of any namespace) is allowed in the current framework.

• A prefix-qualified name of an element.

oxy:allows-global-element("prefix:para")

The prefix is resolved in the context of the framework. The function matches on the element with the para local name from the previously resolved namespace. If the prefix is not resolved to a namespace, the function returns a value of false.
• A specified namespace-URI-qualified name of an element.

```xml
oxy:allows-global-element("{namespaceURI}para")
```

The `namespaceURI` is the namespace of the element. The above example verifies if the `<para>` element (of the specified namespace) is allowed in the current `framework`.

• Any element.

```xml
oxy:allows-global-element("*")
```

The above function verifies if any element is allowed in the current `framework`.

**attributeName**

The attribute of an element that you want to check if it is valid in the current `framework`. Its value is a string that supports the following forms:

• The attribute with the specified name from no namespace.

```xml
oxy:allows-global-element("*", "class", " topic/topic ")
```

The above example verifies if an element with the `class` attribute and the default value of this attribute (that contains the `topic/topic` string) is allowed in the current `framework`.

• The attribute with the local name specified by any namespace.

```xml
oxy:allows-global-element("*", "*:localname", " topic/topic ")
```

• A qualified name of an attribute.

```xml
oxy:allows-global-element("*", "prefix:localname", " topic/topic ")
```

The prefix is resolved in the context of the `framework`. If the prefix is not resolved to a namespace, the function returns a value of `false`.

**defaultAttributeValue**

A string that represents the default value of the attribute. Depending on the value of the next parameter, the default value of the attribute must either contain this value or be equal to it.

**contains**

An optional boolean. The default value is `true`. For the `true` value, the default value of the attribute must contain the `defaultAttributeValue` parameter. If the value is `false`, the two values must be the same.

**oxy:current-selected-element() Function**

This function returns the fully selected element. If no element is selected, the function returns an empty sequence.

*Example: oxy:current-selected-element Function*
oxy:current-selected-element()[self::p]/b

This example returns the &lt;b&gt; elements that are children of the currently selected &lt;p&gt; element.

oxy:selected-elements() Function

This function returns the selected elements from Author mode.

Example: oxy:selected-elements Function

oxy:selected-elements()[self::para][@audience="novice"]

This example would activate an action when at least one of the selected elements is a &lt;para&gt; element with the @audience attribute defined.

oxy:is-required-element() Function

This function checks if the element returned by the given XPath expression is required (based on the rules declared in the schema). It has only one argument, an XPath expression, and the XPath expression must be written in such a way that it returns a single element.

Example: oxy:is-required-element Function

oxy:is-required-element(.)

This example would check to see if the current element is required by the schema.

oxy:is-editable-element() Function

This function checks if the element returned by the given XPath expression is editable (content can be inserted in it), meaning both that the entire XML file is editable and that the current context where the element is placed is editable. For example, if the element is inside an &lt;xi:included&gt; section, it is not editable.

It only has one argument, an XPath expression, and the XPath expression must be written in such a way that it returns a single element.

Example: oxy:is-editable-element Function

oxy:is-editable-element(ancestor-or-self::table)

This example would return true if the cursor is placed inside a table and it is editable or false if it is not editable.

oxy:platform() Function

This function returns the current platform. You can use this if you want to enable or disable an action depending on the platform. The possible values are: standalone, eclipse, or webapp.

Example: oxy:platform Function

oxy:platform()="standalone"
This example would keep the action activated for the *standalone* distribution of Oxygen XML Editor, but disable it for the *Eclipse* and *Web Author* distributions.

**Menu Subtab**

In the **Menu** subtab, you can configure which actions will appear in the framework (on page 3219)-specific menu. The subtab is divided into two sections: *Available actions* and *Current actions*.

To open the **Menu** subtab, open the Preferences dialog box (Options > Preferences) (on page 124), go to Document Type Association, use the New, Edit, Duplicate, or Extend button (on page 138), click on the Author tab, and then the **Menu** subtab.

The **Available actions** section presents a table that displays the actions defined in the Actions subtab, along with their icon, ID, and name. The **Current actions** section holds the actions that are displayed in the Oxygen XML Editor menu. To add an action in this section as a sibling of the currently selected action, use the Add as sibling button. To add an image in this section as a child of the currently selected action, use the Add as child button.

The following actions are available in the **Current actions** section:

- **Edit**
  Edits an item.

- **Remove**
  Removes an item.

- **Move Up**
  Moves an item up.

- **Move Down**
  Moves an item down.

**Contextual Menu Subtab**

In the **Contextual menu** subtab you configure what framework (on page 3219)-specific action the Content Completion Assistant (on page 3217) proposes. The subtab is divided into two sections: *Available actions* and *Current actions*.

To open the **Contextual Menu** subtab, open the Preferences dialog box (Options > Preferences) (on page 124), go to Document Type Association, use the New, Edit, Duplicate, or Extend button (on page 138), click on the Author tab, and then the **Contextual Menu** subtab.
The **Available actions** section presents a table that displays the actions defined in the **Actions** subtab, along with their icon, ID, and name. The **Current actions** section contains the actions that are displayed in the contextual menu for documents that belong to the edited **framework**.

The following actions are available in this subtab:

- **Add as sibling**
  
  Adds the selected action or submenu from the **Available actions** section to the **Current actions** section as a sibling of the selected action.

- **Add as child**
  
  Adds the selected action or submenu from the **Available actions** section to the **Current actions** section as a child of the selected action.

- **Edit**
  
  This option is available for container (submenu) items that are listed in the **Current actions** section. It opens a configuration dialog box that allows you to edit the selected container (submenu).
The following options are available in this dialog box:

**Name**

Specifies the name of the action. This name is displayed as a tooltip or as a menu item.

Tip: You can use the `${i18n('key')}` editor variable (on page 331) to allow for multiple translations of the name.

**Menu access key**

In Windows, you can access menus by holding down **Alt** and pressing the keyboard key that corresponds to the *letter* that is underlined in the name of the menu. Then, while still holding down **Alt**, you can select submenus and menu action the same way by pressing subsequent corresponding keys. You can use this option to specify the *letter* in the name of the action that can be used to access the action.

**Menu icon**

Allows you to select an image for the icon that Oxygen XML Editor uses for the container (submenu).

**Promote items when in a table context**

If this option is selected, when invoking the contextual menu from within a table, all the actions in this container (submenu) will be promoted to the main level in the contextual menu. Actions and submenus that are not promoted are still available in the Other actions submenu when invoking the contextual menu within a table.

**Remove**

Removes the selected action or submenu from the **Current actions** section.

**Move Up**

Moves the selected item up in the list.

**Move Down**
Moves the selected item down in the list.

**Toolbar Subtab**

In the **Toolbar** subtab you configure what framework (on page 3219)-specific action the Oxygen XML Editor toolbar holds. The subtab is divided into two sections: **Available actions** and **Current actions**.

To open the **Toolbar** subtab, open the Preferences dialog box (Options > Preferences) (on page 124), go to Document Type Association, use the New, Edit, Duplicate, or Extend button (on page 138), click on the Author tab, and then the Toolbar subtab.

The **Available actions** section presents a table that displays the actions defined in the Actions subtab, along with their icon, ID, and name. The **Current actions** section holds the actions that are displayed in the Oxygen XML Editor toolbar when you work with a document belonging to the edited framework. To add an action in this section as a sibling of the currently selected action, use the Add as sibling button. To add an action in this section as a child of the currently selected action, use the Add as child button.

The following actions are available in the **Current actions** section:

- **Edit**
  - Edits an item.

- **Remove**
  - Removes an item.

- **Move Up**
  - Moves an item up.

- **Move Down**
  - Moves an item down.

**Content Completion Subtab**

In the **Content Completion** subtab you configure what framework (on page 3219)-specific the Content Completion Assistant (on page 3217) proposes. The subtab is divided into two sections: **Available actions** and **Current actions**.

To open the **Content Completion** subtab, open the Preferences dialog box (Options > Preferences) (on page 124), go to Document Type Association, use the New, Edit, Duplicate, or Extend button (on page 138), click on the Author tab, and then the Content Completion subtab.

**Available and Current Actions**

The **Available actions** section presents a table that displays the actions defined in the Actions subtab, along with their icon, ID, and name. The **Current actions** section holds the actions that the Content Completion Assistant proposes when you work with a document that belongs to the edited framework.
To add the selected available action as a sibling of the currently selected action in the Current actions section, use the Add as sibling button. To add it as a child of the currently selected action, use the Add as child button. To edit an existing action, select it and use the Edit button. To remove an existing action, use the Remove button. You can also move items up and down the list using the Move Up or Move Down buttons.

Adding an action (or editing an existing one) opens the Content Completion Item dialog box.

**Figure 23. Content Completion Item Dialog Box**

Use this dialog box to configure the action:

**Action**

Displays the name of the selected action.

**Display name**

You can use the drop-down menu to choose between displaying the action name or the replaced element name, or you can enter another name to be displayed.

**Replacement for**

Use this section to replace an existing element with the configured action:

- **Element name and Namespace** - The name (and namespace, if needed) of the replaced element. The original element no longer needs to be excluded using the Filter - Remove content completion items table (on page 162).
- **Display item only when element is allowed at cursor** - The configured action is contributed in the UI components selected in the Contribute to section only if the associated schema allows the original element at the current location in the document. This is equivalent to defining activation XPaths in the Author Action dialog box (on page 148) using the oxy:allows-child-element() XPath extension function. Activation XPaths for the action are still checked when the action is invoked.
Contribute to

Use this section to specify where to display the configured item in the interface:

- **Content Completion Window** - The configured item will appear in the *Content Completion Assistant* (on page 3217).
- **Elements View** - The configured item will appear in the *Elements view* (on page 546).
- **Element Insert Menus** - The configured item will appear in the *Append Child, Insert Before, or Insert After* menus that are available in certain contextual menus (for example, the contextual menu of the *Outline view* (on page 539)).

Filter Table

The *Filter* section presents a table that allows you to add elements to be filtered from the *Content Completion Assistant* or from some specific helper views or menus. Use the **Add** button to add more filters to the table, the **Edit** button to modify an existing item in the table, or the **Remove** button to remove a filtered item. The **Add** and **Edit** buttons open a *Remove item* dialog box.

**Figure 24. Remove Item Dialog Box**

Use this dialog box to add or configure the elements that will be filtered:

**Item name**

Use this text field to enter the name of the element to be filtered. The drop-down list also includes a few special content completion actions that can be filtered:

- **<SPLIT> [elementName]** - Filters split entries for elements that have the form *Split elementName* or *New elementName*.
- **<SPLIT>** - Filters split entries for all elements.
- **<ENTER>** - Filters *Insert New Line* entries that appear in elements where whitespace is significant.

The filter list can be used to remove only content completion items contributed by the schema associated to the document and it does not remove actions added to the content completion
list via the framework configuration. The element name specified in the filter is not namespace aware, it matches the name of the element defined in the associated schema exactly as it would appear rendered in the content completion window.

Note:
If you try to insert an element in an invalid position (for example, using the content completion assistant), the editor will attempt to make the insertion valid. This may mean finding an alternate position for the insertion or splitting the element at the current position. If a `<SPLIT>` entry is added in the filter list for an element, the editor will never split that element.

Remove item from
You can choose to filter the element from any of the following:

- **Content Completion Window** - The element will not appear in the Content Completion Assistant (on page 3217).
- **Elements View** - The element will not appear in the Elements view (on page 546).
- **Element Insert Menus** - The element will not appear in the Append Child, Insert Before, or Insert After menus that are available in certain contextual menus (for example, the contextual menu of the Outline view (on page 539)).
- **Entities View** - The element will not appear in the Entities view (on page 547).

Related information
Customizing the Content Completion Assistant Using a Configuration File (on page 2191)

Templates Tab

The **Templates** tab specifies a list of directories where new document templates are located for this particular framework. These directories, along with the document templates that are saved inside them, will appear in the New Document wizard (on page 368) inside the Framework templates category according to your framework and the directory path you specify in this tab.

To open the Templates tab of the Document type configuration dialog box, open the Preferences dialog box (Options > Preferences) (on page 124), go to Document Type Association, use the New, Edit, Duplicate, or Extend button (on page 138), and click on the Templates tab.

The Templates tab includes the following actions:

- **New**
  Opens a dialog box that allows you to specify the path to a directory that contains document templates for this framework. You can specify the path by using the text field, its history drop-down, the Insert Editor Variables (on page 323) button, or the browsing actions in the Browse drop-down list.
Edit

Opens a dialog box that allows you to edit the path of a directory that contains document templates for this framework. You can specify the path by using the text field, its history drop-down, the Insert Editor Variables (on page 323) button, or the browsing actions in the Browse drop-down list.

Tip:
The path can also contain wildcards. For example, using ${frameworkDir}/templates/* would add all the template folders found inside the templates directory.

Delete

Deletes the currently selected template directory from the list.

Move Up

Moves the selected template directory up one spot in the list.

Move Down

Moves the selected template directory down one spot in the list.

Related information
Creating New Document Templates (on page 375)
Customizing Document Templates (on page 377)
Sharing Custom Document Templates (on page 381)

Catalogs Tab

The Catalogs tab specifies a list of XML Catalogs (on page 826), specifically for the edited framework (on page 3219), that are added to list of catalogs that Oxygen XML Editor uses to resolve resources.

To open the Catalogs tab of the Document type configuration dialog box, open the Preferences dialog box (Options > Preferences) (on page 124), go to Document Type Association, use the New, Edit, Duplicate, or Extend button (on page 138), and click on the Catalogs tab.

You can perform the following actions:

Add
Opens a dialog box that allows you to add a catalog to the list. You can specify the path by using the text field, its history drop-down, the
Insert Editor Variables (on page 323) button, or the browsing actions in the Browse drop-down list.

Edit
Opens a dialog box that allows you to edit the path of an existing catalog.

Delete
Deletes the currently selected catalog from the list.

Move Up
Moves the selected catalog up one spot in the list.

Move Down
Moves the selected catalog down one spot in the list.

Transformation Tab
In the Transformation tab, you can configure the transformation scenarios associated with the particular framework (on page 3219) you are editing. These transformation scenarios are presented in the Configure Transformation Scenarios dialog box (on page 1548) when transforming a document and you can specify which scenarios will be used by default for a particular document type.

To open the Transformation tab of the Document type configuration dialog box, open the Preferences dialog box (Options > Preferences) (on page 124), go to Document Type Association, use the New, Edit, Duplicate, or Extend button (on page 138), and click on the Transformation tab.

The Transformation tab offers the following options:

Default checkbox
You can set one or more of the scenarios listed in this tab to be used as the default transformation scenario when another specific scenario is not specified. The scenarios that are set as default are rendered bold in the Configure Transformation Scenarios dialog box (on page 1548).

New
Opens the New scenario dialog box allowing you to create a new transformation scenario for the particular document type (on page 1464).

Duplicate
Allows you to duplicate the configuration of an existing transformation scenario. It opens the Edit scenario dialog box where you can configure the properties of the duplicated scenario (on page 1547).

Edit
Opens the **Edit scenario** dialog box allowing you to **edit the properties of the currently selected transformation scenario** *(on page 1545).*

- **Delete**
  - Deletes the currently selected transformation scenario.

- **Import scenarios**
  - Imports transformation scenarios.

- **Export selected scenarios**
  - Export transformation scenarios.

- **Move Up**
  - Moves the selection to the previous scenario.

- **Move Down**
  - Moves the selection to the next scenario.

**Validation Tab**

In the **Validation** tab, you can configure the validation scenarios associated with the particular **framework** *(on page 3219)* you are editing. These validation scenarios are presented in the **Configure Validation Scenarios** dialog box when validating a document and you can specify which scenarios will be used by default for a particular document type.

**Note:**

If a **main file** is associated with the current file, the validation scenarios defined in the **main file**, along with any Schematron schema defined in the default scenarios for that particular **framework**, are used for the validation. These take precedence over other types of validation units defined in the default scenarios for the particular **framework**. For more information on **main files**, see **Contextual Project Operations Using 'Main Files' Support** *(on page 418)* or **Modular Contextual XML Editing Using 'Main Files' Support** *(on page 829).*

To open the **Validation** tab of the **Document type** configuration dialog box, open the **Preferences** dialog box *(Options > Preferences)* *(on page 124)*, go to **Document Type Association**, use the **New, Edit, Duplicate, or Extend** button *(on page 138)*, and click on the **Validation** tab.

The **Validation** tab offers the following options:

- **Default checkbox**
  - You can set one or more of the scenarios listed in this tab to be used as the default validation scenario when another specific scenario is not specified in the validation process. The scenarios that are set as default are rendered bold in the **Configure Validation Scenarios** dialog box.

- **New**
Opens the **New scenario** dialog box allowing you to create a new validation scenario *(on page 788).*

**Duplicate**

Allows you to duplicate the configuration of an existing validation scenario. It opens the **Edit scenario** dialog box where you can configure the properties of the duplicated scenario *(on page 798).*

**Edit**

Opens the **Edit scenario** dialog box allowing you to edit the properties of the currently selected validation scenario *(on page 798).*

**Delete**

Deletes the currently selected validation scenario.

**Import scenarios**

Imports validation scenarios.

**Export selected scenarios**

Export validation scenarios.

**Move Up**

Moves the selected scenario up one spot in the list.

**Move Down**

Moves the selected scenario down one spot in the list.

### Extensions Tab

The **Extensions** tab specifies implementations of Java interfaces used to provide advanced functionality to the document type.

To open the **Extensions** tab of the **Document type** configuration dialog box, open the **Preferences** dialog box *(Options > Preferences)* *(on page 124)*, go to **Document Type Association**, use the **New, Edit, Duplicate, or Extend** button *(on page 138)*, and click on the **Extensions** tab.

Libraries containing the implementations must be present in the *classpath (on page 144)* of your document type. The Javadoc available at [https://www.oxygenxml.com/InstData/Editor/SDK/javadoc/](https://www.oxygenxml.com/InstData/Editor/SDK/javadoc/) contains details about how each API implementation functions.

### Document Templates Preferences

Oxygen XML Editor provides a variety of built-in document templates that make it easier to create new documents in various formats. The list of available templates is presented in the **New Document wizard (on page 368)** when you create a new document *(New toolbar button or File > New).*
You can also create your own templates (on page 375) and share them with others. You can store your custom document templates in the existing templates folder in the Oxygen XML Editor installation directory or store them in a custom directory. If you store them in a custom directory, you need to use this Document Templates preferences page to add that directory to the list of template directories that Oxygen XML Editor makes available in the New Document wizard.

To add a template directory, follow these steps:

1. open the Preferences dialog box (Options > Preferences) (on page 124) and go to Document Templates.
2. Use the New button to select a location of the new document template folder.
3. You can also use the Edit or Delete buttons to manage folders in the list, and you can alter the order that Oxygen XML Editor looks in these directories by using the Up and Down buttons.

Result: This will add the folder to the list in this preferences page and it will now appear in the New Document wizard (on page 368) in a category based upon the folder path you specified.

Note:
For DITA templates, they will also appear in the dialog box for creating new DITA topics from the DITA Maps Manager, but if you customize the template (on page 377), you need to set the type property to dita in the corresponding properties file.

Encoding Preferences

Oxygen XML Editor lets you configure how character encodings are recognized when opening files and which encodings are used when saving files. To configure encoding options, open the Preferences dialog box (Options > Preferences) (on page 124) and go to Encoding.

The following encoding options are available:

Fallback character encoding
Specifications the default character encoding of non-XML documents if their character encoding cannot be determined from other sources (for example, it is not specified in the document or determined by the file type).

Note:
For certain document types, the following encoding detection rules are used:

- For XML, DTD, and CSS documents, Oxygen XML Editor tries to collect the character encoding from the document. If no such encoding is found, then UTF-8 is used.
- For JavaScript, JSON, SQL, XQuery, and RNC, the UTF-8 encoding is used.
UTF-8 BOM handling

This setting specifies how to handle the Byte Order Mark (BOM) when Oxygen XML Editor saves a UTF-8 XML document:

- **Keep** (default) - Do not alter the BOM declaration of the currently open file.
- **Write** - Save the BOM bytes.
- **Don't Write** - Do not save the BOM bytes. Loaded BOM bytes are ignored.

**Note:**
The UTF-16 BOM is always preserved. UTF-32 documents have a big-endian byte order.

Encoding errors handling

This setting specifies how to handle characters that cannot be represented in the character encoding that is used when the document is opened. The available options are:

- **REPORT** (default) - Displays an error identifying the character that cannot be represented in the specified encoding. Unrecognized characters are rendered as an empty box.
- **REPLACE** - The character is replaced with a standard replacement character. For example, if the encoding is UTF-8, the replacement character has the Unicode code \x{FFFD}, and if the encoding is ASCII, the replacement character code is 63.
- **IGNORE** - The error is ignored and the character is not included in the document displayed in the editor.

**Attention:**
If you edit and save the document, the characters that cannot be represented in the specified encoding are dropped.

Encoding for Base64, Base32, Hex conversions

Specifies the encoding to be used when invoking the **Encode Selection** or **Decode Selection** actions for Base64 (on page 569), Base32 (on page 570), or Hex conversions (on page 571). The default setting is UTF8.

Encode non-ASCII characters in URL paths

If selected (default), Oxygen XML Editor will escape non-ASCII characters (encode them with their hexadecimal equivalent) within URL paths. If you are using a non-Latin alphabet (such as Arab, Japanese, Chinese), it may be beneficial to deselect this option so that non-ASCII characters in URL paths will not be escaped and will remain more readable.
Editor Preferences

Oxygen XML Editor offers the possibility to configure the appearance of various components and features of the main editor. To access these options, open the Preferences dialog box (Options > Preferences) (on page 124) and go to Editor.

The following options are available:

**Selection background color**
Allows you to set the background color of selected text.

**Selection foreground color**
Allows you to set the color of selected text.

**Completion proposal background**
Allows you to set the background color of the Content Completion Assistant (on page 3217).

**Completion proposal foreground**
Allows you to set the color of the text in the Content Completion Assistant (on page 3217).

**Documentation window background**
Allows you to set the background color of the documentation of elements suggested by the Content Completion Assistant (on page 3217).

**Documentation window foreground**
Allows you to set the color of the text for the documentation of elements suggested by the Content Completion Assistant (on page 3217).

**Find highlight color**
Allows you to set the color of the highlights generated by the Find and Find all actions.

**XPath highlight color**
Allows you to set the color of the highlights generated when you run an XPath expression.

**Declaration highlight color**
Allows you to set the color of the highlights generated by the Find declaration action.

**Reference highlight color**
Allows you to set the color of the highlights generated by the Find reference action.

**Maximum number of highlights**
Allows you to set the maximum number of highlights that Oxygen XML Editor displays.

**Show TAB/NBSP/EOL/EOF marks**
Makes the TAB/NBSP/EOL/EOF characters visible in the editor. You can use the color picker to choose the color of the marks.

**Show SPACE marks**
Can edit read-only files

If this option is selected, Oxygen XML Editor will let you edit read-only files. When you try to save them, a Save As dialog box will be displayed to avoid overwriting the initial resource. If the option is not selected, a warning message is displayed when you try to edit a read-only file.

Display quick-assist and quick-fix side hints

Displays the Quick Assist (on page 3223) icon ( ) and Quick Fix (on page 3223) icon ( ) in the line number stripe on the left side of the editor.

Undo history size

Allows you to set the maximum amount of undo operations you can perform in any of the editor modes (Text, Author, Design, Grid).

Enable mouse-wheel zooming

If selected, you can use Ctrl + MouseWheelForward (Command + MouseWheelForward on macOS) to increase the editor font (zoom in) or Ctrl + MouseWheelBackwards (Command + MouseWheelBackwards on macOS) to decrease the editor font (zoom out). It is enabled by default on Windows and Linux, while it is disabled by default on macOS, due to the way inertia affects the mouse wheel on this operating system.

Edit Modes Preferences

Oxygen XML Editor lets you configure which edit mode (on page 353) a file is opened in the first time it is opened. This setting only applies the first time a file is opened. The current editing mode of each file is saved when the file is closed and restored the next time it is opened. To configure the options for editing modes, open the Preferences dialog box (Options > Preferences) (on page 124) and go to Editor > Edit Modes.

Allow Document Type specific edit mode setting to override the general mode setting

If selected, the initial edit mode setting set in the Document Type configuration dialog box (on page 140) overrides the general edit mode setting from the table below.

Select the initial edit mode (page) for each editor

This table specifies the default editing mode that will be opened for each type of document when the Allow Document Type specific edit mode setting to override the general mode setting option is not selected. Use the Edit button to change the initial edit mode for each type of document (editor). The initial edit mode can be one of the following:

- **Text** (on page 353)
- **Author** (on page 354)
- **Grid** (on page 354)
- **Design** (available only for the XSD editor).
Text Preferences

Oxygen XML Editor allows you to configure how the Text mode editor (on page 353) appears. To configure these options, open the Preferences dialog box (Options > Preferences) (on page 124) and go to Editor > Edit modes > Text.

The following options are available:

**Editor background color**

Sets the background color for the Text editing mode, Outline view (on page 539), and some external tool editors (Large File Viewer (on page 2674), Compare Files (on page 474), Compare Directories (on page 494)).

**Editor cursor color**

Sets the color for the cursor in Text mode.

**Highlight current line**

If selected, the current line is highlighted with the foreground color specified with the color chooser.

**Show line numbers**

If selected (default value), line numbers are shown in the editor panels and in the Output view (on page 2103) of the debugger perspectives (on page 3221). You can also specify the color for the line numbers using the color chooser. Printed output will also include the line numbers.

**Show print margin**
If selected, it allows you to set a safe print limit in the form of a vertical line displayed in the right side of the editor pane. You can also customize the print margin line color.

**Print margin column**
Allows you to specify a limit for the print width, measured in the number of characters.

**Line wrap**
If selected, long lines are automatically wrapped in edited documents. The line wrap does not alter the document content since the application does not use *new-line* characters to break long lines.

**Cut / Copy whole line when nothing is selected**
If selected, *Cut* and *Copy* actions operate on the entire current line when nothing is selected in the editor.

**Enable folding**
If selected (default value), the vertical stripe that holds the *folding markers* (on page 528) is displayed in *Text* mode.

**Highlight matching tag**
If selected, when you place the cursor on a start or end tag, Oxygen XML Editor highlights the corresponding member of the pair. You can also customize the highlight color.

**Lock the XML tags**
If selected, XML are locked and cannot be edited in *Text* mode.

**Diagram Preferences**
For certain XML languages, Oxygen XML Editor provides a diagram view as part of the *Text* mode editor. To configure the *Diagram* preferences, open the *Preferences* dialog box (*Options > Preferences*) (on page 124) and go to *Editor > Edit modes > Text > Diagram*.

The following options are available in this preference page:

**Show Full Model XML Schema diagram**
When this option is selected, the *Text* mode editor for XML Schemas includes a split-screen view that shows a diagram of the schema structure. This is useful for seeing the effects of schema changes you make. For editing a schema using a diagram instead of text, use the *schema Design view* (on page 355).

**Note:**
When handling very large schemas, displaying the schema diagram might affect the performance of your system. In such cases, disabling the schema diagram view improves the speed of navigation through the edited schema.
Enable Relax NG diagram and related views

Enables the Relax NG schema diagram and synchronization with the related views (Attributes (on page 542), Model (on page 545), Elements (on page 546), Outline (on page 1091)).

Show Relax NG diagram

Displays the Relax NG schema diagram in the split-screen views (Full Model View (on page 1084) and Logical Model View (on page 1085)).

Enable NVDL diagram and related views

Enables the NVDL schema diagram and synchronization with the related views (Attributes (on page 542), Model (on page 545), Elements (on page 546), Outline (on page 1107)).

Show NVDL diagram

Displays the NVDL schema diagram in the split-screen views (Full Model View (on page 1103) and Logical Model View (on page 1104)).

Location relative to editor

Allows you to specify the location of the schema diagram panel relative to the diagram Text editor.

Show/Hide Annotations link

Use this link to navigate to the Schema Design preferences page (on page 198) where you can choose to show or hide annotations in schema diagrams.

Zoom link

Use this link to navigate to the Schema Design preferences page (on page 198) where you can adjust the default zoom level of schema diagrams.

Grid Preferences

Oxygen XML Editor provides a Grid view (on page 354) of an XML document. To configure the Grid options, open the Preferences dialog box (Options > Preferences) (on page 124) and go to Editor > Edit modes > Grid.

The following options are available:

Compact representation

If selected, the compact representation of the grid is used: a child element is displayed beside the parent element. In the non-compact representation, a child element is nested below the parent.

Format and indent when passing from grid to text or on save

If selected, the content of the document is formatted and indented (on page 555) each time you switch from the Grid view to the Text view.

Default column width (characters)
Sets the default width (in characters) of a table column of the grid. A column may contain the following:

- Element names
- Element text content
- Attribute names
- Attribute values

If the total width of the grid structure is too large you can resize any column by dragging the column margins with the mouse pointer, but the change is not persistent. To make it persistent, set the new column width with this option.

**Active cell color**

Allows you to set the background color for the active cell (on page 3216) of the grid. The keyboard input always goes to the active cell and the selection always contains it.

**Selection color**

Allows you to set the background color for the selected cells of the grid, except the active cell (on page 3216).

**Border color**

Allows you to set the color used for the lines that separate the grid cells.

**Background color**

Allows you to set the background color of grid cells that are not selected.

**Foreground color**

Allows you to set the text color of the information displayed in the grid cells.

**Row header colors**

- **Background color**
  
  Allows you to set the background color of row headers that are not selected.

- **Active cell color**
  
  Allows you to set the background color of the row header cell that is currently active.

- **Selection color**
  
  Allows you to set the background color of the header cells corresponding to the currently selected rows.

**Column header colors**

The column headers are painted with two color gradients, one for the upper 1/3 part of the header and the other for the lower 2/3 part. The start and end colors of the first gradient are set with the first two color buttons. The start and end colors of the second gradient are set with the last two color buttons.
Background color
Allows you to set the background color of column headers that are not selected.

Active cell color
Allows you to set the background color of the column header cell that is currently active.

Selection color
Allows you to set the background color of the header cells corresponding to the currently selected columns.

Author Preferences
Oxygen XML Editor provides an Author editing mode (on page 354) that provides a configurable graphical interface for editing XML documents. To configure the options for the Author mode, open the Preferences dialog box (Options > Preferences) (on page 124) and go to Editor > Edit modes > Author.

The following options are available:

Author default background color
Sets the default background color of the Author editing mode. The background-color property set in the CSS file associated with the currently edited document overwrites this option.

Author default foreground color
Sets the default foreground color of the Author editing mode. The color property set in the CSS file associated with the currently edited document overwrites this option.

Show XML comments
When this option is selected, XML comments are displayed in Author mode. Otherwise, they are hidden.

Show placeholders for empty elements
When this option is selected, placeholders are displayed for elements with no content to make them clearly visible. The placeholder is rendered as a light gray box and displays the element name.

Show processing instructions
When this option is selected, XML processing instructions are displayed in Author mode. Otherwise, they are hidden.

Show Author layout messages
When this option is selected, all errors reported while rendering the document in Author mode are presented in the Results panel (on page 548) at the bottom of the editor.

Show doctype
When this option is selected, the doctype declaration is displayed in Author mode. Otherwise, it is hidden.
Show block range

When this option is selected, a block range indicator is displayed in a stripe located in the left side of the editor. It is displayed as a heavy line that spans from the first line to the last line of the block.

Fast text layout

In certain cases, the widths computed in the Author visual editing mode for lines of text may be larger than expected, leading to an incorrect visual layout. Deactivating this option will improve the computation quality for character widths in the visual editing mode, but it may hinder overall performance for very large documents.

Tip:

For macOS users, some specific examples of this type of situation can be found here: Text Rendering Issues on macOS (on page 2884). Because of such problems, when an installation kit with Java 9 or newer is used on macOS, the checkbox is not selected by default.

Show floating contextual toolbar

When this option is selected (default), the floating contextual toolbar is displayed in the Author mode in certain situations. When not selected, the floating contextual toolbar is never displayed.

Images Section

The following options regarding images in Author mode are available in this section:

Auto-scale images wider than (pixels)

Sets the maximum width that an image will be displayed. Wider images will be scaled to fit.

Show very large images

When this option is selected, images larger than 6 megapixels are displayed in Author mode. Otherwise, they are not displayed.

Important:

If you select this option and your document contains many such images, Oxygen XML Editor may consume all available memory, throwing an OutOfMemory error. To resolve this, increase the available memory limit (on page 339) and restart the application.

Tags Section

In this section, you can configure the following options regarding tags that are displayed in Author mode:
Tags display mode

Sets the default display mode for element tags presented in Author mode. You can choose between the following:

- **Full Tags with Attributes**
  Displays full tag names with attributes for both block (on page 3216) and inline elements (on page 3220).

- **Full Tags**
  Displays full tag names without attributes for both block elements and inline elements.

- **Block Tags**
  Displays full tag names for block elements and simple tags without names for inline elements.

- **Block Tags without Element Names**
  Displays tags for block elements but without element names for a more compact version of Block Tags mode. You can still see the element names by hovering over the tags.

- **Inline Tags**
  Displays full tag names for inline elements, while block elements are not displayed.

- **Partial Tags**
  Displays simple tags without names for inline elements, while block elements are not displayed.

- **No Tags**
  No tags are displayed. This is the most compact mode and is as close as possible to a word-processor view.

Sort attributes alphabetically for "Full Tags with Attributes"

When selected, if you choose Full Tags with Attributes for the Tags Display Mode (on page 594), the attributes will be displayed in alphabetical order. Otherwise, they are displayed in the order that they appear in the XML source code.

Tags background color

Sets the Author mode tags background color.

Tags foreground color

Sets the Author mode tags foreground color.

Tags font
Allows you to change the font used to display tags text in the Author visual editing mode. The default font is computed based on the setting of the Author default font option in the Fonts preferences page (on page 133).

Compact tag layout

If this option is not selected, the Author mode displays the tags in a more decompressed layout, where block tags are displayed on separate lines.

References Section

Display referenced content (entities, XInclude, DITA conref, etc.)

When selected, the references (such as entities, XInclude, DITA conrefs) also display the content of the resources they reference. If you toggle this option while editing, you need to reload the file for the modification to take effect.

Allow referenced content to be edited

When selected, for a specific XML vocabulary that supports this feature, the content referenced from other files and presented in the Author visual editing mode can be edited in-place and saved. For now, if the feature is enabled and you use the Open Map in Editor with Resolved Topics toolbar action in the DITA Maps Manager view, the referenced content in the opened document becomes editable in-place. Saving the document will save all other modified topics.

Local files only

When selected (default), the Allow referenced content to be edited (Experimental) option only works for local files. For files located in remote locations such as a CMS, additional steps might be necessary to save all modified content because this feature might not function properly with remote resources.

For advanced Author configuration see the Document Type Association settings

Click this link to open the Document Type Association preferences page (on page 138).

Cursor Navigation Preferences

Oxygen XML Editor allows you to configure the appearance and behavior of the cursor in the Author mode editor (on page 354). To set cursor navigation preferences, open the Preferences dialog box (Options > Preferences) (on page 124) and go to Author > Cursor Navigation.

The following options are available:

Highlight elements near cursor

When this option is selected, the element that contains the cursor is highlighted. If the cursor is between two elements, both of them are highlighted. You can use the color picker to choose the color of the highlight.

Show cursor position tooltip
Oxygen XML Editor uses tooltips in Author mode to indicate the position of the cursor in the element structure (on page 596) of the underlying document. Depending on context, the tooltips may show the current element name or the names of the elements before and after the current cursor position.

**Show location tooltip on mouse move**

When this option is selected, Oxygen XML Editor displays Location Tooltips (on page 598) when you are editing the document in certain tags display modes (Inline Tags, Partial Tags, No Tags) or when the mouse pointer is moved between block elements (on page 3216).

**Quick up/down navigation**

This option is deselected by default and this means that when you navigate using the up and down arrow keys in Author mode, the cursor is placed within each of the underlying XML elements between two blocks of text (the cursor changes to a horizontal line when it is between blocks of text). This allows you to easily insert elements and manage the structure of your XML content. However, if this option is selected, the cursor ignores the XML structure and jumps from one line of text to another, similar to how the cursor behaves in a word processor.

**Quick navigation in tables**

This option is selected by default and this means that when navigating between table cells with the arrow keys, the cursor jumps from one cell to another. If this option is not selected, the cursor navigates between XML nodes when navigating between table cells with the arrow keys.

**Avoid positioning the cursor between blocks after a deletion**

If selected (default), the cursor will not stay between block element sentinels after a deletion is performed.

**Arrow keys move the cursor in the writing direction**

This setting determines how the left and right arrow keys behave in Author mode for bidirectional (BIDI) text. When this option is selected (default value), the right arrow key advances the cursor in the reading direction and the left arrow moves it in the opposite direction. When this option is not selected, pressing the right arrow will simply move the cursor to the right (and the left arrow moves it to the left), regardless of the text direction.

**Schema-Aware Preferences**

Oxygen XML Editor can use the schema of your XML language to improve the way the Author (on page 354) mode editor handles your content. To configure the Schema-Aware options, open the Preferences dialog box (Options > Preferences) (on page 124) and go to Editor > Edit modes > Author > Schema-Aware.

The following options are available:

**Schema-aware normalization, format, and indent**

When you open or save a document in Author mode, white space is normalized using the display property of the current CSS stylesheet and the values of the settings (on page 205) for Preserve space elements, Default space elements, and Mixed content elements. When
this option is selected, the schema will also be used to normalize white space, based on the content model (*element-only*, *simple-content*, or *mixed*). Note that the schema information takes precedence.

**Indent blocks-only content**

To avoid accidentally introducing inappropriate white space around *inline elements (on page 3220)*, Oxygen XML Editor does not normally apply indenting to the source of an element with mixed content. If this option is selected, Oxygen XML Editor will apply indenting to the source of mixed content elements that only contain *block elements (on page 3216)*.

**Schema-Aware Editing**

The options in this section determine how Oxygen XML Editor will use the schema of a document to control the behavior of the **Author** mode.

- **On** - Enables all schema-aware editing options.
- **Off** - Disables all schema-aware editing options.
- **Custom** - Allows you to select custom schema-aware editing options from the following:

**Schema-Aware Actions section**

**Delete element tags with backspace and delete**

Controls what happens when you attempt to delete an element tag. The two options are:

- **Smart delete** - If deleting the tag would make the document invalid, Oxygen XML Editor will attempt to make the document valid by unwrapping the current element or by appending it to an adjacent element where the result would be valid. For instance, if you delete a bold tag, the content can be unwrapped and become part of the surrounding paragraph, but if you delete a list item tag, the list item content cannot become part of the list container. However, the content could be appended to a preceding list item.

- **Reject action when its result is invalid** - A deletion that would leave the document in an invalid state is rejected.

**Paste and Drag and Drop**

Controls the behavior for paste and drag and drop actions. Available options are:

- **Smart paste and drag and drop** - If the content inserted by a paste or drop action is not valid at the cursor position, according to the schema, Oxygen XML Editor tries to find an appropriate insert position. The possibilities include:
Creating a sibling element that can accept the content (for example, if you tried to paste a paragraph into an existing paragraph).

Inserting the content into a parent or child element (for example, if you tried to paste a list item into an existing list item, or into the space above or below and existing list).

Inserting the content into an ancestor element where it would be valid.

- **Reject action when its result is invalid** - If selected, Oxygen XML Editor will not let you paste content into a position where it would be invalid.

### Typing

Controls the behavior that takes place when typing. Available options are:

- **Smart typing** - If typed characters are not allowed in the element at the cursor position, but the previous element does allow text, then a similar element will be inserted, along with your content.

- **Reject action when its result is invalid** - If selected, and the result of the typing action is invalid, the action will not be performed.

### Content Completion

Controls the behavior that takes place when inserting elements using the [Content Completion Assistant in Author mode](on page 616). Available options are:

- **Press ENTER to show available content completion proposals** - If selected, pressing **Enter** will open the *Content Completion Assistant*. If deselected, there are three possibilities:
  - The current element will be split *(if possible)*.
  - A new element with the same name will be inserted *(if possible)*.
  - Otherwise, a new paragraph will be inserted.

- **Show all possible elements in the content completion list** - If selected, the content completion list will show all the elements in the schema, even those that cannot be entered validly at the current position. If you select an element that is not valid at the current position, Oxygen XML Editor will attempt to find a valid location to insert it and may present you with several options.

- **Allow only insertion of valid elements and attributes** - If selected, you can only select elements in the content completion list that are valid *(according to the schema)* at the current position.

- **Allow only insertion of valid attribute values** - If selected, you cannot enter an attribute value that is not valid *(according to the schema)* in the *Attributes view* *(on page 627)* or *In-place Attributes Editor* *(on page*
If the attribute has a choice of values, you can select a possible value from a drop-down list in the combo box, but you cannot enter a value manually.

**Warn on invalid content when performing action**

A warning message will be displayed when performing an action that will result in invalid content. Available options are:

- **Delete Element Tags** - If selected, a warning message will be displayed if the Delete Element Tags (on page 759) action will result in an invalid document. You will be asked to confirm the deletion.
- **Join Elements** - If selected, a warning message will be displayed if the Join Elements (on page 759) action will result in an invalid document. You will be asked to confirm the join.

**Automatically apply the best schema-aware insertion operation**

If selected, Oxygen XML Editor automatically uses what it considers to be the best insertion solution, when there is an attempt to insert content that is not valid in a specific context. If not selected, Oxygen XML Editor will ask the user to choose from a list of proposed solutions.

**Convert external content on paste**

If selected, the Smart Paste feature (on page 613) is enabled when external content is pasted in Author mode.

**Convert even when pasting inside space-preserve elements**

If selected, the Smart Paste feature (on page 613) will be used even when external content is pasted inside a space-preserve element (such as a `<codeblock>`).

**Convert pasted URLs to links**

If selected, when a URL is pasted into Author mode, a link will be inserted (the type of link depends on the type of document). For example, in DITA documents, an `<xref>` is inserted.

### Related information

- Smart Paste in Author Mode (on page 613)
- Customizing Smart Paste Support (on page 2188)

### Review Preferences

Oxygen XML Editor allows you to add review comments and track changes (on page 642) in your documents. The Review preferences page allows you to control how the Oxygen XML Editor review features work. To configure these options, open the Preferences dialog box (Options > Preferences) (on page 124) and go to Editor > Edit modes > Author > Review.
The available options are as follows:

**Author**

Specifies the name to be attached to all comments and to changes made while **Track Changes** is active. By default, Oxygen XML Editor uses the system user name.

**Track Changes section (applies for all authors)**

**Initial state**

Specifies whether or not the **Track Changes feature (on page 3224)** is enabled when you open a document. You may have the **Track Changes** feature enabled in some documents and disabled in others, or you can choose to always enable or disable the feature for all documents. You can choose between the following options:

- **Stored in document** - The current state of the **Track Changes** feature is stored in the document itself, meaning that it is on or off depending on the state the last time the document was saved. This is the recommended setting when multiple authors work on the same set of documents as it will make it obvious to other authors that changes have been made in the document.
- **Always On** - The **Track Changes** feature is always on when you open a document. You can turn it off for an open document, but it will be turned on for the next document you open.
- **Always Off** - The **Track Changes** feature is always off when you open a document. You can turn it on for an open document, but it will be turned off for the next document you open.

**Initial display mode**

Specifies whether or not all tracked changes and comments are visible when you open a document in the **Author** visual editing mode. You can choose between the following options:

- **View All Changes/Comments** - All tracked changes and comments are visible in the **Author** visual editing mode.
- **View Final** - Comments are hidden, while insertion and deletion track changes are presented as if they would be accepted.

**Display changed lines marker**

A changed line maker is a vertical line on the left side of the editor window indicating where changes have been made in the document. To hide the changed lines marker, deselect this option.

**Inserted content color**
When the Track Changes feature (on page 3224) is on, the newly inserted content is highlighted with an insertion marker that uses a color to adjust the following display properties of the inserted content: foreground, background, and underline. This section allows you to customize the following color options:

- **Automatic** - If this option is selected, Oxygen XML Editor automatically assigns a color to each user who inserted content in the current document. The colors are picked from the Colors for automatic assignment list (on page 186), the priority being established by the type of change (deletion, insertion, or comment) and in the order that you see in the list.
- **Fixed** - If this option is selected, Oxygen XML Editor uses the specified color for all insertion markers, regardless of who the author is.
- **Use same color for text foreground** - If selected, Oxygen XML Editor uses the color defined above (Automatic or Fixed) to render the foreground of the inserted content.
- **Use same color for background** - If selected, Oxygen XML Editor uses the color defined above (Automatic or Fixed) to render the background of the inserted content. A slider control allows you to set the transparency level of the background.

### Deleted content color

When the Track Changes feature (on page 3224) is on, the deleted content is highlighted with a deletion marker that uses a color to adjust the following display properties of the deleted content: foreground, background, and strikethrough. This section allows you to customize the following color options:

- **Automatic** - If this option is selected, Oxygen XML Editor automatically assigns a color to each user who deleted content in the current document. The colors are picked from the Colors for automatic assignment list (on page 186), the priority being established by the type of change (deletion, insertion, or comment) and in the order that you see in the list.
- **Fixed** - If this option is selected, Oxygen XML Editor uses the specified color for all deletion markers, regardless of who the author is.
- **Use same color for text foreground** - If selected, Oxygen XML Editor uses the color defined above (Automatic or Fixed) to render the foreground of the deleted content.
- **Use same color for background** - If selected, Oxygen XML Editor uses the color defined above (Automatic or Fixed) to render the background of the deleted content. A slider control allows you to set the transparency level of the background.

### Comments color section (applies for all authors)

Sets the background color of the text that is commented on. The options are:
• **Automatic** - If this option is selected, Oxygen XML Editor automatically assigns a color to each user who adds a comment in the current document. The colors are picked from the **Colors for automatic assignment list** (on page 186), the priority being established by the type of change (deletion, insertion, or comment) and in the order that you see in the list.

• **Fixed** - If this option is selected, Oxygen XML Editor uses the specified color for all changes, regardless of who the author is. A slider control allows you to set the transparency level of the background.

**Colors for automatic assignment list**

These are the colors that will be automatically assigned for tracked insertions, tracked deletions, and comments if the **Automatic** option is selected in any of the sections in this preferences page. The colors are assigned in the order that you see in this list. You can use the **Add**, **Edit**, or **Remove** buttons to modify the list of colors.

**Related information**

**Reviewing Documents** (on page 642)

---

**Callouts Preferences**

Oxygen XML Editor can display **callouts** (on page 3217) for review items such as comments and **tracked changes** (on page 642). To customize options for review callouts, open the **Preferences** dialog box (Options > Preferences) (on page 124) and go to Editor > Edit modes > Author > Review > Callouts.

The available options are as follows:

**Show Review Callouts section**

- **Comments**
  
  If selected, callouts are displayed for comments, including comments that are added to **tracked changes** (on page 3224). This option is selected by default.

- **Track Changes deletions**
  
  If selected, callouts are displayed for **tracked change** (on page 3224) deletions and the following additional option becomes available:

  - **Show deleted content in callout**
    
    If selected, the deleted content is also displayed in the callout.

- **Track Changes insertions**
  
  If selected, callouts are displayed for **tracked change** (on page 3224) insertions and the following additional option becomes available:

  - **Show inserted content in callout**
If selected, the inserted content is also displayed in the callout.

**Rendering section**

**Show review time**
When selected, timestamp information is displayed in callouts.

**Show all connecting lines**
When selected, lines are shown that connect the callout to the location of the change.

**Initial width (px)**
Specifies the initial width of the callouts each time the document is opened. The default is 250 pixels.

**Text lines count limit**
Specifies the maximum number of lines to be shown in the callouts. The default is 5 lines. Note that this does not limit the number of lines in the actual comment. It only limits the number of lines shown without opening or editing it. To see the full comment, right-click on the callout and select Edit Comment or Show Comment.

**Profiling/Conditional Text Preferences**

Oxygen XML Editor lets you configure how profiling and conditional text (on page 668) is displayed in Author mode. It has built-in support for the standard conditional text features of DITA and DocBook that you can customize for your own projects. You can also add conditional support for other XML vocabularies, including your custom vocabularies.

To configure Profiling/Conditional Text options, open the Preferences dialog box (Options > Preferences) (on page 124) and go to Editor > Edit modes > Author > Profiling/Conditional Text. There are several subpages in this section. This first parent page includes options for determining which types of profiled content is displayed:

**Show profiling attributes**
Toggles whether or not the Show Profiling Attributes option (on page 679) in the Profiling / Conditional Text drop-down menu is enabled by default.

**Show profiling attribute name**
If selected, the names of the profiling attributes are displayed with their values. If unchecked, only the values are displayed.

**Show profiling colors and styles**
Toggles whether or not the Show Profiling Colors and Styles option (on page 679) in the Profiling / Conditional Text drop-down menu is enabled by default.
Show excluded content

Toggles whether or not the Show Excluded Content option (on page 680) in the Profiling / Conditional Text drop-down menu is enabled by default.

Attributes and Condition Sets Preferences

To configure profiling attributes and condition sets, open the Preferences dialog box (Options > Preferences) (on page 124) and go to Editor > Edit modes > Author > Profiling/Conditional Text > Attributes and Condition Sets.

Note:

Note the following when configuring these settings:

• This preferences page is used to define how profiled elements are treated in Author mode. It does not create profiling or conditional text attributes or values in the underlying XML vocabulary. It just changes how the editor displays them.
• This preferences page should be used for profiling / conditional text elements only. To change how other types of attributes are displayed in the text, use a CSS file.
• If you are using the DITA XML vocabulary and a DITA subject scheme map (on page 3224) is defined in the root map (on page 3223) of your document, it will be used in place of anything defined using this dialog box.

This preferences page contains the following options and sections:

Import from DITAVAL

This button allows you to import profiling attributes from DITAVAL files (on page 3155). You can merge these new profiling attributes with the existing ones, or replace them completely. If the imported attributes conflict with the existing ones, Oxygen XML Editor displays a dialog box that contains two tables. The first one previews the imported attributes and the second one previews the already defined attributes. You can choose to either keep the existing attributes or replace them with the imported ones.

Note:

When importing profiling attributes from DITAVAL files, Oxygen XML Editor automatically creates condition sets based on these files.

Profiling Attributes section

Allows you to specify a set of allowable values for each profiling or conditional attribute. You can use the New button at the bottom of the table to add profiling attributes (on page 670), the Edit button to edit existing ones, or the Delete button to delete entries from the table. Use the Up and Down buttons to change the priority of the entries. If you have multiple entries with identical
names that match the same document type, Oxygen XML Editor uses the one that is positioned highest in the table.

**Report invalid profiling attribute values (DITA only)**

If selected, it means the following:

- In DITA, the automatic validation will display a warning when a value that is not defined is found in the document.
- In the DITA Validate and Check for Completeness dialog box, the Report attributes and values that conflict with profiling preferences (on page 2946) option is not displayed. This means that the validation will behave the same as if that option was selected and it will always report such values.

**Allow contributing extra profiling attribute values**

This option is selected by default, which means that users are allowed to add values that are not defined in preferences to profiling attributes. If a user inserts such a value, when invoking the Edit Profiling Attributes action from the contextual menu in Author mode (or for DITA topics, the Edit Properties action in the DITA Maps Manager (on page 2897)), the Profiling Values Conflict dialog box (on page 672) will appear and it includes an Add these values to the configuration action that will automatically add the new value to the particular profiling attribute. If deselected, Oxygen XML Editor behaves as if the Preserve the configuration option has been chosen in the Profiling Values Conflict dialog box (on page 672) and that dialog box will never appear.

**Configure profiling colors and styles link**

Use this link to open the profiling Colors and Styles preference page (on page 190).

**Profiling Condition Sets section**

Allows you to specify a specific set of profiling attributes to be used to specify a particular build configuration for your content. You can use the New button at the bottom of the table to add condition sets (on page 675), the Edit button to edit existing ones, or the Delete button to delete entries from the table. Use the Up and Down buttons to change the priority of the entries. If you have multiple entries with identical names that match the same document type, Oxygen XML Editor uses the one that is positioned highest in the table.

**Related information**

Filtering Profiling Values with a DITAVAL File (on page 3155)
Styling the Rendering of Profiled Content Using a DITAVAL File (on page 3158)
Colors and Styles Preferences

Oxygen XML Editor lets you set the colors and styles used to display profiling / conditional text (on page 187) in the Author mode editor (on page 354). To set Colors and Styles preferences, open the Preferences dialog box (Options > Preferences) (on page 124) and go to Editor > Edit modes > Author > Profiling/Conditional Text > Colors and Styles.

The preference page includes the following options and sections:

**Import from DITAVAL**

Allows you to import profiling styles from a .ditaval file. You can merge these new profiling styles with the existing ones, or replace them completely. If the imported styles conflict with the existing ones, Oxygen XML Editor displays a dialog box containing two tables. The first one previews the imported styles, while the second one previews the already defined styles. You can choose to either keep the existing styles or replace them with the imported ones. This feature works even if you use profiling attribute groups to organize the attributes into subcategories (on page 3149).

**Profiling Colors and Styles Table**

You can use buttons below this table to set specific colors and styles for the listed profiling attribute values. The table includes two categories:

- **Defined attributes values** - Contains the styles for profiling attribute values defined in the Profiling / Conditional Text (on page 187) preferences page. Each profiling attribute value has an associated style. To ease the process of customizing styles, the Defined attributes values category contains by default the list of empty styles. All you have to do is to adjust the colors and decorations, thus skipping the process of manually defining the association rules (document type, attribute name, and value). This is the reason why a style from this category can only be reset (on page 191), not deleted.

- **Other** - This category contains styles for attribute values that are not marked as profiling values, in the Profiling / Conditional Text (on page 187) preferences page. In this category are listed:
  - All the styles that were defined in other projects (with other profiling attribute value sets).
  - All the styles set for the profiling attributes defined in a subject scheme map (on page 3151).

**Automatic styling button**

If you click this button, Oxygen XML Editor will apply automatic styling to the profiling attribute values that do not have a style defined.

**New button**

Opens the Add Profiling Style dialog box that allows you to associate a set of coloring and styling properties to a profiling value.
### Attributes Preferences

When the [Show Profiling Attributes](#) option is selected, the Author mode displays conditional text markers at the end of conditional text blocks. To configure the rendering of these text markers, open the Preferences dialog box (Options > Preferences) and go to Editor > Edit modes > Author > Profiling/Conditional Text > Colors and Styles > Attributes.

The following options are available:

- **Background color**
  - Sets the background color used to display the profiling attributes.

- **Attribute name foreground color**
  - Sets the foreground color used to display the names of the profiling attributes.

- **Attribute values foreground color**
  - Sets the foreground color used to display values of the profiling attributes.

- **Border color**
  - Sets the color of the border of the block that displays the profiling attributes.

### MathML Preferences

Oxygen XML Editor allows you to edit [MathML](#) equations and displays the results in a preview window. For a more specialized MathML editor, you can either install Wiris MathFlow, which is
a commercial product that requires a separate license, or use an external MathML editor (e.g. the LibreOffice equation editor).

**Using MathFlow for Editing and Rendering MathML Equations**

To configure the MathML editor or to enter your MathFlow license information, open the Preferences dialog box (Options > Preferences) (on page 124) and go to Editor > Edit modes > Author > MathML.

You can configure the following options:

- **Equation minimum font size**
  The minimum size of the font used for rendering mathematical symbols when editing in the Author mode.

- **MathFlow installation directory**
  The installation folder for the MathFlow components product (MathFlow SDK).

- **MathFlow license file**
  The license file for the MathFlow components product (MathFlow SDK).

- **MathFlow preferred editor**
  A MathML formula can be edited in one of three editors of MathFlow components product (MathFlow SDK).
    - **Structure Editor** (default selection) - Targets professional XML workflow users.
    - **Style Editor** - Tailored to the needs of content authors.
    - **Simple Editor** - Designed for applications where end-users can enter mathematical equations without prior training and only the meaning of the math matters.

- **Save special characters**
  Specifies how special characters are saved in the XML file.
    - **As entity names** - Saves the characters in \&name; format. It refers to a character by the name of the entity that has the desired character as its replacement text. For example, the Greek Omega character is saved as \&Omega;.
    - **As character entities** (default selection) - Saves the characters in a hexadecimal value, using the \&xNNNN; format. For example, the Greek Omega character is saved as \&x3a9;.
    - **As character values** - Saves the characters as the actual symbol. For example, the Greek Omega character is saved as \Ω.

More documentation is available on the Wiris MathFlow website.

**Using an External Tool for Editing MathML Equations**

- **External application > Command line**
You can use this option to specify an external MathML application for editing MathML equations. For example, the following commands could be used to edit MathML equations with a LibreOffice application (depending on the O.S.):

- **Windows** - "C:\Program Files\LibreOffice\program\smath.exe" --nologo "${cf}"
- **macOS** - /Applications/LibreOffice.app/Contents/MacOS/soffice --math --nologo "${cf}"
- **Linux** - /usr/lib/libreoffice/program/smath --nologo "${cf}"

### AutoCorrect Preferences

Oxygen XML Editor includes an option to automatically correct misspelled words as you type in **Author** mode. To enable and configure this **AutoCorrect** feature (on page 460), open the **Preferences** dialog box (Options > Preferences) (on page 124) and go to Editor > Edit Modes > Author > AutoCorrect.

The following options are available:

**Enable AutoCorrect**

When selected (default state), while editing in **Author** mode, if you type anything that is listed in the Replace column of the Replacements table displayed in this preferences page, Oxygen XML Editor will automatically replace it with the value listed in the With column.

**Use additional suggestions from the spell checker**

If selected, in addition to anything listed in the Replacements table displayed in this preferences page, Oxygen XML Editor will also use suggestions from the Spell Checker to automatically correct misspelled words. Suggestions from the Spell Checker will only be used if the misspelled word is not found in the Replacements table.

**Note:**

The **AutoCorrect** feature shares the same options configured in the Language options (on page 230) and Ignore elements (on page 231) sections in the Spell Check preferences page.

**Include text-to-markup corrections based on the current document type**

If selected, in addition to anything listed in the Replacements table displayed in this preferences page, the **AutoCorrect** mechanism will also include XML markup insertion rules specified in a configuration file for each document type. For example, for default DITA, DocBook, and TEI documents, entering a hyphen (-) followed by a space in an empty paragraph will automatically insert a list element with an empty list item element inside. The configuration file is located at: `{OXYGEN_INSTALL_DIR}/frameworks/{DOC_TYPE}/resources/structureAutocorrect.xml`.
Tip:
By default, the `structureAutocorrect.xml` file only exists for DITA, DocBook, and TEI frameworks, but it is possible to customize your own markup correction rules for your particular document type. For details, see `Customizing Text-to-Markup Shortcut Patterns` (on page 2187).

Spell Check options link

Use this link to navigate to the Spell Check Preferences page (on page 229).

Replacements Table section

The `AutoCorrect` feature uses the Replacements table to automatically replace anything that is listed in the `Replace` column with the value listed in the `With` column for each language.

**Replacements for language drop-down menu**

You can specify the language for the Replacements table, and for each language, you can configure the items listed in the table. The language selected in this page is not the language that will be used by the `AutoCorrect` feature. It is simply the language for the Replacements table.

**Replacements Table**

You can double-click on cells in either column to edit the listed items. Use the Add button to insert new items and the Remove button to delete rows from the table.

Note:
Any changes, additions, or deletions you make to this table are saved to a path that is specified in the `AutoCorrect Dictionaries` preferences page (on page 196).

Smart quotes section

You can also choose to automatically convert double and single quotes to a quotation character of your choice by using the following options in the Smart quotes section:

- Replace "Single quotes" - Replaces single quotes with the quotation symbols you select with the Start quote and End quote buttons.
- Replace "Double quotes" - Replaces double quotes with the quotation symbols you select with the Start quote and End quote buttons.

Note:
These Smart quotes options are ignored for content inside any element listed in the Ignore elements section of the Spell Check preferences page (on page 231).

Global Options (on page 3219)
If this option is selected, the options are stored on your local computer, in a folder that is not accessible to other users.

**Project Options (on page 3222)**

If this option is selected, the options are stored in the project file and can be shared with other users. Selecting **Project Options (on page 3222)** will only save your selections in **Enable AutoCorrect (on page 193)**, **Use additional suggestions from the spell checker (on page 193)**, and the options in the **Smart quotes** section (on page 194). Changes to the Replacements table are not saved in this page. To save changes to the Replacements table at project level, you need to specify a custom location in the **User-defined replacements** section of the **AutoCorrect Dictionaries** preferences page (on page 196) and select **Project Options** from that preferences page instead.

**Restore Defaults**

Restores the options in this preferences page to their default values and also deletes any changes you have made to the **Replacements table** (on page 194).

**AutoCorrect Dictionaries Preferences**

To set the Dictionaries preferences for the **AutoCorrect feature (on page 460)**, open the **Preferences dialog box (Options > Preferences)** (on page 124) and go to **Editor > Edit Modes > Author > AutoCorrect > Dictionaries**. This page allows you to specify the location of the dictionaries that Oxygen XML Editor uses for the **AutoCorrect** feature and the location for saving user-defined replacements.

The following options are available in this preferences page:

**Dictionaries default folder**

Displays the default location where the dictionaries that Oxygen XML Editor uses for the **AutoCorrect** feature are stored.

**Include dictionaries from**

Selecting this option allows you to specify a location where you have stored **AutoCorrect** dictionaries that you want to include, along with the default ones.

---

**Important:**

Consider the following notes regarding this option:

- The **AutoCorrect** mechanism takes into account **AutoCorrect** dictionaries collected both from the default and custom locations and multiple dictionaries from the same language are merged (for example, `en_UK.dat` from the default location is merged with `en_US.dat` from a custom location).
- If you have a generic **AutoCorrect** dictionary file (one that just has a two-letter language code for its file name, such as `en.dat`) saved in either the default or custom location, the other more specific dictionaries (for example, `en_UK.dat`) are included in the merged dictionary.
and en_US.dat) will not be merged and the existing generic dictionary will simply be used instead.

- You can use a custom suffix in the dictionary file name after the language code. For example, en_US_synopsys.dat or en_synopsys.dat.
- If the additional location contains a file with the same name as one from the default location, the file in the additional location takes precedence over the file from the default location.

How to add more dictionaries link

Use this link to open a topic in the Oxygen XML Editor User Guide that explains how to add dictionaries for the AutoCorrect feature (on page 461).

Save user-defined replacements in the following location

Writes user-defined replacements to the following location. By default, the target is the application preferences folder, but you can also choose a custom location.

Tip:
To save changes to the Replacement table (in the AutoCorrect preferences page) (on page 194) at project level (on page 3222), select a custom location for the User-defined replacements and select Project Options (on page 3222) at the bottom of the page.

Related information
Add Dictionaries for the AutoCorrect Feature (on page 461)

Serialization Preferences

To configure the serialization options for the Author mode, open the Preferences dialog box (Options > Preferences) (on page 124) and go to Editor > Edit modes > Author > Serialization.

The following options are available:

Format and indent

Use this option to specify what should be formatted and indented when you save a document (or switch from Author to Text mode). You can choose between the following two options:

Only the modified content
The Save operation only formats the nodes that were modified in the Author mode. The rest of the document preserves its original formatting.
The entire document

The **Save** operation applies the formatting to the entire document regardless of the nodes that were modified in **Author** mode.

**Also apply the format and indent options that are set for Text mode**

If this option is selected, the result of the **Format and indent** operation will be the same as when it is applied in **Text** editing mode. Thus, the content of the document is formatted by applying the **Format and Indent** rules from the **Editor/Format (on page 202)** and **Editor/Format/XML (on page 205)** preference pages. You can use the **Format and indent options** link to navigate to those options.

**Compatibility with other tools**

Use this option to control how line breaks are handled when a document is serialized. This will help to obtain better compatibility with other tools. You can choose one of the following:

- **None** - Choose this option if compatibility with other tools can be ignored.
- **Do not break lines, do not indent** - Choose this option to avoid breaking lines after element start or end tags and indenting will not be used.

**Note:**

New lines that are added by the user in elements where the `@xml:space` attribute is set to `preserve` (such as `<pre>` elements in HTML, or `<codeblock>` elements in DITA) are still inserted. Also, selecting this option automatically disables the **Also apply the format and indent options that are set for Text mode** option **(on page 197)**, since the formatting from **Text** mode does not take the CSS styles into account.

- **Break lines only after elements displayed as blocks, do not indent** - Choose this option to instruct Oxygen XML Editor to insert new lines only after elements that have a CSS display property set to anything other than `inline` or `none` (for example, `block`, `list-item`, `table`, etc.) and indenting will not be used. When selecting this option, the formatting is dictated by the CSS.

**Note:**

New lines that are added by the user in elements where the `@xml:space` attribute is set to `preserve` (such as `<pre>` elements in HTML, or `<codeblock>` elements in DITA)
are still inserted. Also, selecting this option automatically disables the **Also apply the format and indent options that are set for Text mode** option *(on page 197)*, since the formatting from **Text** mode does not take the CSS styles into account.

### Schema Design Preferences

Oxygen XML Editor provides a graphical schema design editor *(on page 355)* to make editing XML Schema easier. To configure the **Schema Design** options, open the **Preferences** dialog box *(Options > Preferences)* *(on page 124)* and go to **Editor > Edit modes > Schema Design**.

The following options are available in the **Schema Design** preferences page:

**Show annotation in the diagram**

When selected, Oxygen XML Editor displays the content of documentation in schema diagrams.

**When trying to edit components from another schema**

The schema diagram editor will combine schemas imported by the current schema file into a single schema diagram. You can choose what happens if you try to edit a component from an imported schema. The options are:

- **Always go to its definition** - Oxygen XML Editor opens the imported schema file so that you can edit it.
- **Never go to its definition** - The imported schema file is not opened and the component cannot be edited in place.
- **Always ask** - Oxygen XML Editor asks if you want to open the imported schema file.

**Zoom**

Allows you to set the default zoom level of the schema diagram.

### XSD Properties Preferences

Oxygen XML Editor lets you control which properties to display for XML Schema components in the XML Schema Design view *(on page 355)*. To configure the schema design properties displayed, open the **Preferences** dialog box *(Options > Preferences)* *(on page 124)* and go to **Editor > Edit modes > Schema Design > XSD Properties**.

This preferences page contains the following:

**Show additional properties in the diagram**

If this option is selected, the properties selected in the property table are shown in the XML Schema Design mode. This option is selected by default.

**Properties Table**

**Show**
Use this column in the table to select the properties that you want to be displayed in the XML Schema Design mode.

**Only if specified**

Use this column to select if you want the property to be displayed only if it is defined in the schema.

### JSON Schema Properties Preferences

Oxygen XML Editor lets you control which properties to display for JSON Schema components in the JSON Schema Design mode. To configure the JSON properties that are displayed, open the Preferences dialog box (Options > Preferences) (on page 124) and go to Editor > Edit modes > Schema Design > JSON Schema Properties.

This preferences page contains the following:

- **Show additional properties in the diagram**
  
  If this option is selected, the properties selected in the property table are shown in the JSON Schema Design mode. This option is selected by default.

- **Properties Table**
  
  - **Show**
    
    Use this column in the table to select the properties that you want to be displayed in the JSON Schema Design mode.
  
  - **Only if specified**
    
    Use this column to select if you want the property to be displayed only if it is defined in the schema.

### Open Preferences

Oxygen XML Editor lets you control how files are opened. To configure the options for opening documents, open the Preferences dialog box (Options > Preferences) (on page 124) and go to Editor > Open.

The following options are available:

- **Open each document in a tab next to the current one**
  
  When selected (default), each new document is opened in a tab next to the currently open tab. If not selected, each new document is opened in a tab at the end of the current tab stack.

- **Restore cursor position**
  
  Selected by default, it ensures that the last position of the cursor will be remembered when a document is re-opened. If this option is not selected, the cursor will always be positioned at the beginning of the document.

- **Lock local resources**
When this option is selected and you open a file from the local file system or a shared network drive, Oxygen XML Editor locks the file for the current user and the file becomes read-only for other users while the lock exists. Locked and read-only files have a lock icon (🔒) displayed on their editor tabs. Newly created files are locked when you first save them. If you select this option with files already opened in Oxygen XML Editor, it will lock all the currently open files. If you deselect this option with files already open, it will unlock them by deleting the corresponding .lock files. When you try to save locked (read-only) files, a Save As dialog box will be displayed to avoid overwriting the initial resource.

**Support for Special Characters section**

![Note:](image)
The options in this section only affect the Text editing mode.

When bidirectional text, Asian languages, or other special characters are detected

You can choose how you want Oxygen XML Editor to handle bidirectional text, Asian languages, or other special characters when they are detected in Text mode. You can choose one of the following:

- **Enable support for special characters** - The support for special characters will always be enabled. For details about what this means, see Bidirectional Text Support in Text Mode - Enabled (on page 564).
- **Disable support for special characters** - The support for special characters will always be disabled. For details about what this means, see Bidirectional Text Support in Text Mode - Disabled (on page 564).
- **Prompt for each document** (default setting) - You will be prompted to choose whether or not to enable the support for special characters whenever they are detected in a newly opened document. For details about which setting to choose, see Special Character Support in Text Mode (on page 564).

**Disable special characters support for documents larger than (characters)**

Enabling bidirectional text editing support can affect performance on large files. When this option is selected, bidirectional editing is disabled for files exceeding the specified size (even if the Enable support for special characters option (on page 200) is selected). The default limit is 300 MB. You can change it to 500 MB or 800 MB, but it is recommended that you always leave this option selected regardless of the limit that is set.

**Performance section**

Optimize loading in the Text edit mode for files over (MB)
File loading is optimized for reduced memory usage for any file whose size is larger than the value specified in this drop-down list. This is useful for editing large files, but there are several restrictions (on page 469) for memory-intensive operations.

**Show warning when loading large documents**

Oxygen XML Editor will warn you if you open a file that is bigger than the specified size.

**Optimize loading for documents with lines longer than (Characters)**

*Line wrap* is automatically performed for documents that contain lines that exceed the length specified in this text field. For a list of the restrictions applied to a document with long lines, see Editing Documents with Long Lines (on page 471).

**Show warning when loading documents with long lines**

When selected, Oxygen XML Editor will warn you when you open a file with lines longer than the specified length. To reduce the length of lines in a file, format and indent the document (on page 555) after it is opened in the editor panel. For a list of the restrictions applied to a document with long lines, see Documents with Long Lines (on page 471).

**Save Preferences**

Oxygen XML Editor lets you control how files are saved. To configure the options for saving documents, open the Preferences dialog box (Options > Preferences) (on page 124) and go to Editor > Save.

The following options are available:

**Show "Save as" option to save newly created documents in the "New" document wizard**

It is selected by default, but if you deselect this option, the Save as option (on page 370) will not be available in the New Document wizard (on page 368), so you will not have the ability to change the default name and path of the new file.

**Safe save (only for local files)**

In the unlikely event of a failure when attempting a Save action, this option provides increased protection from corruption of the original file. When this option is selected, it saves the content to a temporary file and if the save is unsuccessful, the editor preserves its unsaved state status.

**Automatically save the document every**

If selected, your documents are automatically saved after a preset time interval that is specified in the drop-down list.

**On Save, make backup copy with extension (only for local files)**
If selected, a backup copy is made when saving the edited document. This option is available only for local files (files that are stored on the local file system). The default backup file extension is `.bak`, but that can be changed in the text field.

**Save auto-recover information every**

If selected, your documents are automatically saved to a backup file after the time interval specified in the drop-down list.

**Auto-recover file location**

Specifies the location of the backup file for auto-recovery.

**Validate document before saving**

If selected, Oxygen XML Editor runs a validation that checks your document for errors before saving it.

**Save all files before transformation or validation**

Saves all opened files before validating or transforming an XML document. This ensures that any dependencies are resolved when modifying the XML document and its XML Schema.

**Save all files before calling external tools**

If selected, all files are saved before executing an external tool (on page 2852).

**Automatically compile LESS to CSS when saving**

If selected, when you save a LESS file it will automatically be compiled to CSS (deselected by default).

⚠️ **Important:**

If this option is selected, when you save a LESS file, the CSS file that has the same name as the LESS file is overwritten without warning. Make sure all your changes are made in the LESS file. Do not edit the CSS file directly, as your changes might be lost.

**Performance section**

**Clear undo buffer on save**

If selected, Oxygen XML Editor clears its undo buffer when you save a document. Thus, modifications made prior to saving the document cannot be undone. Select this option if you frequently encounter out of memory errors when editing large documents.

**Format Preferences**

This preferences page contains various formatting options that influence editing and formatting in both the **Text** (on page 353) and **Author** (on page 354) editing modes. To control additional options specifically for the **Author** mode editor, see **Whitespace Handling in Author Mode** (on page 598).
Note:

These settings apply to the formatting of source documents. The formatting of output documents is determined by the transformation scenarios that create them (on page 1432).

To configure the Format options, open the Preferences dialog box (Options > Preferences) (on page 124) and go to Editor > Format.

The following options are available:

**Detect indent on open**

If selected, Oxygen XML Editor detects how a document is indented when it is opened. Oxygen XML Editor uses a heuristic method of detection by computing a weighted average indent value from the initial document content. You can deselect this setting if the detected value does not work for your particular case and you want to use a fixed-size indent for all the edited documents. If this option is selected, Oxygen XML Editor detects the following:

- When TAB characters are used to indent content, the size of the TAB characters is used for the indent size.
- Otherwise, the detected size of SPACE characters is used for the indent size.

**Use zero-indent, if detected**

By default, if no indent was detected in the document, the fixed-size indent is used. Select this option if all of your documents have no indentation and you want to keep them that way.

**Indent with tabs**

If selected, indents are created using TAB characters. If unchecked, lines are indented using space characters. Selecting this option automatically disables the Detect indent on open (on page 203) option.

**Indent size**

The meaning of this setting depends on the following:

- If the Detect indent on open option (on page 203) is selected and TAB characters are detected at the beginning of the line, the indent size is the width of a TAB character. Otherwise, the indent size value is ignored and Oxygen XML Editor uses the number of detected SPACE characters.
• If the **Indent with tabs** option *(on page 203)* is selected, the *indent size* is the width of a TAB character.
• If neither of these options are selected, the *indent size* is the number of SPACE characters used for indenting the text lines.

For additional information about changing the *indent size*, see **Setting an Indent Size to Zero *(on page 560)*.**

For information about when this setting is used, see **Where Indent Size and Line Width Settings are Used in Oxygen XML Editor *(on page 205)*.**

**Indent on enter**

If selected, when you press **Enter** to insert a line break in the **Text** editing mode, an indentation will be added to the new line.

**Enable smart enter**

If selected, when you press the **Enter** key between a start and an end XML tag in the **Text** editing mode, the cursor is placed in an indented position on the empty line formed between the start and end tag.

**Format and indent the document on open**

If selected, an XML document is formatted and indented before opening it in **Oxygen XML Editor**.

**Note:**

Some specialized types of XML documents do not benefit from this feature, including Relax NG, XSD, XSL, and Ant. However, the feature is available for some non-XML types of documents, such as CSS and JSON.

**Detect line width on open**

If selected, **Oxygen XML Editor** automatically detects the line width when the document is opened.

**Hard line wrap (Limit to "Line width - Format and Indent")**

If selected, when typing content in the **Text** editing mode and the maximum line width is reached, a line break is automatically inserted.

**Line width - Format and Indent**

Defines the number of characters after which the **Format and Indent** (pretty-print) action performs hard line-wrapping. For example, if set to 100, after a **Format and Indent** action, the longest line will have a maximum of 100 characters. This setting is also used when saving XML content edited in the **Author** editing mode.
Note:
To avoid having an indent that is longer than the line width setting and without having sufficient space available for the text content, the indent limit is actually set at half the value of the Line width - Format and Indent setting. The remaining space is reserved for text.

For information about when this setting is used, see Where Indent Size and Line Width Settings are Used in Oxygen XML Editor (on page 205).

Clear undo buffer before Format and Indent
The Format and Indent operation can be undone, but if used intensively, a considerable amount of the memory allocated for Oxygen XML Editor will be used for storing the undo states. If this option is selected, Oxygen XML Editor empties the undo buffer before doing a Format and Indent operation. This means you will not be able to undo any changes you made before the format and indent operation. Select this option if you encounter out of memory problems (OutOfMemoryError) when performing the Format and Indent operation.

Where Indent Size and Line Width Settings are Used in Oxygen XML Editor
The values set in the Indent Size and Line Width - Format and Indent options are used in various places in the application, including the following:

• When the Format and Indent action is used in the Text editing mode.
• When you press Enter to break a line in the Text editing mode.
• When the Hard line wrap (Limit to "Line width - Format and Indent") option is selected and the maximum line width is reached while editing in the Text mode.
• When the XML is serialized by saving content in the Author editing mode.

For more information about the formatting options offered by Oxygen XML Editor, watch our video demonstration:

https://www.youtube.com/embed/1plmdN0Cfso

XML Preferences
To configure the XML Formatting options, open the Preferences dialog box (Options > Preferences) (on page 124) and go to Editor > Format > XML.

The following options are available:

Format Section
This section includes the following drop-down boxes:

Preserve empty lines
The Format and Indent operation preserves all empty lines found in the document.
Preserve text as it is

The Format and Indent operation preserves text content as it is, without removing or adding any white space.

Preserve line breaks in attributes

Line breaks found in attribute values are preserved.

Note:
When this option is selected, the Break long attributes option (on page 206) is automatically disabled.

Break long attributes

The Format and Indent operation breaks long attribute values.

Indent inline elements

The inline elements are indented on separate lines if they are preceded by white spaces and they follow another element start or end tag. For example:

Original XML:

```xml
<root>
  text <parent> <child></child> </parent>
</root>
```

Indent inline elements selected:

```xml
<root> text <parent>
  <child/>
  </parent>
</root>
```

Indent inline elements not selected:

```xml
<root> text <parent> <child/> </parent> </root>
```

Expand empty elements

If not selected (default), the Format and Indent operation results in an empty XML element being serialized in a compact form (\(<a atr1="v1"/>\)). If selected, the same operation results in empty XML elements being serialized in expanded form (for example, \(<a atr1="v1"></a>\)).

Notes:

• When using the Format and Indent operation in Text mode, if the Schema-aware format and indent option (on page 208) is enabled, Oxygen XML Editor will use information from the
associated schema and avoid expanding tags for elements that are defined as empty in the schema.

- When saving a document in Author mode, if the Schema-aware normalization, format, and indent option in the Schema-Aware preferences page (on page 180) is enabled, Oxygen XML Editor will use information from the associated schema and avoid expanding tags for elements that are defined as empty in the schema (therefore, text or other elements are not allowed inside them).

Sort attributes

The Format and Indent operation sorts the attributes of an element lexicographically.

Add space before slash in empty elements

Inserts a space character before the trailing / and > of empty elements.

Break line before an attribute name

When selected, the Format and Indent operation always breaks the line before any attribute name in an XML element. By default, the setting is not selected, which means that new lines might still be added before the attribute names but only if the line of content would overflow the maximum line width specified in the Format preferences page (on page 202).

Element Spacing Section

This section controls how the application handles whitespaces found in XML content. You can Add or Remove element names or simplified XPath expressions in the various tabs.

The XPath expressions can accept multiple attribute conditions and inside each condition you can use AND/OR boolean operators and parentheses to override the priority.

You can use one or more of the following attribute conditions (default attribute values are not taken into account):

- `element[@attr]` - Matches all instances of the specified element that include the specified attribute.
- `element[not(@attr)]` - Matches all instances of the specified element that do not include the specified attribute.
- `element[@attr = "value"]` - Matches all instances of the specified element that include the specified attribute with the given value.
- `element[@attr != "value"]` - Matches all instances of the specified element that include the specified attribute and its value is different than the one given.
Example: The following is an example of how you could use multiple boolean operators and parentheses inside an attribute condition:

```
[@a and @b or @c and @d]
[@a and (@b or @c) and @d]
```

The following are just examples of how simplified XPath expressions might look like:

```
* elementName
* //elementName
* /elementName1/elementName2/elementName3
* //xs:localName
Note: The namespace prefixes (such as xs) are treated as part of the element name without taking its binding to a namespace into account.
* //xs:documentation[@lang="en"]
```

The tabs are as follows:

Preserve space

List of elements that will have the Format and Indent operation preserve the whitespaces (such as blanks, tabs, and newlines).

Default space

List of elements that will have the content normalized (multiple contiguous whitespaces are replaced by a single space), before applying the Format and Indent operation.

Mixed content

The elements from this list are treated as mixed content when applying the Format and Indent operation. The lines are split only when whitespaces are encountered.

Line break

List of elements that will have line breaks inserted, regardless of their content. You can choose to break the line before the element, after, or both.

Schema-aware format and indent

The Format and Indent operation takes the schema information into account with regard to the space preserve, mixed, or element only properties of an element.

Indent Section

Includes the following options:

Indent (when typing) in preserve space elements

Normally, the Preserve space elements (identified by the xml:space attribute set to preserve or by their presence in the Preserve space tab of the Element Spacing list (on page 207)) are ignored by the Format and Indent operation. When this option is selected and you edit one of these elements, its content is formatted.
Indent on paste - sections with number of lines less than 300

When you paste a chunk of text that has fewer than 300 lines, the inserted content is indented. To keep the original indent style of the document you copy content from, deselect this option.

Whitespaces Preferences

When Oxygen XML Editor formats and indents XML documents, a whitespace normalization process is applied, thus replacing whitespace sequences with single space characters. Oxygen XML Editor allows you to configure which Unicode characters are treated as spaces during the normalization process.

To configure the Whitespace preferences, open the Preferences dialog box (Options > Preferences) (on page 124) and go to Editor > Format > XML > Whitespaces.

This table lists the Unicode whitespace characters. Select any that you want to have treated as whitespace when formatting and indenting an XML document.

The whitespaces are normalized when:

- The Format and Indent action is applied on an XML document.
- You switch from Text mode to Author mode.
- You switch from Author mode to Text mode.

Note:

The whitespace normalization process replaces any sequence of characters declared as whitespaces in the Whitespaces table with a single space character (U+0020). If you want to be sure that a certain whitespace character will not be removed in the normalization process, deselect it in the Whitespaces table.

Important:

The characters with the codes U+0009 (TAB), U+000A (LF), U+000D (CR) and U+0020 (SPACE) are always considered to be whitespace characters and cannot be deselected.

XQuery Preferences

To configure the XQuery Formatting options, open the Preferences dialog box (Options > Preferences) (on page 124) and go to Editor > Format > XQuery.

The following options are available:

- **Preserve line breaks** - All initial line breaks are preserved.
- **Break line before an attribute name** - Each attribute of an XML element is written on a new line and properly indented.
XPath Preferences

To configure the XPath Formatting options, open the Preferences dialog box (Options > Preferences) (on page 124) and go to Editor > Format > XPath.

The following option is available:

Format XPath code embedded in XSLT, XSD and Schematron files

If selected, the Format and Indent action applied on an XSD, XSLT, or Schematron document will perform an XPath-specific formatting on the values of the attributes that accept XPath expressions.

Note:
For XSLT documents, the formatting is not applied to attribute value templates.

CSS Preferences

Oxygen XML Editor can format and indent your CSS files. To configure the CSS formatting options, open the Preferences dialog box (Options > Preferences) (on page 124) and go to Editor > Format > CSS.

The following options control how your CSS files are formatted and indented:

Class body on new line

If selected, the class body (including the curly brackets) is placed on a new line. This option is not selected by default.

Indent class content

When selected (default state), the class content is indented.

Add space before the value of a CSS property

When selected (default state), whitespaces are added between the : (colon) and the value of a style property.

Add new line between classes

If selected, an empty line is added between two classes. This option is not selected by default.

Preserve empty lines

When selected (default state), the empty lines from the CSS content are preserved.

Allow formatting embedded CSS

When selected (default state), CSS content that is embedded in XML is also formatted when the XML content is formatted.

JavaScript Preferences

To configure the JavaScript format options, open the Preferences dialog box (Options > Preferences) (on page 124) and go to Editor > Format > JavaScript.
The following options control the behavior of the **Format and Indent** action:

- **Start curly brace on new line** - Opening curly braces start on a new line.
- **Preserve empty lines** - Empty lines in the JavaScript code are preserved. This option is selected by default.
- **Allow formatting embedded JavaScript** - Applied only to XHTML documents, this option allows Oxygen XML Editor to format embedded JavaScript code, taking precedence over the **Schema-aware format and indent** (on page 208) option. This option is selected by default.

**Content Completion Preferences**

Oxygen XML Editor provides a **Content Completion Assistant** (on page 3217) that provides a list of available options at any point in a document and can auto-complete structures, elements, and attributes. To configure the **Content Completion** preferences, open the **Preferences** dialog box (Options > Preferences) (on page 124) and go to Editor > Content Completion. These options control how the **Content Completion Assistant** works.

The following options are available:

- **Auto close the last opened tag**
  
  When selected, Oxygen XML Editor automatically closes the last open tag when you type `</`.

- **Automatically rename/delete/comment matching tags**
  
  If you rename, delete, or comment out a start tag, Oxygen XML Editor automatically renames, deletes, or comments out the matching end tag.

  **Note:**
  
  If you select **Toggle comment** for multiple starting tags and the matching end tags are on the same line as other start tags, the end tags are not commented.

- **Enable content completion**
  
  Toggles the content completion feature on or off.

- **Consider subsequent sibling elements**
  
  When this option is selected (default), the subsequent sibling elements of the current element are taken into account when using the **Content Completion Assistant**. For example, in DITA, if you invoke the content completion before an already inserted required element (e.g. a `<title>` element), the content completion mechanism will not offer a proposal to insert a title (since it was already inserted).

- **Close the inserted element**
  
  When you choose an entry from the **Content Completion Assistant** list of proposals, Oxygen XML Editor inserts both start and end tags. The following additional options are available with regard to closing the element:
• **If it has no matching tag** - The end tag of the inserted element is automatically added only if it is not already present in the document.

• **Add element content** - Oxygen XML Editor inserts the required elements specified in the DTD, XML Schema, or RELAX NG schema that is associated with the edited XML document *(on page 816).*

  - **Add optional content** - If selected, Oxygen XML Editor inserts the optional elements specified in the DTD, XML Schema, or RELAX NG schema.
  - **Add first Choice particle** - If selected, Oxygen XML Editor inserts the first choice particle specified in the DTD, XML Schema, or RELAX NG schema.

**Case sensitive search**

When selected, the search in the Content Completion Assistant is case-sensitive when you type a character (‘a’ and ‘A’ are different characters).

> **Note:**
> 
> This option is ignored when the current language itself is not case-sensitive. For example, the case is ignored in the CSS language.

**Position cursor between tags**

When selected, Oxygen XML Editor automatically moves the cursor between the start and end tag after inserting the element. This only applies to:

• Elements with only optional attributes or no attributes at all.
• Elements with required attributes, but only when the Insert the required attributes option *(on page 212)* is not selected.

**Show all entities**

Oxygen XML Editor displays a list with all the internal and external entities declared in the current document when you type the start character of an entity reference (for example, `&`).

**Insert the required attributes**

Oxygen XML Editor inserts automatically the required attributes taken from the DTD or XML Schema.

**Insert the fixed attributes**

If selected, Oxygen XML Editor automatically inserts any fixed attributes from the DTD or XML Schema for an element inserted with the help of the Content Completion Assistant.

**Show recently used items**

When selected, Oxygen XML Editor remembers the last inserted items from the Content Completion Assistant window. The number of items to be remembered is limited by the Maximum number of recent items shown list box. These most frequently used items are
displayed on the top of the content completion window and are separated from the rest of the suggestions by a thin gray line.

**Maximum number of recent items shown**

Specifies the limit for the number of recently used items presented at the top of the *Content Completion Assistant window.*

**Learn attributes values**

When selected, Oxygen XML Editor learns the attribute values used in a document.

**Learn on open document**

Oxygen XML Editor automatically learns the document structure when the document is opened.

**Learn words (Dynamic Abbreviations, available on Ctrl+Space (Command+Space on macOS))**

When selected, Oxygen XML Editor learns the typed words and makes them available in a content completion fashion by pressing `Ctrl + Space (Command + Space on macOS)` on your keyboard;

![Note:]

For the words to be learned, they need to be separated by space characters.

**Activation delay of the proposals window (ms)**

Delay in milliseconds from the last key press until the *Content Completion Assistant* is displayed.

**Related information**

[Configuring the Proposals for Attribute and Element Values](on page 2200)

**XSLT Preferences**

XSLT stylesheets are often used to create output in XHTML or XSL-FO. In addition to suggesting content completion options for XSLT stylesheet elements, Oxygen XML Editor can suggest elements from these vocabularies. To configure the XSLT content completion options, open the Preferences dialog box ([Options > Preferences](on page 124)) and go to Editor > Content Completion > XSLT.

The following options are available:

**Include elements declared in the schema section**

This section includes options with regard to detecting elements from the declared schema.

**Automatically detect HTML or Formatting Objects**

Detects if the output being generated is HTML or FO and provides content completion for those vocabularies. Oxygen XML Editor analyzes the namespaces declared in the root element to find an appropriate schema.

If the detection fails, Oxygen XML Editor uses one of the following options:
• **None** - The *Content Completion Assistant (on page 3217)* suggests only XSLT elements.
• **HTML** - The *Content Completion Assistant (on page 3217)* includes HTML elements, including HTML5 elements (such as `<video>`, `<canvas>`, etc.).
• **Formatting objects** - The *Content Completion Assistant (on page 3217)* includes Formatting Objects (XSL-FO) elements as substitutes for `<xsl:element>`.
• **Custom schema** - If you want content completion hints for another output vocabulary, you can use this option to specify the path to the schema for that vocabulary. The supported schema types are DTD, XML Schema, RNG schema, or NVDL schema for inserting elements from the target language of the stylesheet.

**Documentation schema section**

This section specifies an additional schema that will be used for documenting XSL stylesheets. You can choose between the following:

• **Built-in schema** - Uses the built-in schema for documentation.
• **Custom schema** - Allows you to specify a custom schema for documentation. The supported schema types are XSD, RNG, RNC, DTD, and NVDL.

**XPath Preferences**

Oxygen XML Editor provides content-completion support for XPath expressions. To configure the options for the content completion in XPath expressions, open the *Preferences dialog box (Options > Preferences) (on page 124)* and go to *Editor > Content Completion > XPath*.

The following options are available:

• **Enable content completion for XPath expressions** - Enables the *Content Completion Assistant in XPath expressions (on page 894)* that you enter in the `@match`, `@select`, and `@test` XSL attributes and also in the XPath toolbar *(on page 2000)*.
  ◦ **Include XPath functions** - When this option is selected, XPath functions are included in the content completion suggestions.
  ◦ **Include XSLT functions** - When this option is selected, XSLT functions are included in the content completion suggestions.
  ◦ **Include axes** - When this option is selected, XSLT axes are included in the content completion suggestions.
• **Show signatures of XSLT / XPath functions** - Makes the editor indicate the signature of the XPath function located at the cursor position in a tooltip. See the *XPath Tooltip Helper (on page 898)* section for more information.
• **Function signature window background** - Specifies the background color of the tooltip window.
• **Function signature window foreground** - Specifies the foreground color of the tooltip window.
XSD Preferences

Oxygen XML Editor provides content completion assistance when you are writing XML Schema (XSD). To configure XSD preferences, open the Preferences dialog box (Options > Preferences) (on page 124) and go to Editor > Content Completion > XSD. The option in this preferences page allows you to define additional elements to be suggested by the Content Completion Assistant (on page 3217) in `<xs:appinfo>` elements (in addition to the elements defined in the XML Schema).

The following option is available:

**When in "xs:appinfo" context, also include elements declared in the schema**

You can choose between the following:

- **None** - The Content Completion Assistant offers only the XML Schema schema information.
- **ISO Schematron** - The Content Completion Assistant also includes ISO Schematron elements in `<xs:appinfo>`.
- **Schematron 1.5** - The Content Completion Assistant also includes Schematron 1.5 elements in `<xs:appinfo>`.
- **Other** - The Content Completion Assistant also includes elements from an XML Schema identified by a URL in `<xs:appinfo>` elements.

JavaScript Preferences

Oxygen XML Editor can provide content completion suggestions when you are writing JavaScript files. To configure content completion support for JavaScript, open the Preferences dialog box (Options > Preferences) (on page 124) and go to Editor > Content Completion > JavaScript. You can configure the following options:

**Enable content completion**

Enables the content completion support for JavaScript files.

**Use built-in libraries**

Allows Oxygen XML Editor to include components (object names, properties, functions, and variables) collected from the built-in JavaScript library files when making suggestions.

**Use defined libraries**

Oxygen XML Editor can also use JavaScript libraries to when making suggestions. List the paths (URIs) of any JavaScript files you want Oxygen XML Editor to use when making suggestions.

**Note:**

The paths can contain editor variables (on page 323) such as `${pdu}`, or `${oxygenHome}`. You can make these paths relative to the project directory or installation directory.
JSON Preferences

Oxygen XML Editor can provide content completion suggestions when you are editing JSON files. To configure content completion support for JSON, open the Preferences dialog box (Options > Preferences) (on page 124) and go to Editor > Content Completion > JSON. You can configure the following options:

**Generate required content**
- When invoking content completion over JSON files, all contextual required content is automatically generated according to the specifications from the associated JSON Schema.

**Generate optional properties**
- If selected, optional properties that are defined in the associated JSON Schema will be added when using content completion in JSON files.

**Generate additional content**
- If selected, additional properties (or additional items for arrays) that are defined in the associated JSON Schema will be added when using content completion in JSON files.

Annotations Preferences

Certain types of schemas (XML Schema, DTDs, Relax NG) can include annotations that document the various elements and attributes that they define. Oxygen XML Editor can display these annotations when offering content completion suggestions. To configure the Annotations preferences, open the Preferences dialog box (Options > Preferences) (on page 124) and go to Editor > Content Completion > Annotations.

The following options are available:

**Show annotations in Content Completion Assistant**
- If selected, Oxygen XML Editor displays the schema annotations of an element, attribute, or attribute value currently selected in the Content Completion Assistant (on page 3217) proposals list.

**Show annotations in tooltip**
- If selected, Oxygen XML Editor displays the annotation of elements and attributes as a tooltip when you hover over them with the cursor in the editing area or in the Elements view (on page 546). If not selected, tooltips are disabled in all modes.

**Show annotation in HTML format, if possible**
- This option allows you to view the annotations associated with an element or attribute in HTML format. It is available when editing XML documents that have associated an XML Schema or Relax NG schema. If this option is not selected, the annotations are converted and displayed as plain text.

**Prefer DTD comments that start with "doc:" as annotations**
To address the lack of dedicated annotation support in DTD documents, Oxygen XML Editor recommends prefixing with the `doc:` particle all comments intended to be shown to the developer who writes an XML validated against a DTD schema.

If this option is selected, Oxygen XML Editor uses the following mechanism to collect annotations:

- If at least one `doc:` comment is found in the entire DTD, only comments of this type are displayed as annotations.
- If no `doc:` comment is found in the entire DTD, all comments are considered annotations and displayed as such.

If not selected, all comments, regardless of their type, are considered annotations and displayed as such.

**Use all Relax NG annotations as documentation**

If selected, any element outside the Relax NG namespace, that is [http://relaxng.org/ns/structure/1.0](http://relaxng.org/ns/structure/1.0), is considered annotation and is displayed in the annotation window next to the [Content Completion Assistant](on page 3217) window and in the [Model view](on page 545). When this option is not selected, only elements from the Relax NG annotations namespace, that is [http://relaxng.org/ns/compatibility/annotations/1.0](http://relaxng.org/ns/compatibility/annotations/1.0) are considered annotations.

**Related information**

[Schema Annotations in Text Mode](on page 535)

**Code Templates Preferences**

**Code templates** ([on page 536]) are code fragments that can be inserted at the current editing position. Oxygen XML Editor includes a set of built-in templates for CSS, LESS, Schematron, XSL, XQuery, JSON, HTML, and XML Schema document types. You can also define your own code templates ([on page 537]) for any type of file and share them with your colleagues ([on page 538]) using the template export and import functions.

To configure Code Templates, open the Preferences dialog box ([Options > Preferences](on page 124)) and go to Editor > Templates > Code Templates.

This preferences page contains a list of all the available code templates (both built-in and custom created ones) and a code preview area. You can disable any code template by deselecting it.

The following actions are available:

- **New**
  
  Opens the Code template dialog box that allows you to define a new code template. You can define the following fields:
• **Name** - The name of the code template.

• **Description** - [Optional] The description of the code template that will appear in the Code Templates preferences page and in the tooltip message when selecting it from the Content Completion Assistant (on page 3217). HTML markup can be used for better rendering.

• **Associate with** - You can choose to set the code template to be associated with a specific type of editor or for all editor types.

• **Shortcut key** - [Optional] If you want to assign a shortcut key that can be used to insert the code template, place the cursor in the Shortcut key field and press the desired key combination on your keyboard. Use the Clear button if you make a mistake. If the Enable platform-independent shortcut keys checkbox is selected, the shortcut is platform-independent and the following modifiers are used:
  - M1 represents the Command key on macOS, and the Ctrl key on other platforms.
  - M2 represents the Shift key.
  - M3 represents the Option key on macOS, and the Alt key on other platforms.
  - M4 represents the Ctrl key on macOS, and is undefined on other platforms.

• **Content** - Text box where you define the content that is used when the code template is inserted. An editor variable (on page 323) can be inserted in the text box using the Insert Editor Variables button.

**Edit**

Opens the Code template dialog box and allows you to edit an existing code template. You can edit the following fields:

• **Description** - [Optional] The description of the code template that will appear in the Code Templates preferences page and in the tooltip message when selecting it from the Content Completion Assistant (on page 3217). HTML markup can be used for better rendering.

• **Shortcut key** - [Optional] If you want to assign a shortcut key that can be used to insert the code template, place the cursor in the Shortcut key field and press the desired key combination on your keyboard. Use the Clear button if you make a mistake. If the Enable platform-independent shortcut keys checkbox is selected, the shortcut is platform-independent and the following modifiers are used:
  - M1 represents the Command key on macOS, and the Ctrl key on other platforms.
  - M2 represents the Shift key.
  - M3 represents the Option key on macOS, and the Alt key on other platforms.
  - M4 represents the Ctrl key on macOS, and is undefined on other platforms.

• **Content** - Text box where you define the content that is used when the code template is inserted. An editor variable (on page 323) can be inserted in the text box using the Insert Editor Variables button.

**Duplicate**
Creates a duplicate of the currently selected code template.

**Delete**

Deletes the currently selected code template. This action is not available for the built-in code templates.

**Export**

Exports a file with code templates.

**Import**

Imports a file with code templates that was created by the Export action.

You can use the following editor variables (on page 323) when you define a code template in the Content text box:

- **${caret}** - The position where the cursor is located. This variable can be used in a code template, in Author mode operations, or in a selection plugin.

  **Note:**
  
  The ${caret} editor variable is available only for parameters that take XML content as values. It is replaced with the ${UNIQUE_CARET_MARKER_FOR_AUTHOR} macro. The default Author operations process this macro and position the cursor at the designated offset.

  **Note:**
  
  The ${caret} editor variable can be used for setting a fixed cursor position inside an XML fragment. To set the cursor position depending on the fragment inserted in the document, you can use AuthorDocumentFilter and inside the insertFragment(AuthorDocumentFilterBypass, int, AuthorDocumentFragment) method, use the AuthorDocumentFragment.setSuggestedRelativeCaretOffset(int) API on the given fragment.

- **${selection}** - The currently selected text content in the currently edited document. This variable can be used in a code template, in Author mode operations, or in a selection plugin.

- **${ask('message', type, ('real_value1':rendered_value1'; 'real_value2':rendered_value2'; ...), 'default_value', @id)}** - To prompt for values at runtime, use the ask('message', type, ('real_value1':rendered_value1'; 'real_value2':rendered_value2'; ...), 'default_value') editor variable. You can set the following parameters:
  
  - 'message' - The displayed message. Note the quotes that enclose the message.
  - 'default-value' - Optional parameter. Provides a default value.
  - '@id' - Optional parameter. Used for identifying the variable to reuse the answer using the ${answer(@id)} editor variable.
  - 'type' - Optional parameter (defaults to generic), with one of the following values:
Note:
The title of the dialog box will be determined by the type of parameter and as follows:

- For `url` and `relative_url` parameters, the title will be the name of the parameter and the value of the 'message'.
- For the other parameters listed below, the title will be the name of that respective parameter.
- If no parameter is used, the title will be "Input".

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Format:</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>generic</code> (default)</td>
<td><code>{ask('message', generic, 'default')}</code></td>
<td>The input is considered to be generic text that requires no special handling.</td>
</tr>
<tr>
<td><code>url</code></td>
<td><code>{ask('message', url, 'default_value')}</code></td>
<td>Input is considered a URL. Oxygen XML Editor checks that the provided URL is valid.</td>
</tr>
<tr>
<td><code>relative_url</code></td>
<td><code>{ask('message', relative_url, 'default')}</code></td>
<td>Input is considered a URL. Oxygen XML Editor tries to make the URL relative to that of the document you are editing.</td>
</tr>
</tbody>
</table>

Example:

- `${ask('Hello world!')}` - The dialog box has a *Hello world!* message displayed.
- `${ask('Hello world!', generic, 'Hello again!')}` - The dialog box has a *Hello world!* message displayed and the value displayed in the input box is 'Hello again!'.
- `${ask('Input URL', url)}` - The displayed dialog box has the name *Input URL*. The expected input type is URL.
- `${ask('Input URL', url, 'http://www.example.com')}` - The displayed dialog box has the name *Input URL*. The input field displays the default value *http://www.example.com*.

Note:
If the `$ask` editor variable is expanded in content that is not yet saved (such as an *untitled* file, whose path cannot be determined), then Oxygen XML Editor will transform it into an absolute URL.

Example:
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>${ask('File location', relative_url, 'C:/example.txt')}</td>
<td>- The dialog box has the name 'File location'. The URL inserted in the input box is made relative to the currently edited document location.</td>
</tr>
<tr>
<td>password</td>
<td><strong>Format:</strong> ${ask('message', password, 'default')}</td>
</tr>
<tr>
<td></td>
<td><strong>Description:</strong> The input is hidden with bullet characters.</td>
</tr>
<tr>
<td></td>
<td><strong>Example:</strong></td>
</tr>
<tr>
<td></td>
<td>▪ ${ask('Input password', password)} - The displayed dialog box has the name 'Input password' and the input is hidden with bullet symbols.</td>
</tr>
<tr>
<td></td>
<td>▪ ${ask('Input password', password, 'abcd')} - The displayed dialog box has the name 'Input password' and the input hidden with bullet symbols. The input field already contains the default <strong>abcd</strong> value.</td>
</tr>
<tr>
<td>combobox</td>
<td><strong>Format:</strong> ${ask('message', combobox, ('real_value1':'rendered_value1';..'real_valueN':'rendered_valueN'), 'default')}</td>
</tr>
<tr>
<td></td>
<td><strong>Description:</strong> Displays a dialog box that offers a drop-down menu. The drop-down menu is populated with the given <strong>rendered_value</strong> values. Choosing such a value will return its associated value (<strong>real_value</strong>).</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> The list of <strong>real_value</strong>:<strong>rendered_value</strong> pairs can be computed using <strong>${xpath_eval()}</strong>.</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> The <strong>default</strong> parameter specifies the default-selected value and can match either a key or a value.</td>
</tr>
<tr>
<td></td>
<td><strong>Example:</strong></td>
</tr>
<tr>
<td></td>
<td>▪ ${ask('Operating System', combobox, ('win':'Microsoft Windows';'macos':'macOS';'lnx':'Linux/UNIX'), 'macos')} - The dialog box has the name 'Operating System'. The drop-down menu displays the three given operating systems. The associated value will be returned based upon your selection.</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> In this example, the default value is indicated by the <strong>osx</strong> key. However, the same result could be obtained if the default value is indicated by <strong>macOS</strong>, as in the following ex-</td>
</tr>
</tbody>
</table>


### Parameter

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
</table>
| editable_combobox    | **Format:** ${ask('message', editable_combobox, ('real_value1':'rendered_value1';...;'real_valueN':'rendered_valueN'), 'default')}

**Description:** Displays a dialog box that offers a drop-down menu with editable elements. The drop-down menu is populated with the given rendered_value values. Choosing such a value will return its associated real value (real_value) or the value inserted when you edit a list entry.

**Note:**
- The list of 'real_value':'rendered_value' pairs can be computed using ${xpath_eval()}
- The 'default' parameter specifies the default-selected value and can match either a key or a value.

**Example:**
- ${ask('Operating System', editable_combobox, ('win':'Microsoft Windows';'macos':'macOS';'lnx':'Linux/UNIX'), 'macos')} - The dialog box has the name 'Operating System'. The drop-down menu displays the three given operating systems and also allows you to edit the entry. The associated value will be returned based upon your selection or the text you input.
- ${ask('Operating System', editable_combobox, ${xpath_eval(for $pair in ('win', 'Microsoft Windows'), ('macos', 'macOS'), ('lnx', 'Linux/UNIX')) return $pair?1 || $pair?2 || 'ios'))}
<table>
<thead>
<tr>
<th>Parameter</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Note:</strong> The list of 'real_value': 'rendered_value' pairs can be computed using (${xpath_eval()}).</td>
</tr>
<tr>
<td><strong>Note:</strong> The 'default' parameter specifies the default-selected value and can match either a key or a value.</td>
</tr>
</tbody>
</table>

**Example:**

- \(\$\{ask('Operating System', radio, ('win': 'Microsoft Windows'; 'macos': 'macOS'; 'lnx': 'Linux/UNIX'), 'macos')\} - The dialog box has the name 'Operating System'. The radio button group allows you to choose between the three operating systems.

  **Note:** In this example, macOS is the default-selected value and if selected, it would return macOS for the output.

- \(\$\{ask('Operating System', radio, \(\$\{xpath\_eval(\{for$\}$pair in ([\['win', 'Microsoft Windows']\], [\['macos', 'macOS']\], [\['lnx', 'Linux/UNIX']\]) return "'" || $pair?1 || "'':'" || $pair?2 || "';" || $pair?1 || "'":"" || $pair?2 || "';")\}, 'ios')\} |

- **\(\$\{timeStamp\} - The timestamp, which is the current time in Unix format. For example, it can be used to save transformation results in multiple output files on each transformation.**
- **\(\$\{uuid\} - Universally unique identifier, a unique sequence of 32 hexadecimal digits generated by the Java UUID class.**
- **\(\$\{id\} - Application-level unique identifier. It is a short sequence of 10-12 letters and digits that is not guaranteed to be universally unique.**
- **\(\$\{cfn\} - Current file name without the extension and parent folder. The current file is the one currently open and selected.**
- **\(\$\{cfne\} - Current file name with extension. The current file is the one currently open and selected.**
- **\(\$\{cf\} - Current file as file path, that is the absolute file path of the currently edited document.**
- **\(\$\{cfd\} - Current file folder as file path, that is the path of the currently edited document up to the name of the parent folder.**
- **\(\$\{frameworks\_Dir\} - The path (as file path) of the frameworks directory. When used to define references inside a framework configuration, it expands to the parent folder of that specific framework folder. Otherwise, it expands to the main frameworks folder defined in the Document Type Association > Locations preferences page.**
- **\(\$\{pd\} - The file path to the folder that contains the current project file (.xpr).**
• ${oxygenInstallDir} - Oxygen XML Editor installation folder as file path.
• ${homeDir} - The path (as file path) of the user home folder.
• ${pn} - Current project name.
• ${env(VAR_NAME)} - Value of the VAR_NAME environment variable. The environment variables are managed by the operating system. If you are looking for Java System Properties, use the ${system(var.name)} editor variable.
• ${system(var.name)} - Value of the var.name Java System Property. The Java system properties can be specified in the command-line arguments of the Java runtime as -Dvar.name=var.value. If you are looking for operating system environment variables, use the ${env(VAR_NAME)} editor variable instead.
• ${date(pattern)} - Current date. The allowed patterns are equivalent to the ones in the Java SimpleDateFormat class. Example: yyyy-MM-dd.

Note:

This editor variable supports both the xs:date and xs:datetime parameters. For details about xs:date, go to: http://www.w3.org/TR/xmlschema-2/#date. For details about xs:datetime, go to: http://www.w3.org/TR/xmlschema-2/#dateTime.

Related information
Code Templates (on page 536)

Syntax Highlight Preferences

Oxygen XML Editor supports syntax highlighting in the Text mode editors for numerous types of documents, including XML, XHTML, JavaScript, XQuery, XPath, PHP, CSS, LESS, Markdown, Text, DTD, RNC, Java, JSON, Ant, and more.

To configure syntax highlighting, open the Preferences dialog box (Options > Preferences) (on page 124) and go to Editor > Syntax Highlight.

To set syntax colors for a language, expand the listing for that language in the top panel to show the list of syntax items for that type of document. Use the color and style selectors to change how each syntax item is displayed. The results of your changes are displayed in the Preview panel. If you do not know the name of the syntax token that you want to configure, click that token in the Preview area to select it.

Note:

All default color sets come with a high-contrast variant that is automatically used when you switch to a black-background or white-background high-contrast theme in your Windows operating system settings. The high-contrast theme will not overwrite any default color you set in Editor > Syntax Highlight preferences page.

The settings for XML documents are also used in XSD, XSL, RNG documents and the Preview area has a separate tab for each of them when XML is selected in the top pane.
The **Enable nested syntax highlight** option controls whether or not content types that are nested in the same file (such as PHP, JS, or CSS scripts inside an HTML file) are highlighted according to the color schemes defined for each content type.

**Elements/Attributes by Prefix Preferences**

Oxygen XML Editor allows you to specify syntax highlighting colors for XML elements and attributes with specific namespace prefixes. To configure the **Elements/Attributes by Prefix** preferences, open the **Preferences** dialog box (Options > Preferences) (on page 124) and go to Editor > Syntax Highlight > Elements/Attributes by Prefix.

To change the syntax coloring for a specific namespace prefix, choose the prefix from the list, or add a new one using the **New** button, and use the color and style selectors to set the syntax highlighting style for that namespace prefix.

**Note:**

Syntax highlighting is based on the literal namespace prefix, not the namespace that the prefix is bound to in the document.

If you only want the prefix (and not the whole element or attribute name) to be styled with a particular color, select the **Draw only the prefix with a separate color** option.

**Mark Occurrences Preferences**

This preferences page specifies which types of files will have the **Highlight IDs Occurrences** (on page 566) feature activated. To configure these options, open the **Preferences** dialog box (Options > Preferences) (on page 124) and go to Editor > Mark Occurrences:

The following options are available in this preferences page:

**Highlight component occurrences in the current file for:**

- **XML files** - Activates the **Highlight IDs Occurrences** (on page 566) feature in XML files.
- **XSLT files** - Activates the **Highlight Component Occurrences** (on page 911) feature in XSLT files.
- **XML Schema files** - Activates the **Highlight Component Occurrences** (on page 1002) feature in XSD files.
- **WSDL files** - Activates the **Highlight Component Occurrences** (on page 1002) feature in WSDL files.
- **RNG files** - Activates the highlight component occurrences feature in RNG files.
- **Schematron files** - Activates the **Highlight Component Occurrences** (on page 1217) feature in Schematron files.
- **Ant files** - Activates the **Highlight Component Occurrences** (on page 944) feature in Ant files.
Declaration highlight color

Allows you to choose the color to be used for highlighting component declarations.

Reference highlight color

Allows you to choose the color to be used for highlighting component references.

Document Validation Preferences

To configure document validation options, open the Preferences dialog box (Options > Preferences) (on page 124) and go to Editor > Document Validation. This page contains preferences for configuring how a document is checked for both well-formedness and validation errors.

The following options are available:

Maximum number of validation highlights

If a validation generates more errors than the number specified in this option, only the errors up to this number are highlighted in the editor panel and on the stripe that is displayed at the right side of the editor panel. This option applies to both automatic validation (on page 775) and manual validation (on page 775).

Validation fatal error highlight color

The color used to highlight fatal validation errors in the document.

Validation error highlight color

The color used to highlight validation errors in the document.

Validation warning highlight color

The color used to highlight validation warnings in the document.

Validation info highlight color

The color used to highlight validation info messages in the document.

Validation success color

The color used to highlight the success indicator of the validation operation in the vertical ruler bar.

Always show validation status

If this option is selected, the current validation error or warning is always visible in the message line at the bottom of the editor panel. This is useful when the Enable automatic validation option is selected and the vertical scroll bar changes position due to an error message being displayed.

Enable automatic validation

This causes the validation to be automatically executed in the background as the document is modified in Oxygen XML Editor.

Delay after the last key event (s)

The period of keyboard inactivity before starting a new validation (in seconds).
At the bottom of the preferences page you can choose whether or not the saved options will be shared with other users by selecting **Global** or **Project** storage options *(on page 311).*

### Custom Validation Engines Preferences

As the name implies, the **Custom Validation Engines** preferences page displays the list of custom validation engines that can be associated to a particular editor and used for validating documents. To access this page, open the **Preferences** dialog box *(Options > Preferences)* *(on page 124)* and go to **Editor > Document Validation > Custom Validation Engines.**

If you want to add a new custom validation tool or edit the properties of an existing one, you can use the **Custom Validator** dialog box displayed by pressing the **New** or **Edit** button.

![Custom Validator Dialog Box](image)

The **Custom Validator** dialog box allows you to configure the following parameters:

- **Name**
  
  Name of the custom validation engine that will be displayed in the **Validation** toolbar drop-down menu.

- **Executable path**
Path to the executable file of the custom validation tool. You can specify the path by using the text field, the Insert Editor Variables (on page 323) button, or the Browse button.

Working directory

The working directory of the custom validation tool. You can specify the path by using the text field, the Insert Editor Variables (on page 323) button, or the Browse button.

Associated editors

The editors that can perform validation with the external tool (XML editor, XSL editor, XSD editor, etc.)

Command-line arguments for detected schemas

Command-line arguments used in the commands that validate the currently edited file against various types of schema (XML Schema, Relax NG full syntax, Relax NG compact syntax, NVDL, Schematron, DTD, etc.) The arguments can include any custom switch (such as -rng) and the following editor variables (on page 323):

- **${cf}** - Current file as file path, that is the absolute file path of the currently edited document.
- **${currentFileURL}** - Current file as URL, that is the absolute file path of the currently edited document represented as URL.
- **${ds}** - The path of the detected schema as a local file path for the current validated XML document.
- **${dsu}** - The path of the detected schema as a URL for the current validated XML document.

Increasing the Stack Size for Validation Engines

To prevent the appearance of a StackOverflowException error, use one of the following methods:

- Use the com.oxygenxml.stack.size.validation.threads property to increase the size of the stack for validation engines. The value of this property is specified in bytes. For example, to set a value of one megabyte specify 1x1024x1024=1048576. For information about how to setup the system property on the JVM, see Setting a Java Virtual Machine Parameter when Launching Oxygen XML Editor (on page 339).
- Increase the value of the -Xss parameter.

Note:

Increasing the value of the -Xss parameter affects all the threads of the application.
Ignored Validation Problems Preferences

Some validation issues include a Quick Fix (on page 813) proposal that instructs the application to ignore that type of validation problem. The ignored problems are then listed in the Ignored Validation Problems preferences page. To access this page, open the Preferences dialog box (Options > Preferences) (on page 124) and go to Editor > Document Validation > Ignored Validation Problems.

The Ignored Validation Problems preferences page includes the following:

Enable support for ignoring validation problems

If this option is selected, the support for ignoring certain validation problems is enabled. This means that when you choose a Quick Fix proposal to ignore the particular validation problem (on page 812), it is added to the table below this option.

Ignored problems table

The table includes an entry for each validation problem that has been ignored. The columns in the table include information about the Severity, Problem ID, Message, and System ID. The entries are added automatically when you choose a Quick Fix proposal to ignore the particular validation problem (on page 812). You can delete an entry by selecting it and clicking the Delete button at the bottom of the table. The deleted problem is no longer ignored.

Tip:

Changes made in this preferences page can be saved at project level (on page 312) so that you can easily share your ignored problems configuration with others.

Spell Check Preferences

Oxygen XML Editor provides support for spell checking in the Text (on page 353) and Author (on page 354) editing modes. To configure the Spell Check options, open the Preferences dialog box (Options > Preferences) (on page 124) and go to Editor > Spell Check.

The following options are available:

Automatic spell check

This option is not selected by default. When selected, Oxygen XML Editor automatically checks the spelling as you type and highlights misspelled words in the document.

Select editors
You can select which editors (and therefore which file types) will automatically be spell checked. File types such as CSS and DTD are excluded by default since automatic spell checking is not usually helpful in these types of files.

**Spell check highlight color**

Use this option to set the color used by the spell check engine to highlight spelling errors.

**Language options section**

This section includes the following language options:

**Default language**

The default language list allows you to choose the language used by the spell check engine when the language is not specified in the source file. You can add additional dictionaries to the spell check engines (on page 230).

**Use "lang" and "xml:lang" attributes**

When this option is selected, the contents of an element with one of the @lang or @xml:lang attributes is checked in that language. Choose between the following two options for instances when these attributes are missing:

- **Use the default language** - If the @lang and @xml:lang attributes are missing, the selection in the Default language list (on page 230) is used.
- **Do not check** - If the @lang and @xml:lang attributes are missing, the element is not checked.

**XML spell checking in section**

You can choose to check the spelling inside the following XML items:

- Comments
- Processing instructions
- Attribute values
- Text
- CDATA

**Options section**

This section includes the following other options:

**Check capitalization**

When selected, the spell checker reports detected capitalization errors.
Check punctuation

When selected, the spell checker checks punctuation. Misplaced white space and unusual sequences, such as a period following a comma, are highlighted as errors.

Ignore mixed case words

When selected, the spell checker does not check words containing mixed case characters (for example, SpellChecker).

Ignore acronyms

Available only for the Hunspell Spell Checker. When selected, acronyms are not reported as errors.

Ignore words with digits

When selected, the spell checker does not check words containing digits (for example, b2b).

Ignore duplicates

When selected, the spell checker does not signal two successive identical words as an error.

Ignore URL

When selected, the spell checker ignores words recognized as URLs or file names (for example, www.oxygenxml.com or c:\boot.ini).

Allow compounds words

When selected, all words formed by concatenating multiple legal words with hyphens or underscores are accepted.

Allow file extensions

When selected, the spell checker accepts any word ending with recognized file extensions (for example, myfile.txt or index.html).

Ignore elements section

You can use the Add and Remove buttons to configure a list of element names or XPath expressions to be ignored by the spell checker. The following restricted set of XPath expressions are supported:

- ‘/’ and ‘//’ separators
- ‘*’ wildcard
An example of an allowed XPath expression is: `/a/*/b`.

**AutoCorrect options link**

Use this link to navigate to the [AutoCorrect preferences page](on page 193).

## Spell Check Dictionaries Preferences

To set the Dictionaries preferences, open the Preferences dialog box ([Options > Preferences](on page 124)) and go to **Editor > Spell Check > Dictionaries**. This page allows you to configure the dictionaries (.dic files) and term lists (.tdi files) that Oxygen XML Editor uses and to choose where to save new learned words.

The following options are valid when Oxygen XML Editor uses the Hunspell spell checking engine:

### Dictionaries and term lists default folder

Displays the default location where the dictionaries and term lists that Oxygen XML Editor uses are stored.

### Include dictionaries and term list from

Selecting this option allows you to specify a location where you have stored dictionaries and term lists that you want to include, along with the default ones.

---

**Important:**

Consider the following notes regarding this option:

- The spell checker takes into account dictionaries and term lists collected both from the default and custom locations and multiple dictionaries and term lists from the same language are merged (for example, `en_UK.dic` from the default location is merged with `en_US.dic` from a custom location).
- If you have a generic dictionary file (one that just has a two-letter language code for its file name, such as `en.dic`) saved in either the default or custom location, the other more specific dictionaries (for example, `en_UK.dic` and `en_US.dic`) will not be merged and the existing generic dictionary will simply be used instead.
- If the additional location contains a file with the same name as one from the default location, the file in the additional location takes precedence over the file from the default location.

---

**How to add more dictionaries and term lists link**

Use this link to open a topic in the Oxygen XML Editor User Guide that explains how to add more dictionaries and term lists ([on page 451]).

**Save learned words in the following location**

Specifies the target where the newly learned words are saved. By default, the target is the application preferences folder, but you can also choose a custom location.

**Delete learned words**
Opens the list of learned words, allowing you to select the items you want to remove, without deleting the dictionaries and term lists.

**Note:**

Words stored in dictionaries, term lists, and the list of learned words are not handled as case-sensitive. Therefore, you do not need to include both uppercase and lowercase versions of the words.

**Related information**

- Adding Custom Spell Check Dictionaries *(on page 451)*
- Adding Custom Spell Check Term Lists *(on page 453)*

**Print Preferences**

Oxygen XML Editor lets you configure how files are printed out of the editor. Note that these settings cover how files are printed directly from Oxygen XML Editor itself, not how they are printed after the XML source has been transformed by a publishing stylesheet. To configure the Print options, open the Preferences dialog box *(Options > Preferences)* *(on page 124)* and go to Editor > Print.

This page allows you to customize the headers and footers added to a printed page when you print from the Text mode *(on page 353)* or Author mode *(on page 354)* editors. These settings do not apply to the Grid *(on page 354)* and schema Design *(on page 355)* mode.

You can specify what is printed on the Left, Middle, and Right of the header and footer using plain text of any of the following variables:

- **${currentFileURL}** - Current file as URL, that is the absolute file path of the currently edited document represented as URL.
- **${cfne}** - Current file name with extension. The current file is the one currently open and selected.
- **${cp}** - Current page number. Used to display the current page number on each printed page in the Editor / Print Preferences page.
- **${tp}** - Total number of pages in the document. Used to display the total number of pages on each printed page in the Editor / Print Preferences page.
- **${env(VAR_NAME)}** - Value of the VAR_NAME environment variable. The environment variables are managed by the operating system. If you are looking for Java System Properties, use the **${system(var.name)}** editor variable.
- **${system(var.name)}** - Value of the var.name Java System Property. The Java system properties can be specified in the command-line arguments of the Java runtime as -Dvar.name=var.value. If you are looking for operating system environment variables, use the **${env(VAR_NAME)}** editor variable instead.
• \${date(pattern)} - Current date. The allowed patterns are equivalent to the ones in the Java
 SimpleDateFormat class. Example: yyyy-MM-dd.

Note:
This editor variable supports both the xs:date and xs:datetime parameters. For details about
 xs:date, go to: http://www.w3.org/TR/xmlschema-2/#date. For details about xs:datetime, go
to: http://www.w3.org/TR/xmlschema-2/#dateTime.

For example, to show the current page number and the total number of pages in the top right corner of the
page, write the following pattern in the Right text area of the Header section: \${cp} of \${tp}.

You can also set the Color and Font used in the header and footer. Default font is SansSerif.

You can place a line below the header or above the footer by selecting Underline/Overline.

CSS Validator Preferences

To configure the CSS Validator preferences, open the Preferences dialog box (Options > Preferences) (on
page 124) and go to CSS Validator.

You can configure the following options for the built-in CSS Validator of Oxygen XML Editor:

• Profile - Selects one of the available validation profiles: CSS 1, CSS 2, CSS 2.1, CSS 3, CSS 3 + SVG,
 CSS 3 with Oxygen extensions, SVG, SVG Basic, SVG Tiny, Mobile, TV Profile, ATSC TV Profile.
 The CSS 3 with Oxygen extensions profile includes all the CSS 3 standard properties plus the CSS
 extensions specific for Oxygen (on page 2333) that can be used in Author mode (on page 354). That
 means all Oxygen-specific extensions are accepted in a CSS stylesheet by the built-in CSS validator (on
 page 1076) when this profile is selected.

• Media type - Selects one of the available mediums: all, aural, braille, embossed, handheld, print,
 projection, screen, tty, tv, presentation, oxygen.

• Warning level - Sets the minimum severity level for reported validation warnings. Can be one of: All,
 Normal, Most Important, No Warnings.

• Ignore properties - You can type comma separated patterns that match the names of CSS properties
 that will be ignored at validation. The following vendor extensions are specified as ignored by default:
 -ro-* (PDFreactor), -ah-* (Antenna House), prince-* (Prince). As wildcards you can use:
   ◦ * to match any string.
   ◦ ? to match any character.

• Recognize browser CSS extensions (also applies to content completion) - If selected, Oxygen
 XML Editor recognizes browser-specific CSS properties (no validation is performed). The Content
 Completion Assistant (on page 3217) lists these properties at the end of its list, prefixed with the
 following particles:
   ◦ -moz- for Mozilla.
   ◦ -ms- for Edge.
-o- for Opera.

-webkit- for Safari/WebKit.

XML Preferences

This section describes the panels that contain the user preferences related with XML.

XML Catalog Preferences

To configure options that pertain to XML Catalogs (on page 3225), open the Preferences dialog box (Options > Preferences) (on page 124) and go to XML > XML Catalog.

The following options are available:

Prefer

Determines whether public identifiers specified in the catalog are used in favor of system identifiers supplied in the document. Suppose you have an entity in your document that has both a public identifier and a system identifier specified, and the catalog only contains a mapping for the public identifier (for example, a matching public catalog entry). You can choose between the following:

• system - If selected, the system identifier in the document is used.

• public - If selected, the URI supplied in the matching public catalog entry is used.

Generally, the purpose of catalogs is to override the system identifiers in XML documents, so public should usually be used for your catalogs.

Note:

If the catalog contains a matching system catalog entry giving a mapping for the system identifier, that mapping would have been used, the public identifier would never have been considered, and this setting would be irrelevant.

Verbosity

When using catalogs, it is sometimes useful to see what catalog files are parsed, if they are valid, and what identifiers are resolved by the catalogs. This option selects the detail level of such logging messages of the XML catalog resolver that will be displayed in the Catalogs table at the bottom of the window. You can choose between the following:

• None - No message is displayed by the catalog resolver when it tries to resolve a URI reference, a SYSTEM one or a PUBLIC one with the XML catalogs specified in this panel.

• Unresolved entities - Only the logging messages that track the failed attempts to resolve references are displayed.

• All messages - The messages of both failed attempts and successful ones are displayed.

Resolve schema locations also through system mappings
If selected, Oxygen XML Editor analyzes both *uri* and *system* mappings to resolve the location of schema.

**Note:** This option is not applicable for DTD schemas since the public and system catalog mappings are always considered.

**Process "schemaLocation" namespaces through URI mappings for XML Schema**

If selected, the target namespace of the imported XML Schema is resolved through the *uri* mappings. The namespace is taken into account only when the schema specified in the *schemaLocation* attribute was not resolved successfully. If not selected, the system IDs are used to resolve the schema location.

**Use default catalog**

If this option is selected and Oxygen XML Editor cannot resolve the catalog mapping with any other means, the default global catalog (listed below this checkbox) is used. For more information, see *How Oxygen XML Editor Determines which Catalog to Use (on page 827)*.

**Catalogs table**

You can use this table to add or manage global user-defined catalogs. The following actions are available at the bottom of the table:

- **Add**
  
  Opens a dialog box that allows you to add a catalog to the list. You can specify the path by using the text field, its history drop-down, the Insert Editor Variables (on page 323) button, or the browsing actions in the Browse drop-down list.

- **Edit**
  
  Opens a dialog box that allows you to edit an existing catalog. You can specify the path by using the text field, its history drop-down, the Insert Editor Variables (on page 323) button, or the browsing actions in the Browse drop-down list.

- **Delete**
  
  Deletes the currently selected catalog from the list.

- **Up**
  
  Moves the selection to the previous resource.

- **Down**
  
  Moves the selection to the following resource.
Note:
When you add, delete, or edit a catalog in this table, you need to reopen the currently edited files that use the modified catalog or run a manual Validate action (on page 775) so that the changes take full effect.

You can also add or configure catalogs at framework level from the Catalogs tab (on page 164) in the Document Type configuration dialog box (on page 140).

Related information
Controlling the Catalog Resolver
Working with XML Catalogs (on page 826)

XML Parser Preferences
To configure the XML Parser options, open the Preferences dialog box (Options > Preferences) (on page 124) and go to XML > XML Parser.

The configurable options of the built-in XML parser are as follows:

Enable parser caching (validation and content completion)
Enables re-use of internal models when validating and provides content completion in open XML files that reference the same schemas (grammars) such as DTD, XML Schema, or RelaxNG.

Enable system parameter entity expansion in other entity definitions
This security setting controls the expansion of the DTD system parameter entities (the ones that are loaded from disk or from remote sources). This option is off by default, to protect against XXE attacks. If you enable it, make sure the XML files you are opening or processing with the application come from a trusted source.

Ignore the DTD for validation if a schema is specified
Forces validation against a referenced schema (XML Schema, Relax NG schema) even if the document includes also a DTD DOCTYPE declaration. This option is useful when the DTD declaration is used only to declare DTD entities and the schema reference is used for validation against an XML Schema or a Relax NG schema.

Note:
Schematron schemas are treated as additional schemas. The validation of a document associated with a DTD and referencing a Schematron schema is executed against both the DTD and the Schematron schema, regardless of the value of the Ignore the DTD for validation if a schema is specified option.

Enable XInclude processing
Enables XInclude processing. If selected, the XInclude support in Oxygen XML Editor is turned on for validation, rendering in Author mode and transformation of XML documents.

**Base URI fix-up**

According to the specification for XInclude, processors must add an @xml:base attribute to elements included from locations with a different base URI. Without these attributes, the resulting infoset information would be incorrect.

Unfortunately, these attributes make XInclude processing to not be transparent to Schema validation. One solution to this is to modify your schema to allow @xml:base attributes to appear on elements that might be included from different base URIs.

If the addition of @xml:base and / or @xml:lang is not desired by your application, you can deselect this option.

**Language fix-up**

The processor will preserve language information on a top-level included element by adding an @xml:lang attribute if its included parent has a different [language] property. If the addition of @xml:lang is not allowed by your application, you can deselect this option.

**DTD post-validation**

Select this option to validate an XML file against the associated DTD, after all the content merged to the current XML file using XInclude was resolved. If you deselect this option, the current XML file is validated against the associated DTD before all the content merged to the current XML file using XInclude is resolved.

**XML Schema Preferences**

To configure options regarding XML Schema, open the Preferences dialog box (Options > Preferences) (on page 124) and go to XML > XML Parser > XML Schema.

This preferences page allows you to configure the following options:

**Default XML Schema version**

Allows you to select the version of XML Schema to be used as the default. You can choose XML Schema 1.0 or XML Schema 1.1.

**Note:**
You are also able to set the XML Schema version using the Customize option in the New document wizard (on page 368).

**Default XML Schema validation engine**

Allows you to select the default validation engine to be used for XML Schema. You can choose Xerces or Saxon EE.

**Xerces validation features section**
Enable full schema constraint checking

Sets the `schema-full-checking` feature to `true`. This enables a validation of the parsed XML document against a schema (XML Schema or DTD) while the document is parsed.

Enable honour all schema location feature

Sets the `honour-all-schema-location` feature to `true`. All the files that declare XML Schema components from the same namespace are used to compose the validation model. If this option is not selected, only the first XML Schema file that is encountered in the XML Schema import tree is taken into account.

Enable full XPath 2.0 for alternative types

When selected (default value), you can use the full XPath support in assertions and alternative types. Otherwise, only the XPath support offered by the Xerces engine is available.

Assertions can see comments and processing instructions

Controls whether or not comments and processing instructions are visible to the XPath expression used for defining an assertion in XSD 1.1.

Saxon EE validation features section

Multiple schema imports

Forces `<xs:import>` to fetch the referenced schema document. By default, the `<xs:import>` fetches the document only if no schema document for the given namespace has already been loaded. With this option in effect, the referenced schema document is loaded unless the absolute URI is the same as a schema document already loaded.

Assertions can see comments and processing instructions

Controls whether or not comments and processing instructions are visible to the XPath expression used to define an assertion. By default, they are not made visible (unlike Saxon 9.3).

Relax NG Preferences

To configure options regarding Relax NG, open the Preferences dialog box (Options > Preferences) (on page 124) and go to XML > XML Parser > Relax NG.

The following options are available in this page:

Check feasibly valid

Checks if Relax NG documents can be transformed into valid documents by inserting any number of attributes and child elements anywhere in the tree.
**Check ID/IDREF**

Checks the ID/IDREF matches when a Relax NG document is validated.

**Add default attribute values**

Default values are given to the attributes of documents validated using Relax NG. These values are defined in the Relax NG schema.

**Ignore "data-" attributes in XHTML**

This option is selected by default, which means that when XHTML documents are validated with an RNG schema, any `data-` attributes detected in the document will not be taken into account by the validation engine.

**Schematron Preferences**

To configure options regarding Schematron, open the Preferences dialog box (Options > Preferences) (on page 124) and go to XML > XML Parser > Schematron.

The following options are available in this preferences page:

**ISO Schematron Section**

- **Optimize (visit-no-attributes)**
  
  If your ISO Schematron assertion tests do not contain the attributes axis, you should select this option for faster ISO Schematron validation.

- **Allow foreign elements (allow-foreign)**
  
  Enables support for `allow-foreign` on ISO Schematron. This option is used to pass non-Schematron elements to the generated stylesheet.

- **Use associated XML Schema to expand default attribute values**
  
  When selected (default value), if the validated XML document has an XML Schema associated that contains default values for attributes defined in the XML content, the Schematron will be able to match on those default attributes.

- **Use Saxon EE (schema aware) for xslt2/xslt3 query language binding**
  
  When selected, Saxon EE is used for `xslt2/xslt3` query binding. If this option is not selected, Saxon PE is used.

- **Enable Schematron Quick Fixes (SQF) support**
  
  Allows you to enable or disable the support for Quick Fixes (on page 3223) in Schematron files. This option is selected by default.

- **Embedded rules query language binding**
You can control the query language binding used by the ISO Schematron embedded rules. You can choose between: xslt1, xslt2, or xslt3.

**Note:**
To control the query language binding for standalone ISO Schematron, you need to set the query language to be used with a `@queryBinding` attribute on the schema root element.

**Message language**

This option allows you to specify the language to be used in Schematron validation messages. You can choose between the following:

- **Use the language defined in the application** - The language that is specified in the Global Preferences page (on page 126) will be used and only the validation messages that match that language will be presented. You can use the Change application language link to navigate to the preferences page where you can specify the language to be used in the application.
- **Use the "xml:lang" attribute set on the Schematron root** - The language specified in the `@xml:lang` attribute from the Schematron root will be used and only the validation message that match that language will be presented.
- **Ignore the language and show all message** - All messages are displayed in whatever language is defined within the Schematron schema.
- **Custom** - Use this option to specify a custom language to be used and only the messages that match the specified language will be presented.

**Note:**
In all cases, if the selected language is not available for a validation error or warning, all messages will be displayed in whatever language is defined within the Schematron schema.

**Schematron 1.5 Section**

**XPath Version**

Allows you to select the version of XPath for the expressions that are allowed in Schematron assertion tests. You can choose between: 1.0, 2.0, or 3.0. This option is applied in both standalone Schematron 1.5 schemas and embedded Schematron 1.5 rules.

**Security**

**Disable Schematron security checks**
For security reasons, several security checks are performed on Schematron files that are not located inside a framework (on page 3219) or plugin (on page 3222). Select this option if your Schematron files are failing because of these security checks and you are unable to move them to a location recognized as safe (a framework or a plugin).

Sample XML Files Generator Preferences

The Generate Sample XML Files tool (on page 1006) (available on the Tools menu) allows you to generate XML instance documents based on an XML Schema. There are various options that can be configured within the tool and these options are also available in the Sample XML Files Generator preferences page. This allows you to set default values for these options. To configure the options for generating the XML files, open the Preferences dialog box (Options > Preferences) (on page 124) and go to XML > Sample XML Files Generator.

The following options are available:

Generate optional elements

When selected, all elements are generated, including the optional ones (having the minOccurs attribute set to 0 in the schema).

Generate optional attributes

When selected, all attributes are generated, including the optional ones (having the use attribute set to optional in the schema).

Values of elements and attributes

Controls the content of generated attribute and element values. The following choices are available:

- **None** - No content is inserted.
- **Default** - Inserts a default value depending on the data type descriptor of the particular element or attribute. The default value can be either the data type name or an incremental name of the attribute or element (according to the global option from the Sample XML Files Generator preferences page). Note that type restrictions are ignored when this option is selected. For example, if an element is of a type that restricts an xs:string with the xs:maxLength facet to allow strings with a maximum length of 3, the XML instance generator tool may generate string element values longer than 3 characters.
- **Random** - Inserts a random value depending on the data type descriptor of the particular element or attribute.

Important:

If all of the following are true, the Generate Sample XML Files tool outputs invalid values:
At least one of the restrictions is a regexp.

The value generated after applying the regexp does not match the restrictions imposed by one of the facets.

Preferred number of repetitions

Allows you to set the preferred number of repeating elements related to \texttt{minOccurs} and \texttt{maxOccurs} facets defined in the XML Schema.

- If the value set here is between \texttt{minOccurs} and \texttt{maxOccurs}, then that value is used.
- If the value set here is less than \texttt{minOccurs}, then the \texttt{minOccurs} value is used.
- If the value set here is greater than \texttt{maxOccurs}, then \texttt{maxOccurs} is used.

Maximum recursion level

If a recursion is found, this option controls the maximum allowed depth of the same element.

Type alternative strategy

Used for the \texttt{xs:alternative} element from XML Schema 1.1. The possible strategies are:

- **First** - The first valid alternative type is always used.
- **Random** - A random alternative type is used.

Choice strategy

Used for \texttt{xs:choice} or \texttt{substitutionGroup} elements. The possible strategies are:

- **First** - The first branch of \texttt{xs:choice} or the head element of \texttt{substitutionGroup} is always used.
- **Random** - A random branch of \texttt{xs:choice} or a substitute element or the head element of a \texttt{substitutionGroup} is used.

Generate the other options as comments

If selected, generates the other possible choices or substitutions (for \texttt{xs:choice} and \texttt{substitutionGroup}). These alternatives are generated inside comments groups so you can uncomment and use them later. Use this option with care (for example, on a restricted namespace and element) as it may generate large result files.

Use incremental attribute / element names as default

If selected, the value of an element or attribute starts with the name of that element or attribute. For example, for an \texttt{<a>} element the generated values are: \texttt{a1}, \texttt{a2}, \texttt{a3}, and so on. If not selected, the value is the name of the type of that element / attribute (for example: \texttt{string}, \texttt{decimal}, etc.)

Maximum length

The maximum length of string values generated for elements and attributes.

Discard optional elements after nested level
The optional elements that exceed the specified nested level are discarded. This option is useful for limiting deeply nested element definitions that can quickly result in very large XML documents.

Related information
Generating Sample XML Files (on page 1006)

XProc Preferences

Oxygen XML Editor includes a bundled version of the Calabash XProc engine that can be used for XProc transformations and validation, but you also have several ways to integrate other external XProc engines.

If the external engine is Java-based, or it has validation support, or it can receive parameters or ports passed from the transformation, you need to integrate the external XProc engine using a plugin extension procedure (on page 1534).

If you do not need the engine to be used for automatic validation or pass parameters/ports and it is not Java-based, you can add an external XProc engine by using the XProc preferences page. Open the Preferences dialog box (Options > Preferences) (on page 124) and go to XML > XProc.

To add an external engine, click the New button. To configure an existing engine, click the Edit button. This opens the Custom Engine dialog box that allows you to configure an external engine.

The following options can be configure in this custom engine configuration dialog box:
• **Name** - The value of this field will be displayed in the XProc transformation scenario and in the command line that will start it.

• **Description** - A textual description that will appear as a tooltip where the XProc engine will be used.

• **Working directory** - The working directory for resolving relative paths. You can specify the path by using the text field, the [Insert Editor Variables](on page 323) button, or the [Browse](on page 323) button.

• **Command line** - The command line that will run the XProc engine as an external process. You can specify the path by using the text field, the [Insert Editor Variables](on page 323) button, or the [Browse](on page 323) button.

• **Output encoding** - The encoding for the output stream of the XProc engine, used for reading and displaying the output messages.

• **Error encoding** - The encoding for the error stream of the XProc engine, used for reading and displaying the messages from the error stream.

**Note:**

You can configure the built-in Calabash processor by using the `calabash.config` file. This file is located in `[OXYGEN_INSTALL_DIR]\lib\xproc\calabash\lib`. If that file does not exist, you have to create it.

The **Show XProc messages** option at the bottom of the **XProc** preferences page can be selected if you want all messages emitted by the XProc processor during a transformation to be presented in dedicated XProc Results view (on page 548).

**XSLT/XQuery Preferences**

To configure the XSLT options, open the Preferences dialog box (Options > Preferences) (on page 124) and go to XML > XSLT-XQuery. This panel contains only the most generic options for working with XSLT or XQuery processors. The more specific options are grouped in other panels linked as child nodes of this panel in the tree of this Preferences page.

There is only one generic option available:

**Create transformation temporary files in system temporary directory**

It should be selected only when the temporary files necessary for performing transformations are created in the same folder as the source of the transformation (the default behavior when this option is not selected) and this breaks the transformation. An example of breaking the transformation is when the transformation processes all the files located in the same folder as the source of the transformation (including the temporary files) and the result is incorrect or the transformation fails because of this.

**XSLT Preferences**

To configure the XSLT options, open the Preferences dialog box (Options > Preferences) (on page 124) and go to XML > XSLT-XQuery > XSLT.
The XSLT preferences page allows you to customize options for the default XSLT validation engines. You can also specify the engine directly in a validation scenario (on page 798).

**Note:**
If no specific engine is specified in the validation scenario and the XSLT file has a transformation scenario associated, Oxygen XML Editor will use the engine specified in the transformation scenario.

The following options are available in this page:

- **Validation engine - XSLT 1.0**
  Allows you to select the XSLT engine to be used for validation of XSLT 1.0 documents.

- **Validation engine - XSLT 2.0**
  Allows you to select the XSLT engine to be used for validation of XSLT 2.0 documents.

- **Validation engine - XSLT 3.0**
  Allows you to select the XSLT engine to be used for validation of XSLT 3.0 documents.

**Note:**
Saxon-HE does not implement any XSLT 3.0 features. Saxon-PE implements a selection of XSLT 3.0 (and XPath 3.1) features, with the exception of schema-awareness and streaming. Saxon-EE implements additional features relating to streaming (processing of a source document without constructing a tree in memory). For further details about XSLT 3.0 conformance, go to http://www.saxonica.com/documentation/index.html#conformance/xslt30.

**XSLT Editor Content Completion Options link**

Use this link to switch to the XSLT Content Completion preferences page (on page 213), where you can configure the XSLT content completion options.

**Saxon6 Preferences**

To configure the Saxon 6 options, open the Preferences dialog box (Options > Preferences) (on page 124) and go to XML > XSLT-XQuery > XSLT > Saxon > Saxon6.

The built-in Saxon 6 XSLT processor can be configured with the following options:

- **Line numbering** - Specifies whether or not line numbers are maintained and reported in error messages for the XML source document.

- **Disable calls on extension functions** - If selected, external function calls are not allowed. Selecting this option is recommended in an environment where untrusted stylesheets may be executed. It also disables user-defined extension elements and the writing of multiple output files, since they carry similar security risks.
- **Handling of recoverable stylesheet errors** - Allows you to choose how dynamic errors are handled. One of the following options can be selected:
  - **recover silently** - Continue processing without reporting the error.
  - **recover with warnings** - Issue a warning but continue processing.
  - **signal the error and do not attempt recovery** - Issue an error and stop processing.

**Saxon-HE/PE/EE Preferences**

To configure global options for XSLT transformation and validation scenarios that use the Saxon HE/PE/EE engine, open the Preferences dialog box (Options > Preferences) (on page 124) and go to XML > XSLT-XQuery > XSLT > Saxon > Saxon-HE/PE/EE.

**Saxon-HE/PE/EE Options**

Oxygen XML Editor allows you to configure the following XSLT options for the Saxon 10.6 Home Edition (HE), Professional Edition (PE), and Enterprise Edition (EE):

- **Use a configuration file ("-config")**

  Select this option if you want to use a Saxon 10.6 configuration file that will be executed for the XSLT transformation and validation processes. You can specify the path to the configuration file by entering it in the URL field, or by using the Insert Editor Variables button, or using the browsing actions in the Browse drop-down list.

- **Debugger trace into XPath expressions (applies to debugging sessions)**

  Instructs the XSLT Debugger (on page 2121) to step into XPath expressions.

- **Enable Optimizations ("-opt")**

  This option is selected by default, which means that optimization is enabled. If not selected, the optimization is suppressed, which is helpful when reducing the compiling time is important, optimization conflicts with debugging, or optimization causes extension functions with side-effects to behave unpredictably.

- **Line numbering ("-l")**

  Line numbers where errors occur are included in the output messages.

- **Expand attributes defaults ("-expand")**

  Specifies whether or not the attributes defined in the associated DTD or XML Schema are expanded in the output of the transformation you are executing.

- **DTD validation of the source ("-dtd")**

  Specifies whether or not the source document will be validated against their associated DTD. You can choose from the following:

  - **On** - Requests DTD validation of the source file and of any files read using the `document()` function.
  - **Off** - (default setting) Suppresses DTD validation.
• **Recover** - Performs DTD validation but treats the errors as non-fatal.

**Note:**
Any external DTD is likely to be read even if not used for validation, since DTDs can contain definitions of entities.

**Strip whitespaces ("-strip")**

Specifies how the *strip whitespaces* operation is handled. You can choose one of the following values:

- **All ("all")** - Strips *all* whitespace text nodes from source documents before any further processing, regardless of any `@xml:space` attributes in the source document.
- **Ignore ("ignorable")** - Strips *all ignorable* whitespace text nodes from source documents before any further processing, regardless of any `@xml:space` attributes in the source document. Whitespace text nodes are ignorable if they appear in elements defined in the DTD or schema as having element-only content.
- **None ("none")** - Strips *no* whitespace before further processing.

**Saxon-PE/EE Options**

The following options are available for Saxon 10.6 Professional Edition (PE) and Enterprise Edition (EE) only:

**Register Saxon-JS extension functions and instructions**

Registers the Saxon-CE extension functions and instructions when compiling a stylesheet using the Saxon 10.6 processors.

**Note:**
Saxon-CE, being JavaScript-based, was designed to run inside a web browser. This means that you will use Oxygen XML Editor only for developing the Saxon-CE stylesheet, leaving the execution part to a web browser. See more details about executing such a stylesheet on Saxonica's website.

**Allow calls on extension functions ("-ext")**

If selected, the stylesheet is allowed to call external Java functions. This does not affect calls on integrated extension functions, including Saxon and EXSLT extension functions. This option is useful when loading an untrusted stylesheet (such as from a remote site using `http://[URL]`). It ensures that the stylesheet cannot call arbitrary Java methods and thus gain privileged access to resources on your machine.

**Enable assertions ("-ea")**

In XSLT 3.0, you can use the `<xsl:assert>` element to make assertions in the form of XPath expressions, causing a dynamic error if the assertion turns out to be false. If this option is
selected, XSLT 3.0 \texttt{xsl:assert} instructions are enabled. If it is not selected (default), the assertions are ignored.

**Saxon-EE Options**

The options available specifically for Saxon 10.6 Enterprise Edition (EE) are as follows:

**Validation of the source file ("-val")**

Requests schema-based validation of the source file and of any files read using \texttt{document()} or similar functions. It can have the following values:

- **Schema validation** ("strict") - This mode requires an XML Schema and allows for parsing the source documents with strict schema-validation enabled.
- **Lax schema validation** ("lax") - If an XML Schema is provided, this mode allows for parsing the source documents with schema-validation enabled but the validation will not fail if, for example, element declarations are not found.
- **Disable schema validation** - This specifies that the source documents should be parsed with schema-validation disabled.

**Validation errors in the result tree treated as warnings ("-outval")**

Normally, if validation of result documents is requested, a validation error is fatal. Selecting this option causes such validation failures to be treated as warnings.

**Write comments for non-fatal validation errors of the result document**

The validation messages for non-fatal errors are written (wherever possible) as a comment in the result document itself.

**Saxon-HE/PE/EE Advanced Preferences**

To configure the Saxon HE/PE/EE Advanced preferences, open the Preferences dialog box (Options > Preferences) (on page 124) and go to XML > XSLT-XQuery > XSLT > Saxon > Saxon-HE/PE/EE > Advanced.


- **URI Resolver class name** ("-r") - Specifies a custom implementation for the URI resolver used by the XSLT Saxon 10.6 transformer (the \texttt{-r} option when run from the command line). The class name must be fully specified and the corresponding \texttt{jar} or \texttt{class} extension must be configured from the dialog box for configuring the XSLT extension (on page 1467) for the particular transformation scenario.
- **Collection URI Resolver class name** ("-cr") - Specifies a custom implementation for the Collection URI resolver used by the XSLT Saxon 10.6 transformer (the \texttt{-cr} option when run from the command line). The class name must be fully specified and the corresponding \texttt{jar} or \texttt{class} extension must be configured from the dialog box for configuring the XSLT extension (on page 1467) for the particular transformation scenario.
XSLTProc Preferences

To configure XSLTProc options, open the Preferences dialog box (Options > Preferences) (on page 124) and go to XML > XSLT-XQuery > XSLT > XSLTProc.

The following options are available in this preferences page:

- **Enable Xinclude processing** - If selected, XInclude references will be resolved when XSLTProc is used as transformer in XSLT transformation scenarios (on page 1432).
- **Skip loading the document's DTD** - If selected, the DTD specified in the DOCTYPE declaration will not be loaded.
- **Do not apply default attributes from document's DTD** - If selected, the default attributes declared in the DTD and not specified in the document are not included in the transformed document.
- **Do not use Internet to fetch DTD's, entities or docs** - If selected, the remote references to DTD's and entities are not followed.
- **Maximum depth in templates stack** - If this limit of maximum templates depth is reached the transformation ends with an error.
- **Verbosity** - If selected, the transformation will output detailed status messages about the transformation process in the Warnings view.
- **Show version of libxml and libxslt used** - If selected, Oxygen XML Editor will display in the Warnings view the version of the libxml and libxslt libraries invoked by XSLTProc.
- **Show time information** - If selected, the Warnings view will display the time necessary for running the transformation.
- **Show debug information** - If selected, the Warnings view will display debug information about what templates are matched, parameter values, and so on.
- **Show all documents loaded during processing** - If selected, Oxygen XML Editor will display in the Warnings view the URL of all the files loaded during transformation.
- **Show profile information** - If selected, Oxygen XML Editor will display in the Warnings view a table with all the matched templates, and for each template will display: the match XPath expression, the template name, the number of template modes, the number of calls, the execution time.
- **Show the list of registered extensions** - If selected, Oxygen XML Editor will display in the Warnings view a list with all the registered extension functions, extension elements and extension modules.
- **Refuses to write to any file or resource** - If selected, the XSLTProc processor will not write any part of the transformation result to an external file on disk. If such an operation is requested by the processed XSLT stylesheet the transformation ends with a runtime error.
- **Refuses to create directories** - If selected, the XSLTProc processor will not create any directory during the transformation process. If such an operation is requested by the processed XSLT stylesheet the transformation ends with a runtime error.

MSXML Preferences (Deprecated)

To configure the MSXML options, open the Preferences dialog box (Options > Preferences) (on page 124) and go to XML > XSLT-XQuery > XSLT > MSXML (Legacy).
The options in this preferences page for the MSXML 3.0 and 4.0 processors are as follows:

**Validate documents during parse phase**

If selected, and either the source or stylesheet document has a DTD or schema that its content can be checked against, validation is performed.

**Do not resolve external definitions during parse phase**

By default, MSXML instructs the parser to resolve external definitions such as document type definition (DTD), external subsets or external entity references when parsing the source and stylesheet documents. If this option is selected, the resolution is disabled.

**Strip non-significant whitespaces**

If selected, strips non-significant white space from the input XML document during the load phase. Selecting this option can lower memory usage and improve transformation performance while, in most cases, creating equivalent output.

**Show time information**

If selected, the relative speed of various transformation steps can be measured, including:

- The time to load, parse, and build the input document.
- The time to load, parse, and build the stylesheet document.
- The time to compile the stylesheet in preparation for the transformation.
- The time to execute the stylesheet.

**Start transformation in this mode**

Although stylesheet execution usually begins in the empty mode, this default behavior may be changed by specifying another mode. Changing the start mode allows execution to jump directly to an alternate group of templates.

**MSXML.NET Preferences (Deprecated)**

To configure the MSXML.NET options, open the Preferences dialog box (Options > Preferences) (on page 124) and go to XML > XSLT-XQuery > XSLT > MSXML.NET (Legacy).

The options in this preferences page for the MSXML.NET processor are as follows:

**Enable XInclude processing**

If selected, XInclude references will be resolved when MSXML.NET is used as the transformer in the XSLT transformation scenario (on page 1432).

**Validate documents during parse phase**

If selected, and either the source or stylesheet document has a DTD or schema that its content can be checked against, validation is performed.

**Do not resolve external definitions during parse phase**
By default, MSXML instructs the parser to resolve external definitions such as document type definition (DTD), external subsets or external entity references when parsing the source and style sheet documents. If this option is selected, the resolution is disabled.

**Strip non-significant whitespaces**

If selected, strips non-significant white space from the input XML document during the load phase. Selecting this option can lower memory usage and improve transformation performance while, in most cases, creating equivalent output.

**Show time information**

If selected, the relative speed of various transformation steps can be measured, including:

- The time to load, parse, and build the input document.
- The time to load, parse, and build the stylesheet document.
- The time to compile the stylesheet in preparation for the transformation.
- The time to execute the stylesheet.

**Forces ASCII output encoding**

There is a known problem with the .NET 1.X XSLT processor (System.Xml.Xsl.XslTransform class). It does not support escaping of characters as XML character references when they cannot be represented in the output encoding. This means that it will be outputted as '?'.

Usually this happens when output encoding is set to ASCII. If this option is selected, the output is forced to be ASCII encoded and all non-ASCII characters get escaped as XML character references (\&nnnn; form).

**Allow multiple output documents**

This option allows you to create multiple result documents using the exsl:document extension element.

**Use named URI resolver class**

This option allows you to specify a custom URI resolver class to resolve URI references in <xsl:import> and <xsl:include> instructions (during XSLT stylesheet loading phase) and in document() functions (during XSL transformation phase).

**Assembly file name for URI resolver class**

This option specifies a file name of the assembly where the specified resolver class can be found. The Use named URI resolver class option (on page 252) specifies a partially or fully qualified URI resolver class name (for example, Acme.Resolvers.CacheResolver). Such a name requires additional assembly specification using this option or the Assembly GAC name for URI resolver class option (on page 252), but fully qualified class name (which always includes an assembly specifier) is all-sufficient.

**Assembly GAC name for URI resolver class**

This option specifies partially or fully qualified name of the assembly in the global assembly cache (GAC) where the specified resolver class can be found.
List of extension object class names

This option allows to specify extension object classes, whose public methods then can be used as extension functions in an XSLT stylesheet. It is a comma-separated list of namespace-qualified extension object class names. Each class name must be bound to a namespace URI using prefixes, similar to providing XSLT parameters.

Use specified EXSLT assembly

MSXML.NET supports a rich library of the EXSLT and EXSLT.NET extension functions embedded or in a plugin EXSLT.NET library. EXSLT support is enabled by default and cannot be disabled in this version. Use this option if you want to use an external EXSLT.NET implementation instead of a built-in one.

Credential loading source xml

This option allows you to specify user credentials to be used when loading XML source documents. The credentials should be provided in the username:password@domain format (all parts are optional).

Credential loading stylesheet

This option allows you to specify user credentials to be used when loading XSLT stylesheet documents. The credentials should be provided in the username:password@domain format (all parts are optional).

XQuery Preferences

To configure the XQuery options, open the Preferences dialog box (Options > Preferences) (on page 124) and go to XML > XSLT-XQuery > XQuery.

The following generic XQuery preferences are available:

Validation engine

Allows you to select the processor that will be used to validate XQuery documents. If you are validating an XQuery file that has an associated validation scenario, Oxygen XML Editor uses the processor specified in the scenario. If no validation scenario is associated, but the file has an associated transformation scenario, the processor specified in the scenario is used. If the processor does not support validation or if no scenario is associated, then the value from this combo box will be used as validation processor.

Size limit of Sequence view (MB)

When the result of an XQuery transformation is set as a sequence (Present as a sequence option (on page 1486)) in the transformation scenario, the size of one chunk of the result that is fetched from the database in lazy mode in one step is set in this option. If this limit is exceeded, go to the Sequence view (on page 1046) and click More results available to extract more data from the database.

Format transformer output
Specifies whether or not the output of the transformer is formatted and indented (pretty-print (on page 3222)).

Note:
This option is ignored if you choose Present as a sequence (on page 1486) (lazy extract data from a database) from the associated transformation scenario.

Create structure indicating the type nodes

If selected, Oxygen XML Editor takes the results of a query and creates an XML document containing copies of all items in the sequence, suitably wrapped.

Note:
This option is ignored if you choose Present as a sequence (on page 1486) (lazy extract data from a database) from the associated transformation scenario.

Saxon-HE/PE/EE Preferences

To configure global options for XQuery transformation and validation scenarios that use the Saxon HE/PE/EE engine, open the Preferences dialog box (Options > Preferences) (on page 124) and go to XML > XSLT-XQuery > XQuery > Saxon-HE/PE/EE.

Oxygen XML Editor allows you to configure the following XQuery options for the Saxon 10.6 Home Edition (HE), Professional Edition (PE), and Enterprise Edition (EE):

Use a configuration file ("-config")
- Sets a Saxon 10.6 configuration file that is used for XQuery transformation and validation scenarios.

Enable Optimizations ("-opt")
- This option is selected by default, which means that optimization is enabled. If not selected, the optimization is suppressed, which is helpful when reducing the compiling time is important, optimization conflicts with debugging, or optimization causes extension functions with side-effects to behave unpredictably.

Use linked tree model ("-tree:linked")
- This option activates the linked tree model.

Strip whitespaces ("-strip")
- Specifies how the strip whitespaces operation is handled. You can choose one of the following values:
• **All** ("all") - Strips *all* whitespace text nodes from source documents before any further processing, regardless of any \texttt{@xml:space} attributes in the source document.

• **Ignore** ("ignorable") - Strips all *ignorable* whitespace text nodes from source documents before any further processing, regardless of any \texttt{@xml:space} attributes in the source document. Whitespace text nodes are ignorable if they appear in elements defined in the DTD or schema as having element-only content.

• **None** ("none") - Strips *no* whitespace before further processing.

The following option is available for Saxon 10.6 Professional Edition (PE) and Enterprise Edition (EE) only:

**Allow calls on extension functions** ("-ext")

If selected, calls on external functions are allowed. Selecting this option is not recommended in an environment where untrusted stylesheets may be executed. It also disables user-defined extension elements and the writing of multiple output files, both of which carry similar security risks.

The options available specifically for Saxon 10.6 Enterprise Edition (EE) are as follows:

**Validation of the source file** ("-val")

Requests schema-based validation of the source file and of any files read using \texttt{document()} or similar functions. It can have the following values:

• **Schema validation** ("strict") - This mode requires an XML Schema and allows for parsing the source documents with strict schema-validation enabled.

• **Lax schema validation** ("lax") - If an XML Schema is provided, this mode allows for parsing the source documents with schema-validation enabled but the validation will not fail if, for example, element declarations are not found.

• **Disable schema validation** - This specifies that the source documents should be parsed with schema-validation disabled.

**Validation errors in the result tree treated as warnings** ("-outval")

Normally, if validation of result documents is requested, a validation error is fatal. Selecting this option causes such validation failures to be treated as warnings.

**Write comments for non-fatal validation errors of the result document**

The validation messages for non-fatal errors are written (wherever possible) as a comment in the result document itself.

**Enable XQuery update** ("-update:(on|off)")

This option controls whether or not XQuery update syntax is accepted. The default value is off.

**Backup files updated by XQuery** ("-backup:(on|off)")
If selected, backup versions for any XML files updated with an XQuery Update are generated. This option is available when the Enable XQuery update option is selected.

Saxon HE/PE/EE Advanced Preferences

To configure Saxon HE/PE/EE Advanced preferences, open the Preferences dialog box (Options > Preferences) (on page 124) and go to XML > XSLT-XQuery > XQuery > Saxon-HE/PE/EE > Advanced.

The advanced XQuery options that can be configured for the Saxon 10.6 XQuery transformer (all editions: Home Edition, Professional Edition, Enterprise Edition) are as follows:

- **URI Resolver class name** - Allows you to specify a custom implementation for the URI resolver used by the XQuery Saxon 10.6 transformer (the -r option when run from the command line). The class name must be fully specified and the corresponding JAR or class extension must be configured from the dialog box for configuring the XQuery extension (on page 1467) for the particular transformation scenario.

  **Note:** If your URIResolver implementation does not recognize the given resource, the resolve(String href, String base) method should return a null value. Otherwise, the given resource will not be resolved through the XML Catalog (on page 826).

- **Collection URI Resolver class name** - Allows you to specify a custom implementation for the Collection URI resolver used by the XQuery Saxon 10.6 transformer (the -cr option when run from the command line). The class name must be fully specified and the corresponding JAR or class extension must be configured from the dialog box for configuring the XQuery extension (on page 1467) for the particular transformation scenario.

Debugger Preferences

To configure the Debugger preferences, open the Preferences dialog box (Options > Preferences) (on page 124) and go to XML > XSLT-XQuery > Debugger.

The following options are available:

- **Show xsl:result-document output**

  If selected, the debugger presents the output of `<xsl:result-document>` instructions into the debugger output view.

- **Infinite loop detection**

  Select this option to receive notifications when an infinite loop occurs during transformation.

- **Enable Saxon optimizations**

  This option is not selected by default and this means that the optimization for the debugging process is suppressed. This is helpful when reducing the compiling time is important,
optimization conflicts with debugging, or optimization causes extension functions with side-effects to behave unpredictably.

**Maximum depth in templates stack**

Allows you to set how many `<xsl:template>` instructions can appear on the current stack. This setting is used by the infinite loop detection.

**Debugger layout**

If you select the **Horizontal** layout, the stack of XML editors is presented on the left half of the editing area while the stack of XSL editors is on the right half. If you select the **Vertical** layout, the stack of XML editors is presented on the upper half of the editing area while the stack of XSL editors is on the lower half.

**Debugger current instruction pointer**

Allows you to set the background color of the current execution node, both in the document (XML) and XSLT/XQuery views.

**XWatch evaluation timeout (seconds)**

Allows you to specify the maximum time that Oxygen XML Editor allocates to the evaluation of XPath expressions while debugging.

**Messages**

Allows you to specify how to handle the debugging process when the source document involved in a debugging session is edited. You can choose one of the following:

- Ask me what to do
- Always stop the debugging session
- Never stop the debugging session

**Profiler Preferences**

This section explains the settings available for the XSLT/XQuery Profiler. To access and modify these settings, open the **Preferences** dialog box (**Options > Preferences**) (on page 124) and go to **XML > XSLT-XQuery > Profiler** (see Debugger Preferences (on page 256)).

The following profiler settings are available:

**Show time**

Shows the total time that was spent in the call.

**Show inherent time**

Shows the inherent time that was spent in the call. The inherent time is defined as the total time of a call minus the time of its child calls.

**Show invocation count**

Shows how many times the call was called in this particular call sequence.

**Time scale**
Determines the unit of time measurement. You can choose between milliseconds or microseconds.

**Hotspot threshold**

*Hotspots* are ignored below this specified amount (in milliseconds). For more information, see [Hotspots View](on page 2129).

**Ignore invocation less than**

*Invocations* are ignored below this specified amount (in microseconds). For more information, see [Invocation Tree View](on page 2128).

**Percentage calculation**

The percentage base that determines what time span percentages are calculated against. You can choose between the following:

- **Absolute** - Percentage values show the contribution to the total time.
- **Relative** - Percentage values show the contribution to the calling call.

**XPath Preferences**

To configure XPath options, open the [Preferences dialog box](Options > Preferences) (on page 124) and go to XML > XSLT-XQuery > XPath.

Oxygen XML Editor allows you to customize the following options:

**Unescape XPath expression**

If selected, the entities of an XPath expression that you type in the [XPath/XQuery Builder](on page 2002) and the XPath toolbar (on page 2000) are unescaped during their execution. For example, the expression:

```
//varlistentry[starts-with(@os,'s')]
```

is equivalent to:

```
//varlistentry[starts-with(@os,'s')]
```

**Multiple XPath results**

Select this option to display the results of an XPath expression in separate tabs in the [Results view](on page 548).

**XPath Default Namespace (only for XPath version 2.0)**

Specifies the default namespace to be used for unprefixed element names. You can choose between the following four options:

- **No namespace** - If selected, Oxygen XML Editor considers unprefixed element names of the evaluated XPath expressions as belonging to no namespace.
- **Use the default namespace from the root element** (default selection) - Oxygen XML Editor considers unprefixed element names of the evaluated XPath expressions as belonging
to the default namespace declared on the root element of the XML document you are querying.

- **Use the namespace of the root** - If selected, Oxygen XML Editor considers unprefixed element names of the evaluated XPath expressions as belonging to the same namespace as the root element of the XML document you are querying.
- **This namespace** - If selected, you can use the corresponding text field to enter the namespace of the unprefixed elements.

**Default prefix-namespace mappings**

You can use this table to associate prefixes with namespaces. Use these mappings when you want to define them globally (not for each document). Use the **New** button to add mappings to the list and the **Delete** button to remove mappings.

**Custom Engines Preferences**

Oxygen XML Editor allows you to configure custom processors to be used for running XSLT and XQuery transformations.

**Note:**

You can not use these custom engines in the Debugger perspective (on page 2102).

To configure the **Custom Engines** preferences, open the Preferences dialog box (Options > Preferences) (on page 124) and go to XML > XSLT-XQuery > Custom Engines.

The table in this preferences page displays the custom engines that have been defined. Use the **New** or **Edit** button at the bottom of the table to open a dialog box that allows you to add or configure a custom engine.
The following parameters can be configured for a custom engine:

**Engine type**

Specifies the transformer type. You can choose between XSLT and XQuery engines.

**Name**

The name of the transformer displayed in the dialog box for editing transformation scenarios.

**Description**

A textual description of the transformer.

**Working directory**

The start directory of the executable program for the transformer. The following editor variables (on page 323) are available for making the path to the working directory independent of the location of the input files:

- `${homeDir}` - The user home directory in the operating system.
- `${cfd}` - The path to the directory of the current file.
- `${pd}` - The path to the directory of the current project.
- `${oxygenInstallDir}` - The Oxygen XML Editor install directory.

**Command line**

The command line that must be executed by Oxygen XML Editor to perform a transformation with the engine. The following editor variables (on page 323) are available for making the parameters in the command line (the transformer executable, the input files) independent of the location of the input files:
• ${xml} - The XML input document as a file path.
• ${xmlu} - The XML input document as a URL.
• ${xsl} - The XSL / XQuery input document as a file path.
• ${xslu} - The XSL / XQuery input document as a URL.
• ${out} - The output document as a file path.
• ${outu} - The output document as a URL.
• ${ps} - The platform separator that is used between library file names specified in the class path.

Output Encoding

The encoding of the transformer output stream.

Error Encoding

The encoding of the transformer error stream.

PDF Output Preferences

The PDF Output preferences page simply includes links to sub-pages for configuring PDF output options.

FO Processors Preferences

Oxygen XML Editor includes a built-in formatting objects processor (Apache FOP), but you can also configure other external processors and use them in the transformation scenarios for processing XSL-FO documents.

Oxygen XML Editor provides an easy way to add two of the most commonly used commercial FO processors: RenderX XEP and Antenna House Formatter. You can easily add RenderX XEP as an external FO processor if you have the XEP installed. Also, if you have the Antenna House Formatter, Oxygen XML Editor uses the environment variables set by the XSL formatter installation to detect and use it for XSL-FO transformations. If the environment variables are not set for the XSL formatter installation, you can browse and choose the executable file just as you would for XEP. You can use these two external FO processors for DITA-OT transformations scenarios (on page 1490) and XML with XSLT transformation scenarios (on page 1464).

To configure the options for the FO processors, open the Preferences dialog box (Options > Preferences) (on page 124) and go to XML > PDF Output > FO Processors. This preferences page includes the following options:

Apache FOP Section

In this section, you can configure options for the built-in Apache processor. The following options are available:

Use built-in Apache FOP

Instructs Oxygen XML Editor to use the built-in Apache FO processor. To see the version of the built-in XSL-FO processor for your installation, go to Help > About > Libraries and search for Apache FOP.
Use other Apache FOP

Instructs Oxygen XML Editor to use another Apache FO processor that is installed on your computer. You can specify the path by using the text field, the Insert Editor Variables (on page 323) button, or the Browse button.

Enable the output of the built-in FOP

All Apache FOP output is displayed in a results pane at the bottom of the Oxygen XML Editor window, including warning messages about FO instructions not supported by Apache FOP.

Memory available to the Apache FOP

If your Apache FOP transformations fail with an Out of Memory error (OutOfMemoryError), use this combo box to select a larger value for the amount of memory reserved for Apache FOP transformations.

Configuration file for the built-in FOP

Use this option to specify the path to an Apache FOP configuration file (for example, to render to PDF a document containing Unicode content using a special true type font). You can specify the path by using the text field, the Insert Editor Variables (on page 323) button, or the Browse button.

Generates PDF/A-1b output

When selected, PDF/A-1b output is generated.

Notes:

- All fonts have to be embedded, even the implicit ones. More information about configuring metrics files for the embedded fonts can be found in Add a font to the built-in FOP (on page 1528).
- You cannot use the <filterList> key in the configuration file since the FOP would generate the following error: The Filter key is prohibited when PDF/A-1 is active.

External FO Processors Section

In this section, you can manage the external FO processors you want to use in transformation scenarios. You can use the two options at the bottom of the section to use the RenderX XEP or Antenna House Formatter commercial FO processors.

Add 'XEP' FO processor (executable file is needed)

If RenderX XEP is already installed on your computer, you can use this button to choose the XEP executable script (xep.bat for Windows, xep for Linux).

Add 'Antenna House' FO processor (executable file is needed)
If Antenna House Formatter is already installed on your computer, you can use this button to choose the Antenna House executable script (AHFCmd.exe or XSLCmd.exe for Windows, and run.sh for Linux/macOS).

**Note:**
The built-in Antenna House Formatter GUI transformation scenario requires that you configure an external FO processor that runs AHFormatter.exe (Windows only). In the external FO Processor configuration dialog box (on page 263), you could use "${env(AHF63_64_HOME)}}\AHFormatter.exe" -d ${fo} -s for the value in the Command line field, although the environment variable name changes for each version of the AH Formatter and for each system architecture (you can install multiple versions side-by-side). For more information, see https://github.com/AntennaHouse/focheck/wiki/focheck.

You can also add external processors or configure existing ones. Click the New button to open a configuration dialog box that allows you to add a new external FO processor. Use the other buttons (Edit, Duplicate, Delete, Up, Down) to configure existing external processors.

**Figure 29. External FO Processor Configuration Dialog Box**

The external FO Processor configuration dialog box includes the following options:

**Name**

The name that will be displayed in the list of available FO processors on the FOP tab of the transformation scenario dialog box.

**Description**
A textual description of the FO processor that will be displayed in the FO processors table and in tooltips of UI components where the processor is selected.

**Working directory**

The directory where the intermediate and final results of the processing are stored. You can specify the path by using the text field, the [Insert Editor Variables](on page 323) button, or the [Browse] button. You can use one of the following editor variables (on page 323):

- `${homeDir}` - The path to the user home directory.
- `${cfd}` - The path of the current file directory. If the current file is not a local file, the target is the user desktop directory.
- `${pd}` - The project directory.
- `${oxygenInstallDir}` - The Oxygen XML Editor installation directory.

**Command line**

The command line that starts the FO processor, specific to each processor. You can specify the path by using the text field, the [Insert Editor Variables](on page 323) button, or the [Browse] button. You can use one of the following editor variables (on page 323):

- `${method}` - The FOP transformation method: pdf, ps, or txt.
- `${fo}` - The input FO file.
- `${out}` - The output file.
- `${pd}` - The project directory.
- `${frameworksDir}` - The path of the frameworks subdirectory of the Oxygen XML Editor installation directory.
- `${oxygenInstallDir}` - The Oxygen XML Editor installation directory.
- `${ps}` - The platform-specific path separator. It is used between the library files specified in the class path of the command line.

**Output Encoding**

The encoding of the FO processor output stream that is displayed in a Results panel (on page 548) at the bottom of the Oxygen XML Editor window.

**Error Encoding**

The encoding of the FO processor error stream that is displayed in a Results panel (on page 548) at the bottom of the Oxygen XML Editor window.

**CSS-based Processors Preferences**

Oxygen XML Editor includes a built-in XML to PDF transformation with CSS scenario type for generating PDF output using a CSS-based processor.
To configure the options for the CSS-based processors, open the Preferences dialog box (Options > Preferences) (on page 124) and go to XML > PDF Output > CSS-based Processors. This preferences page includes the following options:

**Oxygen PDF Chemistry Section**

**Auto-detect**

If selected, the directory of the Chemistry processor will be automatically detected. This is based on the system's PATH environmental variable. If none is detected, it will use the path of the built-in distribution.

**Custom installation directory**

Use this option to select an external directory of a custom installation of the Chemistry processor.

**Memory available to the processor (MB)**

Specifies the maximum amount of memory that is available for the transformation. If your transformations fail with an Out of Memory error (OutOfMemoryError), you can use this option to select a bigger value for the amount of memory reserved for the process.

**Generates PDF/UA-1 output**

Use this option to produce output that conforms with the PDF/UA-1 accessibility standards.

**Note:**

This mode has some special requirements. For example, all fonts have to be embedded and the title of documents must be marked using the metadata. For more information, see Oxygen PDF Chemistry User Guide: Fully Accessible PDF (PDF/UA1).

**Show console output**

Allows you to specify when to display the console output log in the message panel at the bottom of the editor. The following options are available:

- **When build fails** - Displays the console output log only if the build fails.
- **Always** - Displays the console output log, regardless of whether or not the build fails.

**Ant Preferences**

To set Ant preferences, open the Preferences dialog box (Options > Preferences) (on page 124) and go to XML > Ant. This panel allows you to choose the directory containing the Apache Ant (on page 3216) libraries (the so-called Ant Home) that Oxygen XML Editor uses to handle Ant build files.

There are two options available:

- **Built-in** - the path to the Ant distribution that comes bundled with Oxygen XML Editor installation kit.
- **Custom** - the path to an Ant distribution of your choice.
Import Preferences

To configure importing options, open the Preferences dialog box (Options > Preferences) (on page 124) and go to XML > Import. This page allows you to configure how empty values and null values are handled when they are encountered in imported database tables or Excel sheets. Also you can configure the format of date / time values recognized in the imported database tables or Excel sheets.

The following options are available:

**Create empty elements for empty values**
If selected, an empty value from a database column or from a text file is imported as an empty element.

**Create empty elements for null values**
If selected, null values from a database column are imported as empty elements.

**Escape XML content**
Selected by default, this option instructs Oxygen XML Editor to escape the imported content to an XML-safe form.

**Add annotations for generated XML Schema**
If selected, the generated XML Schema contains an annotation for each of the imported table columns. The documentation inside the annotation tag contains the remarks of the database columns (if available) and also information about the conversion between the column type and the generated XML Schema type.

**Date / Time Format section**
Specifies the format used for importing date and time values from Excel spreadsheets or database tables, and in the generated XML schemas. You can choose from the following format types:

- **Unformatted** - The date and time formats specific to the database are used for import. When importing data from Excel a string representation of date or time values are used. The type used in the generated XML Schema is `xs:string`.
- **XML Schema date format** - The XML Schema-specific format ISO8601 is used for imported date / time data (`yyyy-MM-dd'T'HH:mm:ss` for `datetime`, `yyyy-MM-dd` for `date` and `HH:mm:ss` for `time`). The types used in the generated XML Schema are `xs:datetime`, `xs:date` and `xs:time`.
- **Custom format** - If selected, you can define a custom format for timestamp, date, and time values or choose one of the predefined formats. A preview of the values is presented when a format is used. The type used in the generated XML Schema is `xs:string`. 
### Table 3. Pattern Letters

<table>
<thead>
<tr>
<th>Letter</th>
<th>Date or Time Component</th>
<th>Presentation</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>G</td>
<td>Era designator</td>
<td>Text</td>
<td>AD</td>
</tr>
<tr>
<td>y</td>
<td>Year</td>
<td>Year</td>
<td>1996; 96</td>
</tr>
<tr>
<td>M</td>
<td>Month in year</td>
<td>Month</td>
<td>July; Jul; 07</td>
</tr>
<tr>
<td>w</td>
<td>Week in year</td>
<td>Number</td>
<td>27</td>
</tr>
<tr>
<td>W</td>
<td>Week in month</td>
<td>Number</td>
<td>2</td>
</tr>
<tr>
<td>D</td>
<td>Day in year</td>
<td>Number</td>
<td>189</td>
</tr>
<tr>
<td>d</td>
<td>Day in month</td>
<td>Number</td>
<td>10</td>
</tr>
<tr>
<td>F</td>
<td>Day of week in month</td>
<td>Number</td>
<td>2</td>
</tr>
<tr>
<td>E</td>
<td>Day in week</td>
<td>Text</td>
<td>Tuesday; Tue</td>
</tr>
<tr>
<td>a</td>
<td>Am / pm marker</td>
<td>Text</td>
<td>PM</td>
</tr>
<tr>
<td>H</td>
<td>Hour in day (0-23)</td>
<td>Number</td>
<td>0</td>
</tr>
<tr>
<td>k</td>
<td>Hour in day (1-24)</td>
<td>Number</td>
<td>24</td>
</tr>
<tr>
<td>K</td>
<td>Hour in am / pm (0-11)</td>
<td>Number</td>
<td>0</td>
</tr>
<tr>
<td>h</td>
<td>Hour in am / pm (1-12)</td>
<td>Number</td>
<td>12</td>
</tr>
<tr>
<td>m</td>
<td>Minute in hour</td>
<td>Number</td>
<td>30</td>
</tr>
<tr>
<td>s</td>
<td>Second in minute</td>
<td>Number</td>
<td>55</td>
</tr>
<tr>
<td>S</td>
<td>Millisecond</td>
<td>Number</td>
<td>978</td>
</tr>
<tr>
<td>z</td>
<td>Time zone</td>
<td>General time zone</td>
<td>PST; GMT-08:00</td>
</tr>
<tr>
<td>Z</td>
<td>Time zone</td>
<td>RFC 822 time zone</td>
<td>-0800</td>
</tr>
</tbody>
</table>

Pattern letters are usually repeated, as their number determines the exact presentation:

- **Text** - If the number of pattern letters is 4 or more, the full form is used. Otherwise, a short or abbreviated form is used if available.
- **Number** - The number of pattern letters is the minimum number of digits, and shorter numbers are zero-padded to this amount.
- **Year** - If the number of pattern letters is 2, the year is truncated to 2 digits. Otherwise, it is interpreted as a number.
- **Month** - If the number of pattern letters is 3 or more, the month is interpreted as text. Otherwise, it is interpreted as a number.
- **General time zone** - Time zones are interpreted as text if they have names. For time zones representing a GMT offset value, the following syntax is used:
• **GMTOffsetTimeZone** - GMT Sign Hours: Minutes
  • **Sign** - one of + or -
  • **Hours** - one or two digits
  • **Minutes** - two digits
  • **Digit** - one of 0 1 2 3 4 5 6 7 8 9

Hours must be between 0 and 23, and Minutes must be between 00 and 59. The format is locale independent and digits must be taken from the Basic Latin block of the Unicode standard.

• **RFC 822 time zone**: The RFC 822 4-digit time zone format is used:
  • **RFC822TimeZone**
  • **TwoDigitHours** (must be between 00 and 23)

### XML Signing Certificates Preferences

Oxygen XML Editor provides two types of **keystores (on page 3220)** for certificates that are used for digital signatures of XML documents: Java Keystore (**JKS**) and Public-Key Cryptography Standards version 12 (**PKCS-12**). A **keystore** file is protected by a password. To configure a certificate **keystore**, open the Preferences dialog box (**Options > Preferences**) (on page 124) and go to **XML > XML Signing Certificates**. You can customize the following parameters of a **keystore**:

**Figure 30. Certificates Preferences Panel**

- **Keystore type** - The type of **keystore (on page 3220)** that Oxygen XML Editor uses (**JKS** or **PKCS-12**).
- **Keystore file** - The location of the imported file.
- **Keystore password** - The password that is used for protecting the privacy of the stored keys.
- **Certificate alias** - The alias used for storing the key entry (the certificate or the private key) inside the **keystore (on page 3220)**.
- **Private key password** - The private key password of the certificate (required only for JKS **keystores (on page 3220)**).
- **Validate** - Click this button to verify the configured **keystore (on page 3220)** and the validity of the certificate.
XML Refactoring Preferences

To specify a folder for loading the custom XML refactoring operations, open the Preferences dialog box (Options > Preferences) and go to XML > XML Refactoring. The following option is available in this preferences page:

**Load additional refactoring operations from**

Use this text box to specify a folder for loading custom XML refactoring operations. You can specify the path by using the text field, the Insert Editor Variables button, or the Browse button. Oxygen XML Editor looks for XML refactoring operations recursively in the specified folder, so they can be stored in descendant folders.

DITA Preferences

To access the DITA Preferences page, open the Preferences dialog box (Options > Preferences) and go to DITA. This preferences page includes the following sections and options:

**DITA Open Toolkit section**

This section allows you to specify the default directory of the DITA Open Toolkit distribution (bundled with the Oxygen XML Editor installation) to be used for validating and publishing DITA content. You can select from the following:

**Built-in Oxygen Publishing Engine (based on DITA-OT 3.x)**

Oxygen XML Editor comes bundled with the Oxygen Publishing Engine (based on DITA-OT 3.7.0). By default, all defined DITA transformation/validation scenarios will run with this version. The default publishing engine directory is: [OXYGEN_INSTALL_DIR]/frameworks/dita/DITA-OT3.x.

**Custom**

Allows you to specify a custom directory for your DITA-OT distribution.

**Location**

You can either provide a new file path for the specific DITA-OT that you want to use or select a previously used one from the drop-down list. You can specify the path by using the text field, the Insert Editor Variables button, or the Browse button.

**Important:**

Using a custom DITA Open Toolkit may disable certain features in the application. Examples of features that may not work properly:
• If the custom DITA-OT is missing certain publishing plugins, default transformation scenarios such as DITA Map WebHelp Responsive (on page 3082) or DITA Map PDF - based on HTML5 & CSS (on page 3095) may no longer work properly.

• Validation of Markdown documents using Schematron may not work because it is based on a certain DITA Open Toolkit plugin.

• The DITA framework (defined in the Preferences > Document Type Associations page) will use the XML catalogs specified in the DITA-OT configured in the Preferences > DITA page to perform the validation of all DITA topic types. If this DITA-OT is different from the one that comes bundled with the Oxygen XML Editor default distribution, you might encounter validation-related issues.

**CAUTION:**

Oxygen XML Editor support engineers cannot offer support and troubleshooting assistance for a custom DITA-OT. If you discover any issues or inconsistent behavior while using a custom DITA-OT, you should revert to the default built-in DITA-OT.

Enable DITA 2.0 editing support (Experimental)

If selected, you will have access to a DITA 2.0 folder in the New Document Wizard (on page 368) where you can find new document templates for creating DITA 2.0 maps or topics based on the DITA 2.0 standard DTDs. For example, in a DITA topic based on the DITA 2.0 DTDs, you can insert an `<include>` element that is not found in the DITA 1.3 DTDs.

**DITA Maps Preferences**

To access the DITA Maps preferences page, open the Preferences dialog box (Options > Preferences) (on page 124) and go to DITA > Maps. This preferences page includes the following options:

**DITA Maps file patterns**

Allows you to specify the extension types that will be handled as DITA maps when opened in Oxygen XML Editor.

**When opening a map**

Oxygen XML Editor can open a DITA map in the regular editor view or in the DITA Maps Manager (on page 2897). This option allows you to specify how a map will be opened. You can choose one of the following options:
• Always open in the DITA Maps Manager - A DITA map file is always opened in the DITA Maps Manager view.
• Always open as XML - A DITA map file is always opened in the XML editor.
• Always ask - When opening a DITA map, you are prompted to choose between opening it in the XML editor panel or in the DITA Maps Manager view.

Expand references to other maps when opening a map in Author mode

Enabling this option will improve performance (decrease the loading time) when opening maps in Author mode (particularly maps that contain a large amount of submaps). This option is disabled by default.

Note:
You must close and reopen the map to see the effects of enabling/disabling this option.

DITA Maps Manager section

Prefer using the navigation title for rendering topic references

If selected and there is a @navtitle attribute set on a <topicref>, then the @navtitle is used to render the title of the topic in the DITA Maps Manager (on page 2897).

Allow referenced submaps to be edited

If selected, all DITA maps referenced directly or indirectly in a DITA map that is open in the DITA Maps Manager view will be fully editable. You will be able to add new topic references, modify properties, and move topic references from one submap to another. Saving the main DITA map will also save the contents of the modified submaps.

Attention:
The documents must be reopened to apply a change to this option.

Local files only

If selected, only submaps that are located on local disk drives will be editable. As for maps located in remote locations (e.g. content management systems), the save operation might not work on all submaps. This checkbox is selected by default.

Inserting Topic References section

Always set values for the following attributes

Allows you to specify that when inserting a topic reference (using the Insert Reference dialog box (on page 2923) and Edit Properties dialog box (on page 2932)), the values for certain attributes will always be automatically populated with a detected value (based on the specifications), even if it is the same as the default value. You can choose to always populate the values for the following attributes:
• **Format** - If selected, the `@format` attribute will always be automatically populated with a detected value.

• **Scope** - If selected, the `@scope` attribute will always be automatically populated with a detected value.

• **Type** - If selected, the `@type` attribute will always be automatically populated with a detected value.

• **Navigation title** - If selected, the `@navtitle` attribute will always be automatically populated with a detected value.

**Use the file name as the value of the "keys" attribute**

If selected, when inserting a topic reference into a map, the file name will be used as the value of the `@keys` attribute for the new `<topicref>`. This option has a slightly different effect depending on the method used for inserting the topic reference:

• **Drag/Drop or Copy/Paste in the DITA Maps Manager** - If you drag or copy a resource from another view (or outside Oxygen XML Editor) and drop or paste it into the **DITA Maps Manager** and a key is already defined for that resource, a `@keyref` attribute will be inserted instead. This even works for two consecutive drag/drop or copy/paste operations without saving the file and it works for multiple selections of topic references.

• **Drag/Drop or Copy/Paste in a Map opened in Author Mode** - If you drag or copy a resource from another view (or outside Oxygen XML Editor) and drop or paste it into a map that is open in **Author** mode, and a key is already defined for that resource, a `@keyref` attribute will be inserted instead.

**Restriction:**

In this particular scenario, if you perform two identical, consecutive drag/drop or copy/paste operations without saving the file between operations, the value of the `@keys` attribute will be the same for both inserted topic references. The workaround for this limitation is to simply save the map after each drag/drop or copy/paste operation.

• **Using the Fast Create Topics or Duplicate Actions** - If you use the **Fast Create Topics feature** *(on page 2964)* or **Duplicate action** *(on page 2961)* to insert topic references, the newly created `<topicref>` elements will contain the `@keys` attribute with its value set depending on the file name.

• **Using the Insert Topic Reference Dialog Box from the DITA Maps Manager** - If you use an action in the **DITA Maps Manager** to insert topic references (e.g. **Append Child > Reference**), a `@keys` attribute will be set for each inserted topic reference and the value depends on the file name. If a single target was selected, you can see the value in the **Define keys** field from the **Keys** tab of the **Insert Reference** dialog box *(on page 2923)*.
and you can change it, while if multiple targets were selected, the values are automatically generated based on each file name when the insertion is performed, and you cannot see or change the values in that dialog box.

**Note:**
This option also has an effect on image references. When inserting a reference to an image in a DITA map and this option is selected, a `<keydef>` element is created if it is allowed by the schema. If it is not allowed (or this option is deselected), a specific topic reference element is created with the value of the `@processing-role` attribute set to `resource-only`.

**Review section**

**When the last/first item is reached while navigating review items**

This option allows you to specify what should happen when you are navigating review items in the Review view (on page 664) and you reach the last or first review item. You can choose one of the following options:

- **Open the next/previous document (in the current DITA map hierarchy) that contains review items** - If you reach the last/first review item in the document, clicking the **Next** or **Previous** navigation buttons will open the next/previous document (from the current DITA map hierarchy) that contains review items.

- **Do nothing** - If you reach the last/first review item in the document, clicking the **Next** or **Previous** navigation buttons will do nothing.

- **Always ask** - If you reach the last/first review item in the document, clicking the **Next** or **Previous** navigation buttons will open a dialog box asking if you want to open the next/previous document (from the current DITA map hierarchy) that contains review items.

**DITA New Topics Preferences**

To access the DITA New Topics preferences page, open the Preferences dialog box (Options > Preferences) (on page 124) and go to DITA > New Topics. This preferences page includes the following options:

**New Topics section**

**Use the title to generate the file name**

This option (and its sub-options) pertain to the rules that will be used to generate file names in the New DITA File dialog box (on page 2961). Select this option to use the text entered in the **Title** field to automatically generate a file name (the generated name can be seen in the **Save as** field). By default, the generated name will replace spaces with underscores (_), all illegal characters will be removed, and all upper case characters changed to lower case, but you can use the sub-options to change this.
Replace non-alphanumeric characters with

If selected, the file name generation mechanism will replace all non-alphanumeric characters in the title with the character entered in this option.

**Lower case only**

If selected, the file name generation mechanism will only use lower case letters.

**Use camel case**

If selected, the file name generation mechanism will convert the title to a file name using the camel case convention where the first word starts with a lower case letter and all subsequent words begin with upper case (for example, `myFileName`).

**Upper case first letter**

Select this option if you want the file name generation mechanism to convert the title to a file name using the camel case convention but with an upper case letter for the first word (for example, `MyFileName`).

**Use the file name as the value of the root ID attribute**

If selected, when creating a new topic, the file name (as seen in the `Save as` field but without the file extension) will be used as the value of the root `@id` attribute for the new topic.

**Inserting Links section**

**Always set values for the following attributes**

Allows you to specify that when a link reference is inserted (using actions in the Link drop-down menu), the values for certain attributes will always be automatically populated with a detected value (based on the specifications), even if it is the same as the default value. You can choose to always populate the values for the following attributes:

- **Format** - If selected, the `@format` attribute will always be automatically populated with a detected value.
- **Scope** - If selected, the `@scope` attribute will always be automatically populated with a detected value.
- **Type** - If selected, the `@type` attribute will always be automatically populated with a detected value.

**Use "." instead of the ID of the parent topic (DITA 1.3)**

When addressing a non-topic element within the topic that contains the URI reference, the URI reference can use an abbreviated fragment-identifier syntax that replaces the topic ID with `"." (\.elementId). For more information, see [https://www.oxygenxml.com/dita/1.3/specs/index.html#archSpec/base/uri-based-addressing.html](https://www.oxygenxml.com/dita/1.3/specs/index.html#archSpec/base/uri-based-addressing.html).
DITA Publishing Preferences

To access the DITA Publishing preferences page, open the Preferences dialog box (Options > Preferences) (on page 124) and go to DITA > Publishing. You can also open this page by clicking the Configure Publishing Templates Gallery link in the Templates tab of the transformation scenario dialog box for WebHelp Responsive transformations.

You can use this preferences page to specify additional directories where custom publishing templates are stored. The templates stored in these directories will appear in the preview pane in the Templates tab of the transformation scenario dialog box, along with all the built-in templates.

DITA Logging Preferences

To access the DITA Logging preferences page, open the Preferences dialog box (Options > Preferences) (on page 124) and go to DITA > Logging. This preferences page includes the following sections and options:

Show console output

Allows you to specify when to display the console output log in the message panel at the bottom of the editor. The following options are available:

- **When build fails** - Displays the console output log only if the build fails.
- **Always** - Displays the console output log, regardless of whether or not the build fails.

Show the following types of messages in a new tab

This section allows you to specify which types of messages will be displayed in separate tabs in the message panel at the bottom of the editor if a DITA transformation results in errors or warnings. You can choose whether or not to display the following types of messages:

- DITA-OT errors
- DITA-OT warnings
- DITA-OT info
- FOP errors
- FOP warnings
- FOP info
- XSLT problems

Markdown Preferences

The Markdown preferences page makes it possible to validate Markdown documents with Schematron. To access the page, open the Preferences dialog box (Options > Preferences) (on page 124) and go to Markdown. This preferences page includes the following options:

Validate converted HTML content

If selected, converted HTML content will be validated using the Schematron file specified in this option.
Validate converted DITA content

If selected, converted DITA content will be validated using the Schematron file specified in this option.

Note:
It is also possible to create a Schematron association for Markdown documents by adding a catalog mapping (on page 826) for one of the following URIs:

- http://www.oxygenxml.com/schematron/validation/markdown-as-dita

The catalog mapping is a fallback in case the validation is disabled in this preferences page or the path to the Schematron is empty. The associations configured in this preferences page take precedence.

Data Sources Preferences

To configure the Data Sources preferences, open the Preferences dialog box (Options > Preferences) (on page 124) and go to Data Sources. This preferences page allows you to configure data sources and connections to relational and native XML databases. For a list of drivers that are available for the major database servers, see Download Links for Database Drivers (on page 281).

Connection Wizards Section

Create eXist-db XML connection

Click this link to open the dedicated Create eXist-db XML connection dialog box (on page 2045) that provides a quick way to create an eXist connection.

Data Sources Section

This section allows you to add and configure data sources.

Figure 31. Data Sources Preferences Panel

The following buttons are available at the bottom of the Data Sources panel:

- New
Opens the **Data Sources Drivers** dialog box that allows you to configure a new database driver.

**Figure 32. Data Sources Drivers Dialog Box**

The following options are available in the **Data Source Drivers** dialog box:

- **Name** - The name of the new data source driver that will be used for creating connections to the database.
- **Type** - Selects the data source type from the supported driver types.
- **Help button** - Opens the User Manual at the list of the sections ([on page 281](#)) where the configuration of supported data sources is explained and the URLs for downloading the database drivers are specified.
- **Driver files (JAR, ZIP)** - Lists download links for database drivers ([on page 281](#)) that are necessary for accessing databases in Oxygen XML Editor.
- **Add Files** - Adds the driver class library.
- **Add Recursively** - Adds driver files recursively.
- **Remove** - Removes the selected driver class library from the list.
- **Detect** - Detects driver file candidates.
- **Stop** - Stops the detection of the driver candidates.
- **Driver class** - Specifies the driver class for the data source driver.
Opens the **Data Sources Drivers** dialog box for editing the selected driver. See above the specifications for the **Data Sources Drivers** dialog box. To edit a data source, there must be no connections using that data source driver.

**Duplicate**

Creates a copy of the selected data source.

**Delete**

Deletes the selected driver. To delete a data source, there must be no connections using that data source driver.

**Connections Section**

This section allows you to add and configure data source connections.

---

**Figure 33. Connections Preferences Panel**

<table>
<thead>
<tr>
<th>Connections</th>
<th>URL</th>
</tr>
</thead>
<tbody>
<tr>
<td>MySQL</td>
<td>jdbc:mysql://10.0.0.16:3306/test</td>
</tr>
<tr>
<td>Documentum</td>
<td><a href="http://10.0.0.150:9080">http://10.0.0.150:9080</a></td>
</tr>
</tbody>
</table>

The following buttons and options are available at the bottom of the **Connections** panel:

**New**

Opens the **Connection** dialog box that allows you to configure a new database connection.
The following options are available in the **Connection** dialog box:

- **Name** - The name of the new connection that will be used in transformation scenarios and validation scenarios.
- **Data Source** - Allows selecting a data source defined in the **Data Source Drivers** dialog box.

Depending upon the selected data source, you can set some of the following parameters in the **Connection details** area:

- **URL** - The URL for connecting to the database server.
- **User** - The user name for connecting to the database server.
- **Password** - The password of the specified user name.
- **Host** - The host address of the server.
- **Port** - The port where the server accepts the connection.
- **XML DB URI** - The database URI.
- **Database** - The initial database name.
- **Collection** - One of the available collections for the specified data source.
- **Environment home directory** - Specifies the home directory (only for a Berkeley database).
- **Verbosity** - Sets the verbosity level for output messages (only for a Berkeley database).
- **Use a secure HTTPS connection (SSL)** - Allows you to establish a secure connection to an eXist database through the SSL protocol.
Edit

Opens the Connection dialog box, allowing you to edit the selected connection. See above the specifications for the Connection dialog box.

Duplicate

Creates a copy of the selected connection.

Delete

Deletes the selected connection.

Move Up

Moves the selected connection up one row in the list.

Move Down

Moves the selected connection down one row in the list.

Limit the number of cells

For performance issues, you can set the maximum number of cells that will be displayed in the Table Explorer view (on page 2017) for a database table. Leave this field empty if you want the entire content of the table to be displayed. By default, this field is set to 2000. If a table that has more cells than the value set here is displayed in the Table Explorer view (on page 2017), a warning dialog box will inform you that the table is only partially shown.

Maximum number of children for container nodes

In Oracle XML, a container can hold millions of resources. If the node corresponding to such a container in the Data Source Explorer view (on page 2015) would display all the contained resources at the same time, the performance of the view would be very slow. To prevent this, only a limited number of the contained resources is displayed as child nodes of the container node. You can navigate to other contained resources from the same container by using the Up and Down buttons in the Data Source Explorer view (on page 2015). This limited number is set in the field. The default value is 200 nodes.

Table Filters Preferences

The Table Filters preferences page allows you to choose the types of tables to be shown in the Data Source Explorer view (on page 2015). To open this preferences page, open the Preferences dialog box (Options > Preferences) (on page 124) and go to Data Sources > Table Filters.

You can choose to display the following types of tables:

- Alias
- Global Temporary
- Local Temporary
- Synonym
- System Table
Download Links for Database Drivers

For a list of major relational databases and the drivers that are available for them, see https://www.oxygenxml.com/database_drivers.html.

In addition, the following is a list of other popular databases along with instructions for getting the drivers that are necessary to access the databases in Oxygen XML Editor:

- **IBM DB2 Pure XML database** - Go to the IBM website and in the DB2 Clients and Development Tools category select the DB2 Driver for JDBC and SQLJ download link. Fill out the download form and download the zip file. Unzip the file and use the db2jcc.jar and db2jcc_license_cu.jar files in Oxygen XML Editor for configuring a DB2 data source (on page 2022).

- **eXist database** - Copy the jar files from the eXist database install directory to the Oxygen XML Editor install directory as described in the procedure for configuring an eXist data source (on page 2046).

- **MarkLogic database** - Download the MarkLogic driver from MarkLogic Community site.

- **Oracle 11g database** - Go to http://www.oracle.com/technetwork/database/enterprise-edition/jdbc-112010-090769.html and download the Oracle 11g JDBC driver called ojdbc6.jar.

- **PostgreSQL database** - Go to http://jdbc.postgresql.org/download.html and download the PostgreSQL JDBC driver specific for your server version.

- **Berkeley DB XML database (Deprecated)** - Copy the jar files from the Berkeley database install directory into the Oxygen XML Editor install directory as described in the procedure for configuring a Berkeley DB data source (on page 2038).

- **Microsoft SQL Server 2005/2008 database (Deprecated)** - Download the appropriate MS SQL JDBC driver from the Microsoft website.

SVN Preferences

To configure the options for the SVN client tool, open the Preferences dialog box (Options > Preferences) (on page 124) and go to SVN. Some other preferences for the embedded SVN client tool can be set in the global files called config and servers. These files contain parameters that act as defaults applied to all the SVN client tools that are used by the same user on their computer login account. To open these files for editing, launch the embedded SVN client tool (Tools > SVN Client) and select Global Runtime Configuration > Edit 'config' file or Global Runtime Configuration > Edit 'servers' file from the SVN client Options menu.
The following SVN options can be configured in this preferences page:

**Enable symbolic link support (available only on macOS and Linux)**

Apache Subversion™ can put a symbolic link under version control, via the usual SVN add command. The Subversion repository has no internal concept of a symbolic link. It stores a versioned symbolic link as an ordinary file with a `svn:special` property attached. On Unix/Linux, the SVN client sees the property and translates the file into a symbolic link in the working copy. If the symbolic link support is disabled, the versioned symbolic links appear as a text file instead of symbolic link.

**Note:**

Windows file systems have no symbolic links, so a Windows client will not do any such translation and the object appears as a normal file.

**Important:**

It is recommended to disable symbolic links support if you do not have versioned symbolic links in your repository, since the SVN operations will work faster. However, you should not disable this option when you do have versioned symbolic links in repository. In that case a workaround would be to reference the working copy by its real path, instead of a path that includes a symbolic link.

**Allow unversioned obstructions**

Controls how to handle a situation where working copy resources are ignored / unversioned when performing an update operation and incoming files (from the repository) with the same name and location intersect with those being ignored / unversioned. If the option is selected, the incoming items will become BASE revisions of the ones already present in the working copy, and those present will be made versioned resources and will be marked as modified (exactly as if the user first made the update operation and then modified the files). If the option is not selected,
the update operation will fail when encountering files in this situation, possibly leaving other files
not updated. By default, this option is selected.

**Use unsafe copy operations**

Sometimes when the working copy is accessed through Samba and the SVN client cannot make
a safe copy of the committed file due to a delay in getting a write permission, the result is that
the committed file will be saved with zero length (the content is removed) and an error will be
reported. In this case, this option should be selected so that the SVN client does not try to make
the safe copy.

**Results Console**

Specifies the maximum number of lines displayed in the **Console** view. The default value is 1000.

**Annotations View**

Sets the color used in the editor panel for highlighting all the changes contributed to a resource
by the revision selected in the **Annotations view** (on page 2835).

**Revision Graph**

Enables caching for the action of computing a revision graph. When a new revision graph is
requested, one of the caches from the previous actions may be used that will avoid running the
whole query again on the SVN server. If a cache is used, it will finish the action much faster.

**Working Copy Preferences**

To configure the **Working Copy** preferences, open the **Preferences** dialog box (Options > Preferences) (on
page 124) and go to **SVN > Working Copy**. The options in this preferences page are specific to SVN working
copies and they include the following:

**Working copies location**

Allows you to define a location where you keep your working copies. This location is
automatically suggested when you checkout a new working copy.

**Working copy administrative directory**

Allows you to customize the directory name where the SVN entries are kept for each directory in
the working copy.

**When loading an old format working copy**

You can instruct the SVN client to do one of the following:
• **Always ask** - You are notified when such a working copy is used and you are allowed to choose what action to be taken (whether or not to upgrade the format of the current working copy).

• **Never upgrade** - Older format working copies are left untouched. No attempt to upgrade the format is made.

**Note:**
SVN 1.6 and older working copies still need to be upgraded before loading them.

**Enable working copy caching**
If selected, the content of the working copies is cached for refresh operations.

**Automatically refresh the working copy**
If selected, the working copy is refreshed from cache. Only the new changes (modifications with a date/time that follows the last refresh operation) are refreshed from disk. This option is not selected by default.

**Allow moving/renaming mixed revision directories**
If selected, Oxygen XML Editor will allow you to move or rename a directory even if its child items have a different revision. Otherwise, an error message is displayed when there are multiple revisions to avoid unnecessary conflicts. It is recommended to leave this option deselected and to **Update** the subtree to a single revision before moving or renaming it.

**When synchronizing with repository**
The action that will be executed automatically after the **Synchronize** action. The possible actions are:

• **Always switch to 'Modified' mode** - The **Synchronize** action is followed automatically by a switch to **Modified** mode of **Working Copy** view, if **All Files** mode is currently selected.

• **Never switch to 'Modified' mode** - Keeps the currently selected view mode unchanged.

• **Always ask** - The user is always asked if they want to switch to **Modified** mode.

**Application global ignores**
Allows you to set file patterns that may include the * and ? wildcards for unversioned files and folders that must be ignored when displaying the working copy resources in the **Working Copy view (on page 2813)**. These patterns are case-sensitive. For example, *.txt matches file.txt, but does not match file.TXT.

**Diff Preferences**
To configure the SVN Diff options, open the **Preferences** dialog box (**Options > Preferences**) (on page 124) and go to **Diff**.

The following option is available:
Compare With External Application

Specifies an external application to be launched for compare operations in the following cases:

- When two history revisions are compared.
- When the working copy file is compared with a history revision.
- When a conflict is edited.

The parameters $\text{\texttt{firstFile}}$ and $\text{\texttt{secondFile}}$ specify the positions of the two compared files in the command line for the external diff application. The parameter $\text{\texttt{ancestorFile}}$ specifies the common ancestor (that is, the BASE revision of a file) in a three-way comparison. The working copy version of a file is compared with the repository version, with the BASE revision (the latest revision read from the repository by an Update or Synchronize operation) being the common ancestor of these two compared versions.

**Important:**
If the path to the external compare application includes spaces (or any of the subsequent options or arguments), then each of these paths or tokens must be double-quoted for the Oxygen XML Editor to correctly parse and identify them. For example,

```
C:\Program Files\compareDir\app name.exe
```

must be written as

```
"C:\Program Files\compareDir\app name.exe"
```

Messages Preferences

The Messages preferences page allows you to disable certain warning messages that may appear in the application. To configure these options, open the Preferences dialog box (Options > Preferences) (on page 124) and go to SVN > Messages.

This preferences page allows you to disable the following warning messages:

**Show confirmation dialog when using the "Update All" action**

Allows you to avoid performing accidental update operations by requesting you to confirm them before execution.

**Show confirmation dialog for drag and drop actions in Working Copy**

This option avoids doing a drag and drop when you just want to select multiple files in the Working Copy view.

**Show warning dialog when editing conflicts**

When the Edit Conflicts action is executed, a warning dialog box notifies you that the action overwrites the conflicted version of the file created by an update operation. The conflicted file is overwritten with the version of the same file that existed in the working copy before the update operation and then proceeds with the visual editing of the conflicting file (on page 2755).

**Show warning dialog when "svn:externals" definitions are ignored**
A warning dialog box is displayed when "svn:externals" definitions are ignored before performing any operation that updates resources of the working copy (such as Update and Override and Update).

**Diff Preferences**

The Diff Preferences Page has sub-pages for configuring File Comparisons and Directory Comparisons.

**Files Comparison Preferences**

To configure the Files Comparison options, open the Preferences dialog box (Options > Preferences) *(on page 124)* and go to Diff > Files Comparison.

This preferences page allows you to configure the following options:

**Enable file comparison in Author mode**

If selected, a visual Author mode is available in the file comparison tool. It displays the files in a visual mode similar to the Author editing mode in Oxygen XML Editor/Author. This visual mode is available when both compared files are detected as being XML and from a recognized document type.

**Ignore Whitespaces (Not applicable for the visual Author comparison mode)**

If selected, before performing the comparison, the application normalizes the content (collapses any sequence of whitespace characters into a single space) and trims its leading and trailing whitespaces.

**Note:**

If the Ignore Whitespaces checkbox is selected, comparing the \texttt{a b} sequence with \texttt{a b}, Oxygen XML Editor finds no differences, because after normalization, all whitespaces from the first sequence are collapsed into a single space character. However, when comparing \texttt{a b} with \texttt{ab} (no whitespace between \texttt{a} and \texttt{b}), Oxygen XML Editor signals a difference.

**Two-Way Diff section**

**Default algorithm**

The default algorithm used for comparing two files. The following options are available:

- **Auto** - Selects the most appropriate algorithm, based on the compared content and its size (selected by default).
- **Characters** - Computes the differences at character level, meaning that it compares two files or fragments looking for identical characters. This algorithm is not available when the file comparison is in Author comparison mode.
• **Words** - Computes the differences at word level, meaning that it compares two files or fragments looking for identical words. This algorithm is not available when the file comparison is in **Author** comparison mode.

• **Lines** - Computes the differences at line level, meaning that it compares two files or fragments looking for identical lines of text. This algorithm is not available when the file comparison is in **Author** comparison mode.

• **Syntax Aware** - Computes differences for the file types or fragments known by Oxygen XML Editor, taking the syntax (the specific types of tokens) into consideration. This algorithm is not available when the file comparison is in **Author** comparison mode.

• **XML Fast** - Comparison that works well on large files or fragments, but it is less precise than **XML Accurate**.

• **XML Accurate** - Comparison that is more precise than **XML Fast**, at the expense of speed. It compares two XML files or fragments looking for identical XML nodes.

**Algorithm strength**

Controls the amount of resources allocated to the application to perform the comparison. The algorithm stops searching more differences when reaches the maximum allowed resources. A dialog box is displayed when this limit is reached and partial results are displayed. Four settings are available: **Low**, **Medium** (default), **High** and **Very High**.

**Three-Way Diff section**

**Default algorithm**

The default algorithm used for performing a three-way comparison. The following options are available:

• **Auto** - Selects the most appropriate algorithm, based on the compared content and its size (selected by default).

• **Lines** - Computes the differences at line level, meaning that it compares two files or fragments looking for identical lines of text. This algorithm is not available when the file comparison is in **Author** comparison mode.

• **XML Fast** - Comparison that works well on large files or fragments, but it is less precise than **XML Accurate**.

• **XML Accurate** - Comparison that is more precise than **XML Fast**, at the expense of speed. It compares two XML files or fragments looking for identical XML nodes.

**Algorithm strength**

Controls the amount of resources allocated to the application to perform the comparison. The algorithm stops searching more differences when reaches
the maximum allowed resources. A dialog box is displayed when this limit is reached and partial results are displayed. Four settings are available: Low, Medium (default), High and Very High.

**Show pseudo conflicts**

Specifies whether or not the file comparison displays pseudo-conflicts. A pseudo-conflict occurs when two users make the same change (for example, when they both add or remove the same line of code).

**XML Diff section**

**Ignore (Not applicable for the visual Author comparison mode)**

Allows you to specify the types of XML nodes that will be ignored in the file comparison for the XML Fast and XML Accurate algorithms. You can choose to ignore Processing Instructions, Comments, CDATA, DOCTYPE, Text, Namespaces, Prefixes, Namespace declarations, and Attribute order.

**Ignore nodes by XPath (Not applicable for the visual Author comparison mode)**

If selected, you can enter an XPath expression (on page 1999) to ignore certain nodes from the comparison. It will be processed as XPath version 2.0. The XPath expression specified in this option is used as the default ignore instructions only when the application is started. If you enter an XPath expression in the similar option on the Diff Files toolbar, that expression will be used instead.

**Merge adjacent differences (Not applicable for the visual Author comparison mode)**

If selected, the application considers two adjacent differences as one when the differences are painted in the side-by-side editors. If not selected, every difference is represented separately.

**Mark end tags as different for modified elements (Not applicable for the visual Author comparison mode)**

If selected, end tags of modified elements are also presented as differences. Otherwise, only the start tags are presented as differences.

**Ignore expansion state for empty elements (Not applicable for the visual Author comparison mode)**

If selected, empty elements in both expansion states are considered matched (that is <element/> and <element></element> are considered equal).

**Appearance Preferences**

To configure the appearance options for the Files Comparison tool, open the Preferences dialog box (Options > Preferences) (on page 124) and go to Diff > Files Comparison > Appearance. This preferences page offers the following options:

**Line wrap**
Wraps the lines presented in the two diff panels at the right margin of each panel, so no horizontal scrollbar is necessary.

**Incoming color**

Specifies the color used on the vertical bar for incoming changes.

**Outgoing color**

Specifies the color used on the vertical bar for outgoing changes.

**Conflict color**

Specifies the color used on the vertical bar for conflicts between the compared files.

### Directories Comparison Preferences

To configure the Directories Comparison preferences, open the Preferences dialog box (Options > Preferences) (on page 124) and go to Diff > Directories Comparison.

![Diff Preferences Page](image)

You can specify the following options for the directories comparison tool:

**Compare files by**

Controls the method used for comparing two files:

- **Content** - The file content is compared using the current diff algorithm (on page 286). This option is applied for a pair of files only if that file type is associated with a built-in editor type (either associated by default or associated by the user when prompted to do so on opening a file of that type for the first time).

You can use the Configure content comparison link to open the Files Comparison preferences page (on page 286) where you can configure options for comparing files. However, the Ignore nodes by XPath option is ignored when using the Compare Directories tool.
• Binary Compare - The files are compared at byte level.
• Timestamp (last modified date / time) - The files are compared only by their last modified timestamp.

Look in archives
If selected, known archive types (on page 291) are considered directories and their content is compared just like regular files.

Navigation
This options control the behavior of the differences traversal actions (Go to previous modification, Go to next modification) when the first or last difference in a file is reached:

• Ask what to do next - A dialog box is displayed asking you to confirm that you want the application to display modifications from the previous or next file.
• Go to the next/previous file - The application opens the next or previous file without waiting for your confirmation.
• Do nothing - No further action is taken.

Appearance Preferences
To configure the appearance options for the Directories Comparison tool, open the Preferences dialog box (Options > Preferences) (on page 124) and go to Diff > Directories Comparison > Appearance.

Figure 37. Diff Appearance Preferences Panel

- Added/Deleted - Color used for marking added or deleted files and folders.
- Modified - Color used for marking modified files.

Archive Preferences
To configure Archive options, open the Preferences dialog box (Options > Preferences) (on page 124) and go to Archive.

The following options are available in the Archive preferences page:

  Archive backup options

  Controls if the application makes backup copies of the modified archives. The following options are available:
• **Always create backup copies of modified archives** - When you modify an archive, its content is backed up.

• **Never create backup copies of modified archives** - No backup copy is created.

• **Ask for each archive once per session** - Once per application session for each modified archive, user confirmation is required to create the backup. This is the default setting.

**Note:**
Backup files have the name `originalArchiveFileName.bak` and are located in the same folder as the original archive.

### Archive types

This table contains all known archive extensions mapped to known archive formats. Each row maps a list of extensions to an archive type supported in Oxygen XML Editor. You can use the **Edit** button at the bottom of the table to edit an existing mapping or the **New** button to create a new one and associate your own list of extensions to an archive format.

**Figure 38. Edit Archive Extension Mappings**

![Edit Archive Extension Mappings](image)

**Important:**
You have to restart Oxygen XML Editor after removing an extension from the table for that extension to not be recognized as an archive extension.

**Store Unicode file names in Zip archives**

Use this option when you archive files that contain international (non-English) characters in file names or file comments. If this option is selected and an archive is modified in any way, UTF-8 characters are used in the names of all files in the archive.

### Plugins Preferences

You can add **plugins** that extend the functionality of Oxygen XML Editor. The **plugins** are shipped as separate packages. To check for new **plugins**, go to [http://www.oxygenxml.com/oxygen_sdk.html](http://www.oxygenxml.com/oxygen_sdk.html).
A *plugin* consists of a separate sub-folder in the `Plugins` folder of the Oxygen XML Editor installation folder (for example, `[OXYGEN_INSTALL_DIR]/plugins/myPlugin`). This sub-folder must contain a valid plugin.xml file in accordance with the plugin.dtd file located in the `Plugins` folder.

Oxygen XML Editor automatically detects and loads *plugins* installed correctly in the `Plugins` folder and displays them in the `Plugins` preferences page. To configure *plugins*, open the `Preferences` dialog box (*Options > Preferences*) *(on page 124)* and go to `Plugins`.

You can use the checkboxes in front of each *plugin* to enable or disable them. To display the properties of a *plugin* in the second section of the `Plugins` preferences page, click the name of the *plugin*.

Also, you can install a *plugin* as an add-on. For further details about this, go to *Deploying Add-ons* *(on page 2445)*

### External Tools Preferences

A command-line tool can be started in the Oxygen XML Editor user interface as if from the command line of the operating system shell. The `External Tools` preferences page allows you to add and configure these external tools that could be used while working with Oxygen XML Editor. To access this preferences page, open the `Preferences` dialog box (*Options > Preferences*) *(on page 124)* and go to `External Tools` *(or select `Configure` from the `Tools > External Tools` menu)*.

This preferences page presents a list of the external tools that have been configured. You can use the buttons at the bottom of the page to configure the items in the list. Once a tool has been configured, you can open it by selecting it from the `Tools > External Tools` menu or from the `External Tools` drop-down menu on the toolbar *(the `Tools` toolbar needs to be selected in the `Configure Toolbars` dialog box *(on page 365))*.

### How to Configure an External Tool

To configure an external tool in the `External Tools` preferences page, use any of the following buttons at the bottom of the page:

- **New** - Adds a new external tool to the list.
- **Edit** - Allows you to configure an existing external tool, selected from the list.
- **Duplicate** - Duplicates an existing external tool, selected from the list, to use as a template for configuring a similar tool.

Any of those three buttons opens the `External Tools` configuration dialog box.
This configuration dialog box includes the following options:

**Name**

The name of tool that will be displayed in the **Tools > External Tools** menu and in the **External Tools** drop-down menu on toolbar.

**Description**

A description of the tool displayed as a tooltip where the tool name is used.

**Working directory**

The directory that the external tool will use to store intermediate and final results. You can specify the path by using the text field, the **Insert Editor Variables** button, or the **Browse** button. You can use one of the following editor variables: **${cfd}** (on page 329), **${pd}** (on page 331), **${oxygenInstallDir}** (on page 331), **${homeDir}** (on page 331), **${system(var.name)}** (on page 332), **${date(pattern)}** (on page 329), **${xpath_eval(expression)}** (on page 332).

**Command line**

The command line that will start the external tool. You can specify the path by using the text field, the **Insert Editor Variables** button, or the **Browse** button. You can use one of the following editor variables: **${homeDir}** (on page 331), **${home}** (on page 331), **${cfn}** (on page 329), **${cfne}** (on page 329), **${cf}** (on page 329), **${currentFileURL}** (on page 329), **${cfdu}** (on page 329), **${tsf}** (on page 332), **${pd}** (on page 331).
Show output messages

When this option is selected, all the messages emitted by the external tool are displayed in the Results view (on page 548). When this option is not selected, only the error messages are displayed. You can also choose the output encoding and content type:

- **Output encoding** - The encoding of the output stream of the external tool that will be used by Oxygen XML Editor to read the output of the tool.

- **Output content type** - A list of predefined content type formats that instruct Oxygen XML Editor how to display the generated output. For example, setting the **Output content type** to text/xml enables the syntax coloring of XML output.

**Error Encoding**

The encoding of the error stream of the external tool that will be used by Oxygen XML Editor to read the error stream.

**Shortcut key**

You can choose a keyboard shortcut that can be used to launch the external tool.

**Menu Shortcut Keys Preferences**

You can use the Menu Shortcut Keys preferences page to configure shortcut keys for the actions available in Oxygen XML Editor. The shortcuts assigned to actions are displayed in a table in this preference page. To access the full list of shortcut keys, open the Preferences dialog box (Options > Preferences) (on page 124) and go to Menu Shortcut Keys (or simply go to Options > Menu Shortcut Keys).

For a list of the most commonly used shortcuts, see Frequently Used Shortcut Keys (on page 55).
The **Menu Shortcut Keys** preferences page also contains the shortcuts that you define at document type level (on page 150).

**Note:**
A shortcut defined at document type level overwrites a default shortcut.

Furthermore, the shortcuts table also contains entries for actions that show side-views contributed by plugins.

To find a specific action, you can use the filter text field to search through any of the columns in the table. You can also press shortcut key combinations on your keyboard to filter the list and click on a column header to sort that column.

The table includes the following columns or options:

- **Description** - A short description of the action.
- **Category** - A classification of the actions in categories for easier management and more flexibility in assigning multiple keys for the same action.
• **Shortcut key** - The combination of keyboard keys that can be used to launch the action. To add or change a shortcut key, you can either double-click a row or select the row and click the **Edit** button.

• **'Home' and 'End' keys are applied at line level** (available on macOS only) - Controls the way the HOME and END keys are interpreted. If selected, the default behavior of these keys is overridden and the cursor only moves on the current line.

### How to Assign a Shortcut Key or Edit an Existing Shortcut

To assign a shortcut key to an action or edit an existing shortcut configuration, follow these steps:

1. Select the action in the table.

2. Click the **Edit** button.

   **Step Result:** The **Shortcut key** configuration dialog box is displayed.

   ![Shortcut Key Configuration Dialog Box](image)

   3. Press the desired shortcut keys on your keyboard.

   4. If you need the shortcut to work on multiple platforms, select the **Enable platform-independent shortcut keys** option. In this case, the following modifiers are used:

      - M1 represents the **Command** key on macOS, and the **Ctrl** key on other platforms.
      - M2 represents the **Shift** key.
      - M3 represents the **Option** key on macOS, and the **Alt** key on other platforms.
      - M4 represents the **Ctrl** key on macOS, and is undefined on other platforms.

   5. Click **OK** to save your configuration.

### Troubleshooting:

If you encounter problems with keyboard shortcuts not working as expected, see **Keyboard Shortcuts Result in Unexpected Behavior (on page 2876)** or **Keyboard Shortcuts Do Not Work At All (on page 2876)**.

### Related information

- [Frequently Used Shortcut Keys](on page 55)
File Types Preferences

Oxygen XML Editor offers built-in editing support for a wide variety of file types, but you can also add new file extensions and associate them with whatever editor type fits your needs. The associations set here between a file extension and the type of editor will determine which editor will be opened for editing purposes when that type of file is created or opened.

To configure the File Types options, open the Preferences dialog box (Options > Preferences) (on page 124) and go to File Types.

![Figure 42. File Types Preferences Page](image)

The table contains the following columns:

- **Extension** - The extensions of the files that will be associated with an editor type.
- **Editor** - The type of editor which the extensions will be associated with. Some editors provide easy access to frequent operations via toolbars (XML editor, XSL editor, DTD editor) while others provide just a syntax highlight scheme (Java editor, SQL editor, Shell editor).

If the editor set here is not one of the XML editors (XML editor, XSL editor, XSD editor, RNG editor, WSDL editor) then the encoding set in the Encoding for non-XML files option (on page 168) is used for opening and saving a file of this type.

The files that match the Ant build patterns will be associated with the Ant editor.
The files that match the **Binary file patterns** patterns are handled as binary and opened in the associated system application. Also, they are excluded from the following actions available in the **Project view (on page 402)**: File/Replace in Files, Check Spelling in Files, Validate.

### Open/Find Resource Preferences Page

You can configure various options that pertain to the **Open/Find Resource dialog box (on page 426)** and **Open/Find Resource view (on page 422)**. To access these options, open the **Preferences dialog box (Options > Preferences) (on page 124)** and go to **Open/Find Resource**.

The following options are available in this **Open/Find Resource** preferences page:

- **Refresh index when opening a map in DITA Maps Manager**
  - If selected, DITA maps that are opened in the DITA Maps Manager will automatically be re-indexed.

- **Limit search results to**
  - Specifies the maximum number of results that are displayed in the **Open/Find Resource dialog box/view (on page 426)**.

- **Enable searching in content**
  - This option is selected by default and it allows you to use the **Open/Find Resource dialog box/view (on page 426)** to search in content or reviews, as well as in file paths. If this option is not selected, you can only use the **Open/Find Resource** feature to search in file paths.

- **Content search scope section**
  - **Ignore content of these files**
    - Allows you to select specific directories, files, or file types that are ignored when you perform a search. For example, `*.txt` ignores all the .txt files, `*/topics/*` ignores all the files from the topics directory, regardless of their depth, and `file:/C:/tmp/*` ignores everything from the tmp directory.

  - **Include the contents of these binary files**
    - Results from a search in the **Open/Find Resource dialog box/view (on page 426)** will be returned as a PDF if the PDF contains the searched text. If you want to disable this feature, delete the contents of the text field.

  - **Index the content of remote resources**
    - Controls the indexing of resources that are not local. For example, the resources referenced in a **DITA map (on page 3218)** opened from a remote server (from a CMS or from a WebDAV location) are not indexed by default. To index the content of these resources, select this option.

**Note:**

Selecting this option may lead to delays when the indexing is computed.
Content search options section

Content language

Use this option to specify a language for the search engine to use for the current document. This is helpful if you have multiple languages within the content of a document. The search engine will use a set of *stop words* and analyzers tuned specifically for that specific language. By default, it is mapped to the *UI language specified in the Global preferences page (on page 126)*. Therefore, you need to change this option only if the language of the text you want to perform the search in differs from the UI language.

**Tip:**

If you select *<Generic language (no stemming)>* from the drop-down list, no word stemming is performed when creating the index. This might be useful if your content has many technical terms that should be indexed as they are.

Stop words

A list of *stop words* that will be filtered out of the search processing. The list is automatically populated based upon the specified **Content language**, but you can add or remove words from the list.

When searching in content, return

This option specifies how matches are returned when doing searches in content. You can choose between two options:

- **Exact matches** - The search results match the exact whole words that you enter in the search field of the *Open/Find Resource* dialog box/view.
- **Prefix matches** (default) - The search results match documents that contain words starting with the search terms. For instance, searching for "pref page" will also find documents containing "preference page".

Automatically join search terms using:

Allows you to select the default boolean operator that Oxygen XML Editor applies when you perform a search. For example, if the AND operator is selected and you search for "car assembly", the matches must contain both of the words. If you choose OR, the matches must contain one of the selected search terms and results that contain both words are promoted to the top of the list.

Enable XML-aware searching

When selected, you can perform **XML-specific searches (on page 429)** for XML elements and attributes.
Note:
Selecting this option may slow down the indexing of your documents and increase the index size on the disk.

Index files with size less than (KB)
Since indexing can be slowed down when the Enable XML-aware searching option (on page 299) is active, you can use this option to set a maximum file size to be indexed.

Stop Words
A list of comma-separated stop words, meaning that the words added in this list are filtered out prior to processing a search query.

Related information
Open/Find Resource View (on page 422)
Open/Find Resource Dialog Box (on page 426)

Custom Editor Variables Preferences
An editor variable (on page 323) is useful for making a transformation scenario, validation scenario, or other tool independent of its file path. An editor variable is specified as a parameter in a transformation scenario, validation scenario, or command line of an external tool. Such a variable is defined by a name, a string value, and a text description. A custom editor variable is defined by the user and can be used in the same expressions as the built-in editor variables (on page 323).

Custom editor variables are created and configured in the Custom Editor Variables preferences page. To access this page, open the Preferences dialog box (Options > Preferences) (on page 124) and go to Custom Editor Variables.

This preferences page displays a table of all the custom editor variables that have been defined. The table includes three columns for the editor variable Name, its Value, and its Description. The create a new variable, click the New button at the bottom of the table and define your custom editor variable in the subsequent dialog box. To edit an existing custom editor variable, click the Edit button and configure the variable in the subsequent dialog box. You can also use the Delete button to remove custom editor variables that are no longer needed.
Network Connection Settings Preferences

This section presents the options available in the **Network Connection Settings** preferences pages.

**Proxy Preferences**

Some networks use proxy servers to provide internet services to LAN clients. Therefore, clients behind the proxy may only connect to the Internet via the proxy service. If you are not sure if your computer is required to use a proxy server to connect to the Internet or you do not know the proxy parameters, consult your network administrator.

To configure the **Proxy** options, open the Preferences dialog box (**Options > Preferences**) (on page 124) and go to **Network Connection Settings > Proxy**. The following options are available:

**Proxy section**

Specifies how HTTP(S) connections go through the proxy server. You can choose between the following three options:

- **Direct connection**: HTTP(S) connections will go directly to the target host without going through a proxy server.
- **Use system settings** (default setting): HTTP(S) connections will go through the proxy server set in the operating system.

**Attention:**

The system settings for the proxy cannot be read correctly from the operating system on some Linux systems. The system settings option should work properly
on Gnome-based Linux systems, but it does not work on KDE-based ones as the Java virtual machine does not offer the necessary support yet.

• **Manual proxy configuration** - HTTP(S) connections will go through the proxy server specified in the Web Proxy (HTTP/HTTPS) section.

**Web Proxy (HTTP/HTTPS) section**

**Address**

The address of the proxy server used for manual configurations.

**Port**

The port of the proxy server used for manual configurations.

**No proxy for**

Specifies the hosts that the connections must not go through a proxy server. A host needs to be written as a fully qualified domain name (for example, myhost.example.com) or as a domain name (for example, example.com). Use a comma to separate multiple hosts.

**User**

The user name for authentication with the proxy server.

**Password**

The password for authentication with the proxy server.

**SOCKS Proxy section**

**Address**

The address of a SOCKS proxy that all connections will pass through. If this field is empty, the connections do not use a SOCKS proxy.

**Port**

The port of a SOCKS proxy that all connections will pass through.

**Using a Proxy Auto-Configuration Script (PAC)**

If you have set up the path to a Proxy auto-configuration script in your system, Oxygen XML Editor cannot detect this setting.

You can create a new folder ([OXYGEN_INSTALL_DIR]/lib/endorsed) where you should copy two additional Java libraries: deploy.jar and plugin.jar. These libraries can be found in the [OXYGEN_INSTALL_DIR]/jre/lib folder if the application came with a bundled Java VM (otherwise, in the Java VM installation used to run the application).
HTTP(S)/WebDAV Preferences

To set the HTTP(S)/WebDAV preferences, open the Preferences dialog box (Options > Preferences) (on page 124) and go to Network Connection Settings > HTTP(S)/WebDAV. The following options are available:

**Maximum number of simultaneous connections per host**

Defines the maximum number of simultaneous connections established by the application with a distinct host. Servers might consider multiple connections opened from the same source to be a Denial of Service attack. You can avoid that by lowering the value of this option.

*Note:*

The minimum value that can be set in this option is 5.

**Read Timeout (seconds)**

The period (in seconds) after which the application considers that an HTTP server is unreachable if it does not receive any response from that server.

**Enable HTTP 'Expect: 100-continue ' handshake (for HTTP/1.1 protocol)**

Activates *Expect: 100-Continue* handshake. The purpose of the *Expect: 100-Continue* handshake is to allow a client that is sending a request message with a request body to determine if the origin server is willing to accept the request (based on the request headers) before the client sends the request body. The use of the *Expect: 100-continue* handshake can result in noticeable performance improvement when working with databases. The *Expect: 100-continue* handshake should be used with caution, as it may cause problems with HTTP servers and proxies that do not support the HTTP/1.1 protocol.

**Use the 'Content-Type' header field to determine the content type**

When selected, Oxygen XML Editor tries to determine a resource type using the **Content-Type** header field. This header indicates the *Internet media type* of the message content, consisting of a type and subtype. For example:

```
Content-Type: text/xml
```

When unchecked, the resource type is determined by analyzing its extension. For example, a file ending in .xml is considered to be an XML file.

**Automatic retry on recoverable error**

When selected, if an HTTP error occurs when Oxygen XML Editor communicates with a server via HTTP (for example, sending or receiving a SOAP request to or from a Web services server) and the error is recoverable, Oxygen XML Editor tries to re-send the request to the server.

**Automatically accept a security certificate, even if invalid**

When selected, the HTTPS connections that Oxygen XML Editor attempts to establish with will accept all security certificates, even if they are invalid.
Important:
By accepting an invalid certificate, you accept (at your own risk) a potential security threat, since you cannot verify the integrity of the certificate’s issuer.

Lock WebDAV files on open
If selected, the files opened through WebDAV are locked on the server so that they cannot be edited by other users while the lock placed by the current user still exists on the server.

(S)FTP Preferences
To configure the (S)FTP options, open the Preferences dialog box (Options > Preferences) (on page 124) and go to Network Connection Settings > (S)FTP. You can customize the following options:

Encoding for FTP control connection
The encoding used to communicate with FTP servers: either ISO-8859-1 or UTF-8. If the server supports the UTF-8 encoding, Oxygen XML Editor will use it for communication. Otherwise, it will use ISO-8859-1. This section also includes a Show hidden files toggle option.

Public known hosts file
Specifies the file that contains the list of all SSH server host keys that you have determined are accurate. The default value is `${homeDir}/.ssh/known_hosts`.

Private key file
The path to the file that contains the private key used for the private key method of authentication of the secure FTP (SFTP) protocol. Only RSA private keys in PEM (Base64) and PPK (PuTTY) formats are supported. Other keys (such as OpenSSH) are not supported. The user / password method of authentication has precedence if it is used in the Open URL dialog box (on page 386).

Passphrase
The passphrase used for the private key method of authentication of the secure FTP (SFTP) protocol. The user / password method of authentication has precedence if it is used in the Open URL dialog box (on page 386).

Trusted Hosts Preferences
Oxygen XML Editor comes with a built-in firewall that controls the access to external resources. Anytime the application detects a request to connect to a remote resource, it checks to see if the URL belongs to a domain that has been identified as trusted. If not, a confirmation dialog box will be displayed where you can choose whether to allow or reject access to the remote connection.
You can configure the list of trusted hosts using the **Trusted Hosts** preferences page. It contains a list of domains that have been identified as trusted. You can add or remove domains from the list and Oxygen XML Editor will allow connections to the listed hosts without requesting user confirmation.

**Note:**
Connections defined in the [Data Sources preferences page](on page 276) or accepted by add-ons are also considered trusted.

To add or remove domains, open the **Preferences** dialog box ([Options > Preferences](on page 124)) and go to **Network Connection Settings > Trusted Hosts**. The following options are available:

- **New** - Allows you to manually add a new entry to the list of trusted hosts.

**Tip:**
You can specify a specific port at the end of the URL (for instance, `www.example.com:8080`). Otherwise, if no port is specified, connections will be allowed on all ports for the particular host.

- **Delete** - Allows you to remove an entry from the list of trusted hosts.

**SSH Preferences**

To configure the **SSH** options, open the **Preferences** dialog box ([Options > Preferences](on page 124)) and go to **Connection settings > SSH**. The following options are available:

**SSH**

Specifies the command line for an external SSH client that will be used when connecting to a SVN+SSH repository. Absolute paths are recommended for the SSH client executable and the file paths given as arguments (if any). Depending on the SSH client used and your SSH server configuration, you may need to specify the user name and/or private key/passphrase in the command line. You can also choose whether to use the **Default SVN user** (the same user name as the SSH client user) or **Prompt for a SVN user** for SVN repository operations whenever SVN
authentication is required. For example, on Windows the following command line uses the `plink.exe` tool as the external SSH client for connecting to the SVN repository with SVN+SSH:

```
C:\plink-install-folder\plink.exe -l username -pw password -ssh -batch host_name_or_IP_address_of_SVN_server
```

## XML Structure Outline Preferences

To configure options regarding the **Outline view** *(on page 539)*, open the **Preferences** dialog box *(Options > Preferences)* *(on page 124)* and go to XML Structure Outline. It contains the following options:

**Preferred attribute names for display**

The preferred attribute names when displaying the attributes of an element in the **Outline** view. If there is no preferred attribute name specified, the first attribute of an element is displayed.

**Enable outline drag and drop**

Drag and drop is disabled for the tree displayed in the **Outline** view only if there is a possibility to accidentally change the structure of the document by such operations.

## Views Preferences

The **Views** preferences page allows you to configure some options regarding certain views. To edit these options, open the **Preferences** dialog box *(Options > Preferences)* *(on page 124)* and go to Views.

The following options are available:

**Project view section**

**Enable drag-and-drop in Project view**

Enables drag and drop support in the **Project view** *(on page 402)*. It should be disabled only if there is a possibility of accidentally changing the structure of the project by drag and drop actions.

**Information view section**

**Maximum number of lines**

Specifies the maximum number of lines that can be written in the **Information view** *(on page 512)*.

**Elements view section**

**Show only allowed items**

If selected, when editing in **Author** mode, only the elements that are allowed at the current cursor position will be listed in the **Elements view** *(on page 633)*. If not selected, all elements allowed by the schema will be listed, even if they are already used.
Messages Preferences

The Messages preference page allows you to specify whether or not certain messages are displayed. To configure these options, open the Preferences dialog box (Options > Preferences) (on page 124) and go to Messages.

The following warning messages can be enabled or disabled:

**Show Java vendor warning at startup**

If this option is selected, Oxygen XML Editor displays a warning on startup if a non-recommended version of the Java virtual machine is being used.

**Show confirmation dialog when moving resources**

Specifies whether or not to display a confirmation dialog box when you move a resource in the Project view (on page 402), Data Source Explorer view (on page 2015), and Archive Browser (on page 2008). In the confirmation dialog box, there is an option to choose to not show this dialog box in the future. To reset that behavior, simply select Restore Defaults at the bottom of this preferences page.

**Show warning when adding resources already included in the project**

Specifies whether or not to display a dialog box that warns you if you try to add files that already exist in your project.

**Show warning for document size limit for bidirectional text, Asian languages, and other special characters**

Specifies whether or not to display a warning message when an open file that contains bidirectional characters is too large and bidirectional support is disabled.

**Show warning message when changing the text orientation in the editor**

Specifies whether or not to display a warning message when you change the text orientation in the editor.

**Show warning when editing long expressions in the XPath toolbar**

Specifies whether or not to display an information dialog box that allows you to specify if you want to use the XPath/XQuery Builder (on page 2002) view when editing long XPath expressions.

**Show MathML editor recommendation**

Specifies whether or not to display an information dialog box that recommends using the MathFlow Editor (on page 751) to edit MathML equations.

**Show SFTP certificate warning dialog**

Specifies whether or not to display a warning dialog box each time the authenticity of the SFTP server host cannot be established.

**Show Enterprise license related message when trying to connect to a Microsoft SharePoint server**
Specifies whether or not to display an error message if you try to connect to a Microsoft SharePoint server without having the proper license.

**Show the dialog box for choosing the encoding for Base64, Base32, Hex conversions**

Specifies whether or not to display a dialog box that allows you to choose a specific encoding whenever you use the **Encode Selection** or **Decode Selection** actions for Base64 (on page 569), Base32 (on page 570), or Hex conversions (on page 571). In the dialog box, there is an option to choose to not show this dialog box in the future. To reset that behavior, simply select **Restore Defaults** at the bottom of this preferences page.

**Show the dialog box that suggests switching to the DITA perspective**

Specifies whether or not to display a dialog box that asks you if you want to switch to the DITA perspective when you open a DITA resource from the DITA Maps Manager (on page 2897).

**Convert DB Structure to XML Schema**

When tables from a database schema are selected in the **Select database table** section of the Convert DB Structure to XML Schema dialog box (on page 1024) and another database schema is expanded, a confirmation is needed since the previous selection will be discarded. This option specifies whether or not you are always asked if you want the other database schema to always be expanded without asking you, or it is never expanded.

### Configuring Options

A set of options controls the behavior of Oxygen XML Editor, allowing you to configure most of the features. To offer you the highest degree of flexibility in customizing the application to fit the needs of your organization, Oxygen XML Editor includes several distinct layers of option values.

**Figure 45. Option Lookup Priority**

The option layers are as follows (sorted from high priority to low):

- **Project Options**

  Allows project managers to establish a set of rules for a specific project. These rules standardize the information exchanged by the team members (for example, if the project is stored in a repository, a
common set of formatting rules avoid conflicts that may appear when documents modified by various team members are committed to the repository).

- **Global Options** *(on page 312)*

  Allows individual users to personalize Oxygen XML Editor according to their specific needs.

- **Customized Default Options** *(on page 309)*

  Designed to customize the initial option values for a group of users, this layer allows an administrator to deploy the application preconfigured with a standardized set of option values.

  **Note:**
  Once this layer is set, it represents the initial state of Oxygen XML Editor when an end-user selects the **Restore defaults** *(on page 125)* or **Reset Global Options** *(on page 314)* actions.

- **Default Options**

  The predefined default values, tuned so that Oxygen XML Editor behaves optimally in most working environments.

  **Important:**
  If you set a specific option in one of the layers, but it is not applied in the application, make sure that one of the higher priority layers does not overwrite it.

**Customizing Default Options**

Oxygen XML Editor has an extensive set of options that you can configure. When Oxygen XML Editor is installed, these options are set to default values. You can provide a different set of default values for an installation using an XML options file.

**Creating an XML Options File**

To create an options file, follow these steps:

1. It is recommended that you use a fresh install for this procedure, to make sure that you do not copy personal or local preferences.
2. Open Oxygen XML Editor and open the **Preferences** dialog box (**Options > Preferences**) *(on page 124)*.
3. Go through the options and set them to the desired defaults. Make sure that **Global Options** *(on page 3219)* is selected in each page.
4. Click **OK** and close the **Preferences** dialog box.
5. Go to **Options > Export Global Options** to create an XML options file.
Controlling Which Options are Stored in the Default Options File

If you want to control exactly which option pages will be stored in the default options file, you can choose to attach them to a project file (.xpr file extension) by following this procedure:

1. You may want to use a fresh install for this procedure, to make sure that you do not copy personal or local preferences.
2. In the Project view (on page 402), create a project or open an existing one.
3. Open the Preferences dialog box (Options > Preferences) (on page 124).
4. Configure the options in each preferences page that you want to be included in the project file and switch the storage preference to Project Options (on page 3222) in each page.

Note:
Some pages do not have the Project Options button, since the options they host might contain sensitive data (such as passwords, for example) that is unsuitable for sharing with other users.

5. Click OK and close the Preferences dialog box.

All explicitly set values are now saved in the project file. You can then share the project file so that your team will have the same option configuration that you stored in the project file.

Note:
The project file extension (.xpr) must be preserved when the file is distributed to others.

Notice:
When a project is opened for the first time, a confirmation dialog box will be displayed that asks you to confirm that the project came from a trusted source. This is meant to help prevent potential security issues.

Configuring an Installation to Use Customized Default Options

There are several methods that you can use to configure an Oxygen XML Editor installation to use the customized default options from the created XML options file.

Warning:
The disadvantage of customizing the default options is that if the end-user manually changes an option, the default value will no longer be used. An alternative would be to use a plugin to impose a set of options (on page 311).

The possible methods for using customized default options during an installation include:
• Copy the XML Options File to the Installation Directory

In the \[OXYGEN_INSTALL_DIR\], create a folder called preferences and copy the created XML options file into it (for example, \[OXYGEN_INSTALL_DIR\]/preferences/default.xml).

• Specify a Path to the XML Options File in a Startup Parameter

Set the path to the XML options file as the value of the \texttt{com.oxygenxml.default.options} system property in the startup parameters (on page 339). The path can be specified with any of the following:

◦ A URL or file path relative to the application installation folder. For example:

\begin{verbatim}
-Dcom.oxygenxml.default.options=options/default.xml
\end{verbatim}

◦ A system variable that specifies the file path. For example:

\begin{verbatim}
com.oxygenxml.default.options=${system(CONFIG)}/default.xml
\end{verbatim}

◦ An environmental variable that specifies the file path. For example:

\begin{verbatim}
com.oxygenxml.default.options=${env(CONFIG)}/default.xml
\end{verbatim}

Impose a Set of Options Using a Plugin

The Oxygen XML SDK includes a sample Java-based oxygen-sample-plugin-impose-options plugin that shows how to impose a set of options for the end-users every time the API is called. It is possible to use this plugin to impose options but still allow the end-user to change options by calling the API only once, the first time the plugin starts along with Oxygen XML Editor.

A similar JavaScript-based sample impose-options plugin is also available here: https://github.com/oxygenxml/wsaccess-javascript-sample-plugins. This plugin imports a fixed set of options (saved in XML format) when Oxygen XML Editor starts.

Related information

Sharing Application Settings (on page 312)

Storing Global and Project Level Options

When you configure the Oxygen XML Editor options, you can store them globally or bind them to a specific project by choosing the appropriate setting in the preferences pages. They can then be shared with others by exporting the global options (on page 313) or by sharing the stored project-level files (on page 313). The same is true with transformation and validation scenarios.

For each preferences page, you can choose between Global Options (on page 312) and Project Options (on page 312) depending upon how you want to store the options in that particular preferences page.

Notice:

Some pages do not have the Project Options button, since the options they host might contain sensitive data (passwords, for example), unsuitable for sharing with other users.
If changes have been made to the options in a preferences page and you switch between **Project Options** and **Global Options**, a dialog box will be displayed that allows you to select one of the following:

- **Overwrite** - The existing options from the current preferences page will be overwritten.
- **Preserve** - The existing options from the current preferences page will be preserved.

*Figure 46. Controlling the Storage Options for the Preferences*

### Global Options

By default, **Global Options** is selected in the preferences pages, meaning that the options are stored locally on your computer and are not accessible to other users (unless you [export them into an XML options file that can then be shared](on page 312)).

Global options are stored locally in option files (for example, `oxyOptionsSa19.1.xml` for a standalone distribution of Oxygen XML Editor version 19.1) located in the following directories:

- **Windows (7, 8, 10)** - `[user_home_directory]\AppData\Roaming\com.oxygenxml`  
- **macOS** - `[user_home_directory]/Library/Preferences/com.oxygenxml`  
- **Linux/Unix** - `[user_home_directory]/.com.oxygenxml`

### Project Options

If you select **Project Options**, the preferences are stored in the project file (`.xpr`), which can easily be [shared with other users](on page 312).

**Notice:**

Some pages do not have the **Project Options** button, since the options they host might contain sensitive data (passwords, for example), unsuitable for sharing with other users.

---

**Related information**

- Sharing Application Settings ([on page 312](on page 312))
- Customizing Default Options ([on page 309](on page 309))
- Importing/Exporting/Resetting Global Options ([on page 314](on page 314))

---

**Sharing Application Settings**

There are a variety of ways that you can share the settings in Oxygen XML Editor with other members of your team so that you all use a common set of options. This topic describes various possibilities.
Share Settings Through a Project File

Most of the preference pages in Oxygen XML Editor include a Project Options (on page 3222) button that allows you to pass changes to the settings to the current project file that is opened in the Project view (on page 402). That project file can then be shared with other users. For instance, if your project file is saved on a version control system (such as SVN, CVS, or Source Safe) or in a shared folder, your team will have access to the same option configuration that you stored in the project file.

For more information about sharing projects, see Sharing a Project - Team Collaboration (on page 415).

Share Settings by Exporting/Importing Global Options

Oxygen XML Editor includes actions in the Options menu that allow you to export and import the global settings (on page 3219). The Export Global Options action will save the global settings as an XML properties file. You can then share those settings with others by using the Import Global Options action to import that properties file on their computer.

For more information about global options, see Importing/Exporting/Resetting Global Options (on page 314).

Share Settings with a Custom Options File During Installation

When Oxygen XML Editor is installed, all the settings are set to default values. You can customize the set of default values by creating an XML options file that you will use when installing Oxygen XML Editor on each computer. You can then copy the XML options file to the installation directory or specify its path in a startup parameter.

For more information about creating and referencing a custom options file, see Customizing Default Options (on page 309).

Share Settings by Imposing Fixed Options with an API

The Maven-based Oxygen XML SDK includes a sample plugin called ImposeOptions that imposes a fixed set of options when the application starts. This can be achieved by using the 

PluginWorkspaceProvider.getPluginWorkspace().setGlobalObjectProperty(key, value) API method.

For more information about this API, see PluginWorkspaceProvider Class.

Related information

Sharing a Project - Team Collaboration (on page 415)
Sharing Transformation Scenarios (on page 1555)
Sharing Validation Scenarios (on page 808)
Customizing Default Options (on page 309)
Importing/Exporting/Resetting Global Options (on page 314)
Sharing a Framework (on page 2288)
Importing/Exporting/Resetting Global Options

Actions for importing, exporting, and resetting global options are available in the Options menu. The export operation allows you to save global preferences (on page 3219) as an XML properties file and the import operation allows you to load the property file. You can use this file to reload saved options on your computer or to share with others (on page 311).

The following actions are available in the Options menu:

**Reset Global Options**

Restores the preference to the factory defaults or to customized defaults (on page 309). This action also resets the transformation and validation scenarios to the default scenarios and clears recently used document templates.

**Import Global Options**

Allows you to import a set of Global Options from an exported XML properties file. You can also select a project-level options file (on page 415) (.xpr) to import all the Global Options that are set in that project file. After you select a file, the Import Global Options dialog box is displayed, and it informs you that the operation will only override the options that are included in the imported file. You can select the Reset all other options to their default values option to reset all options to the default values before the file is imported.

**Export Global Options**

Allows you to export Global Options to an XML properties file. Some user-specific options that are private are not included. For example, passwords and the name of the Review Author is not included in the export operation.

Oxygen XML Editor automatically stores your global options in an XML properties file. Depending on the platform you are using, this file is located in the following directories:

- [user-home-folder]\AppData\Roaming\com.oxygenxml for Windows
- [user-home-folder]/Library/Preferences/com.oxygenxml for macOS
- [user-home-folder]/.com.oxygenxml for Linux

The name of the options file of Oxygen XML Editor 24.1 is oxyOptionsSa24.1.xml.

Configuring the Layout of the Views and Editors

All of the side-views available in Oxygen XML Editor are dockable (on page 3218) and there are various ways to configure and arrange the layout of the views and editing panes. You can also configure the layout of the toolbars (on page 365).

To open a view, select it from the Window > Show View menu. You can hide a view by closing it with the X button at the top-right corner of the view, or with the Window > Hide current view action.
Arranging the Layout

You can drag any view to any margin of another view or editor inside the Oxygen XML Editor window. Once you create a layout that suits your needs, you can save it from Window > Export Layout. Oxygen XML Editor creates a layout file containing the preferences of the saved layout. To load a layout, go to Window > Load Layout. To reset it, select Window > Reset Layout.

Note:
The Load Layout menu lets you select between the default layout, a predefined layout, or a custom layout. The changes you make using the Load Layout menu are also reflected in the Application Layout preferences page.

The changes you make to any layout are preserved between working sessions. The predefined layout files are saved in the preferences directory (on page 125) of Oxygen XML Editor.

You can drag the editors and arrange them in any order, both horizontally and vertically.

The following image presents two editors arranged as horizontal tiles. To arrange them vertically, drag one of them on top of the other. In the following example, the personal.xml file was dragged over the personal-schema.xml file:
Figure 47. Drag and Drop Editors

---

Hide or Float Views

Hide

To gain more editing space in the Oxygen XML Editor window, click \( \text{Toggle auto-hide} \) in any view. This button sets the view in the *auto-hide* state, making it visible only as a vertical tab, at the margins of the Oxygen XML Editor window. To display a view in the *auto-hide* state, hover its side-tab with your cursor, or click it to keep the view visible until you click elsewhere.

Float

A view can also be set to a floating state by using the \( \text{Toggle floating} \) action, making it independent from the rest of the Oxygen XML Editor window.

Maximize the Editing Environment

You can configure the interface to maximize the editing area, leaving more vertical screen space available for the main editing pane. This is, for example, useful for presentations on low-resolution screens or for laptops.
with small screen space. You can use the following two actions that are available in the Window menu to create a near full-screen editing environment:

**Maximize Editor Area**

If toggled on, all side views are minimized to give you more horizontal space in the editing pane.

**Hide All Toolbars**

If toggled on, all toolbar buttons are hidden to give you more vertical space in the interface.

### Tile/Stack Editor Actions

You can also tile or stack all open editors using the following actions from the toolbar or Window menu:

- **Tile Editors Horizontally**
  
  Splits the editing area into horizontal tiles, one for each open file.

- **Tile Editors Vertically**
  
  Splits the editing area into vertical tiles, one for each open file.

- **Stack Editors**
  
  The reverse of the Tile Editors Horizontally/Vertically actions. Stacks all open editors.

- **Synchronous Scrolling**

  Select this action to scroll through the tiled editors at the same time.

---

**Note:**

When tiled, you can still drag and drop the editors, but note that they are docked in the same way as a window/view (instead of just tabs). You are actually rearranging the editor windows, so drag the editor tab and drop it to one of the sides of an editor (left/right/top/bottom). While dragging, you will see the dark gray rectangle aligned to one of the sides of the editor, or around the entire editor window. If you drop it to one of the sides, it will dock to that side of the editor. If you drop it when the rectangle is around the entire window of the editor, it will get stacked on top of that editor. You can also grab one of the stacked editors and tile it to one of the sides.

### Split Editor Actions

You can divide the editing area vertically and horizontally using the following actions available in the toolbar and Window menu:

- **Split Editor Horizontally** - Splits the editor horizontally so that two editor panes are displayed with one on top of the other. This is useful for comparing and merging content between two documents.

- **Split Editor Vertically** - Splits the editor vertically so that two editor panes are displayed side by side. This is useful for comparing and merging content between two documents.

- **Unsplit Editor** - Removes a split action on the editing area.
To maximize or restore the editors, go to Window > Maximize Editing Area.

Switch, Move, or Hide Editor Tabs

Each file that has been opened has a tab at the top of the editing pane and there are several ways to switch between tabs or move them, and you can even hide the tabs to only show the currently open file.

Note:

If multiple file tabs are left open when you close the application, upon startup, Oxygen XML Editor will not load the file content until you switch to the corresponding file tab. The tabs remain visible as a placeholders until the focus is switched to them. This helps to improve the application's startup time. If you want to disable this feature (meaning that the previously open files will all be re-loaded at startup), deselect the Load file content only when switching to its corresponding editor tab option in the Global preferences page (on page 127).

Switching Editor Tabs

You can switch between editor tabs by using any of the following methods:

Mouse and Scroll Wheel

Of course, you can switch to a different editor tab by left-clicking the tab with your mouse, but when there are too many open tabs to fit on the screen, you can hover over the tab stripe and use the scroll wheel on your mouse to scroll to the left or right (same as using the two arrows on the far-right of the tab stripe).

Buttons on the Far-Right of the Tab Stripe (➡️)

You can use the arrow buttons (⬅️➡️) on the right side of the tab stripe to scroll to the left or right and the Show List button opens a pop-up window that displays all the open file tabs and allows you to select and switch to a specific open file.

Ctrl + Tab (Command + Tab on macOS) [NOTE: Ctrl + Page Down (Ctrl + Alt + Right Arrow on macOS) does the same]

Switches to the next open tab in the order specified in the Order of switching between editor tabs option (on page 128).

Ctrl + Shift + Tab (Command + Shift + Tab on macOS) [NOTE: Ctrl + Page Up (Ctrl + Alt + Left Arrow on macOS) does the same]

Switches to the previous open tab in the order specified in the Order of switching between editor tabs option (on page 128).

Window > Switch editor tab (Ctrl + F9 (Command + F9 on macOS))

This action opens a dialog box that allows you to switch to a particular editor tab by selecting it from a filterable list. This is especially helpful when you have a large amount of open file tabs and you want to switch to a certain tab this is not shown on the screen. It includes a search filter field and several options to help you find specific open file tabs.
The **Switch Editor Tab** dialog box contains the following options and features:

**Search Filter**

You can enter text in the filter field at the top of the dialog box to filter the list and search for specific open files. You can enter any number of terms, separated by space, and wildcards are allowed (for example, * to match any sequence of characters, or ? to match a single character). This field also has a history drop-down that allows you to select previously used search terms.

**Match all terms**

If this option is selected, only the files that match all of your search terms will be displayed. If you use a wildcard in the search filter, this option is automatically disabled.

**Include file paths**

If this option is selected, the search is expanded to include file paths, and also the paths are displayed in this dialog box.

**Case sensitive**

If this option is selected, the search operation will be case-sensitive.

**List of Open File Tabs**

You can click the files in the list to switch to the corresponding tab in the editor.
All files that are currently open are displayed in the upper part of the main pane of the dialog box, followed by recently closed files. Files that have been modified but not yet saved are prefixed by an asterisk. To switch to a particular file tab, double-click the file or select it and click OK.

**Moving Editor Tabs**

You can move editor tabs by using any of the following methods:

**Mouse Drag**

You can use your mouse to drag editor tabs to a new location on the tab stripe.

**Ctrl + Alt + Comma**

Moves the current file tab one position to the left.

**Ctrl + Alt + Period**

Moves the current file tab one position to the right.

**Hiding Editor Tabs**

If you want to hide all the file tabs and only show the currently open file, select *Hide editor tabs* from the *Window* menu. This does not close the other tabs, just hides them. You can still navigate between tabs using keyboard shortcuts (**Ctrl + Tab**, **Ctrl + Shift + Tab**, **Ctrl + F6**, **Ctrl + Shift + F6**) or by selecting *Next editor* or *Previous editor* from the *Window* menu.

For more information about configuring the interface of Oxygen XML Editor, watch our video demonstration:

https://www.youtube.com/embed/anwjepfAdEk

---

**Tip:**

To get more ideas for more advanced customization possibilities, watch our Webinar: *Working with DITA in Oxygen - Customizing the Editing Experience*. It offers a visual demonstration of how to customize actions, document validation, content completion, new document templates, *Author* mode rendering, and more.

---

**Related information**

- Configuring Toolbars *(on page 365)*

---

**Configuring Toolbars**

You can configure the toolbars in Oxygen XML Editor to personalize the interface for your specific needs. You can choose which toolbars to show or hide in the current editor mode (**Text**, **Author**, **Design**, or **Grid**) and in the current *perspective* *(on page 3221)* (**Editor**, **XSLT Debugger**, **XQuery Debugger**, or **Database**). You can also choose which actions to display in each toolbar, add actions to toolbars, and customize the layout of the toolbars.
To configure the toolbars, open the **Configure Toolbars** dialog box by doing one of the following:

- Right-click any toolbar and select **Configure Toolbars**.
- Select **Configure Toolbars** from the **Window** menu.

**Figure 49. Configure Toolbars Dialog Box**

The **Configure Toolbars** dialog box provides the following features:

**Filter Text Box**
You can use the filter text box at the top of the dialog box to search for a specific toolbar or action.

**Show or Hide Toolbars**
You can choose whether to show or hide a toolbar by using the checkbox next to the toolbar name. This checkbox is only available for toolbars that are available for the current **perspective (on page 3221)** and editing mode.

**Show or Hide Actions in a Toolbar**
To show or hide actions in a toolbar, expand it by clicking the arrow next to the toolbar name, then use the checkbox to select or deselect the appropriate actions. The toolbar configuration changes in the **Preview** column according to your changes.

**Add Actions to a Toolbar**
Use the **Add Actions** button to open the **Add Actions** dialog box that displays all the actions that can be added to any of the toolbars, with the exception of those that are contributed from **frameworks (on page 3219)** or 3rd party **plugins (on page 3222)**.

**Remove Actions from a Toolbar**
You can remove actions that you have previously added to toolbars by using the Remove Action button.

**Move Actions in a Toolbar**

Use the **Move Up** and **Move Down** actions to change the order of the actions in a toolbar.

The **Configure Toolbars** dialog box also provides a variety of other ways to customize the layout in Oxygen XML Editor.

**Customize My Toolbar**

You can customize the **My Toolbar** to include your most commonly used actions. By default, this toolbar is listed first. Also, it is hidden until you add actions to it and you can easily hide it with the **Hide "My Toolbar" Toolbar** action that is available when you right-click anywhere in the toolbar area.

**Drop-down Menu Actions**

Composite actions that are usually displayed as a drop-down menu can only be selected in one toolbar at a time. These actions are displayed in the **Configure Toolbars** dialog box with the name in brackets.

**Configure External Tools Action**

There is a **Configure external tools** composite action that appears in the toolbar called **Tools**. It is a drop-down menu that contains any external tools that are configured in the **External Tools** preferences page.

**Note:**

If no external tools are configured, this drop-down menu is not shown in the toolbar.

Additional actions are available from the **Window** menu or contextual menu when invoked from a toolbar that allows you to further customize your layout. These actions include:

**Reset Toolbars**

To reset the layout of toolbars to the default setting, select the **Reset Toolbars** action from the contextual menu or **Window** menu.

**Reset Layout**

To reset the entire layout (including toolbars, editing modes, views, etc.) to the default setting, select **Reset Layout** from the contextual menu or **Window** menu.

**Export Layout**

You can use the **Export Layout** action that is available in the **Window** menu to export the entire layout of the application to share it with other users.
Hide Toolbars

You can use the **Hide Toolbar** action from the contextual menu to easily hide a displayed toolbar. When you right-click a toolbar it will be highlighted to show you which actions are included in that toolbar.

Related information

Configuring the Layout of the Views and Editors *(on page 360)*

Import/Export Transformation or Validation Scenarios

You can export global transformation and validation scenarios into specialized *scenarios* files. You can import transformation and validation scenarios from various sources (such as project files, *framework* *(on page 3219)* option files, or exported scenario files). The import and export scenario actions are available in the **Options** menu. The following actions are available:

- **Import Transformation Scenarios**
  
  Loads a set of transformation scenarios from a project file, *framework* options file, or exported scenarios file.

- **Export Global Transformation Scenarios**
  
  Stores a set of global transformation scenarios in a specialized *scenarios* file.

- **Import Validation Scenarios**
  
  Loads a set of validation scenarios from a project file, *framework* options file, or exported scenarios file.

- **Export Global Validation Scenarios**
  
  Stores a set of global validation scenarios in a specialized *scenarios* file.

The **Export Global Transformation Scenarios** and **Export Global Validation Scenarios** options are used to store all the scenarios in a separate file. Associations between document URLs and scenarios are also saved in this file. You can load the saved scenarios using the **Import Transformation Scenarios** and **Import Validation Scenarios** actions. To distinguish the existing scenarios and the imported ones, the names of the imported scenarios contain the word *import*.

Editor Variables

An editor variable is a shorthand notation for context-dependent information, such as a file or folder path, a time-stamp, or a date. It is used in the definition of a command (for example, the input URL of a transformation, the output file path of a transformation, or the command line of an external tool) to make a command or a parameter generic and re-usable with other input files. When the same command is applied to multiple files, the notation is expanded at the execution of the command so that the same command has different effects depending on the actual file.
Oxygen XML Editor includes a variety of built-in editor variables. You can also create your own custom editor variables (on page 332) by using the Custom Editor Variables preferences page (on page 300).

Editor variables are evaluated and automatically expanded in many places in the application, when:

- Creating new documents from file templates (on page 375).
- Inserting code templates (on page 375) in the Text or Author editor modes.
- Running custom configured External Tools (on page 2852).
- Executing predefined Built-in Author Mode Operations (on page 2152) that have editor variables given as parameter values.
- Running validation scenarios (on page 798) that use editor variables inside to reference various resources.
- Executing transformation scenarios (of type ANT, DITA-OT (on page 3112), XSLT (on page 1464), etc.) that have editor variables set as parameter values or as values for references to various resources.
- Expanding CSS imports (on page 2309) for editing in the Author visual editing mode.
- Using specific Java API `UtilAccess.expandEditorVariables(String, URL)` from plugins and framework extensions.

You can use the following editor variables in Oxygen XML Editor commands of external engines or other external tools, and in various places in the application, such as in transformation scenarios, Author mode operations, and validation scenarios:

- `$\{activeConditionSet\}` - Current active profiling condition set (on page 675) name. If there is no active condition set, the variable will be replaced with an empty string.
- `$\{af\}` - The local file path of the ZIP archive that includes the currently edited document.
- `$\{afd\}` - The local directory path of the ZIP archive that includes the currently edited document.
- `$\{afdu\}` - The URL path of the directory of the ZIP archive that includes the currently edited document.
- `$\{afn\}` - The file name (without parent directory and without file extension) of the zip archive that includes the currently edited file.
- `$\{afne\}` - The file name (with file extension, for example .zip or .epub, but without parent directory) of the zip archive that includes the currently edited file.
- `$\{afu\}` - The URL path of the ZIP archive that includes the currently edited document.
- `$\{answer(@id)\}` - Used in conjunction with the `$\{ask\}` editor variable. The `@id` parameter is required and identifies the answer from the `$\{ask\}` editor variable with the same ID.

**Example:**

```xml
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE topic PUBLIC "-//OASIS//DTD DITA Topic//EN" "topic.dtd">
<topic id="topic_lcf_lc4_tdb">
  <title></title>
  <body>
    <data name="\{ask('Set a data name', String, 'name', @name)\}"></data>
    <p>The name is: \{answer(@name)\}</p>
  </body>
</topic>
```
You can set the following parameters:

- **message**: The displayed message. Note the quotes that enclose the message.
- **default-value**: Optional parameter. Provides a default value.
- **@id**: Optional parameter. Used for identifying the variable to reuse the answer using the ${answer(@id)} editor variable.
- **type**: Optional parameter (defaults to `generic`), with one of the following values:

  - **url**
  - **relative_url**
  - **text**
  - **boolean**
  - **integer**
  - **number**
  - **date**
  - **time**
  - **choice**

Note:
The title of the dialog box will be determined by the type of parameter and as follows:

- For `url` and `relative_url` parameters, the title will be the name of the parameter and the value of the `message`.
- For the other parameters listed below, the title will be the name of that respective parameter.
- If no parameter is used, the title will be "Input".

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Format:</th>
<th>Description:</th>
<th>Example:</th>
</tr>
</thead>
</table>
| `generic` (default) | ${ask('message', generic, 'default')} | The input is considered to be generic text that requires no special handling. | - ${ask('Hello world!')}: The dialog box has a **Hello world!** message displayed.  
- ${ask('Hello world!', generic, 'Hello again!')}: The dialog box has a **Hello world!** message displayed and the value displayed in the input box is **Hello again!**. |
<p>| <code>url</code> | ${ask('message', url, 'default_value')} | Input is considered a URL. Oxygen XML Editor checks that the provided URL is valid. | |</p>
<table>
<thead>
<tr>
<th>Parameter</th>
<th></th>
</tr>
</thead>
</table>
| **relative_url** | Format: `{ask('message', relative_url, 'default')}`  
Description: Input is considered a URL. Oxygen XML Editor tries to make the URL relative to that of the document you are editing.  

**Note:**  
If the `ask` editor variable is expanded in content that is not yet saved (such as an *untitled* file, whose path cannot be determined), then Oxygen XML Editor will transform it into an absolute URL.  

Example:  
```
{ask('File location', relative_url, 'C:/example.txt')} - The dialog box has the name 'File location'. The URL inserted in the input box is made relative to the currently edited document location.
```
| password | Format: `{ask('message', password, 'default')}`  
Description: The input is hidden with bullet characters.  

Example:  
```
- `{ask('Input password', password)} - The displayed dialog box has the name 'Input password' and the input is hidden with bullet symbols.
- `{ask('Input password', password, 'abcd')} - The displayed dialog box has the name 'Input password' and the input hidden with bullet symbols. The input field already contains the default `abcd` value.
```
| combobox | Format: `{ask('message', combobox, ('real_value1':'rendered_value1';..::;':'real_valueN':'rendered_valueN'), 'default')}`  
Description: Displays a dialog box that offers a drop-down menu. The drop-down menu is populated with the given `rendered_value` values. Choosing such a value will return its associated value `real_value`.  

**Note:**  
The list of `real_value':'rendered_value` pairs can be computed using `{xpath_eval()}`.
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Note:</td>
<td>The ‘default’ parameter specifies the default-selected value and can match either a key or a value.</td>
</tr>
<tr>
<td>Example:</td>
<td><strong>▪</strong> ${ask('Operating System', combobox, ('win':Microsoft Windows';'macos':'macOS';'lnx':'Linux/UNIX'), 'macos') - The dialog box has the name 'Operating System'. The drop-down menu displays the three given operating systems. The associated value will be returned based upon your selection.**</td>
</tr>
<tr>
<td>Note:</td>
<td>In this example, the default value is indicated by the osx key. However, the same result could be obtained if the default value is indicated by macOS, as in the following example: <strong>▪</strong> ${ask('Operating System', combobox, ('win':Microsoft Windows';'macos':'macOS';'lnx':'Linux/UNIX'), 'macOS')**</td>
</tr>
</tbody>
</table>
| editable_combobox     | Format: **▪** ${ask('message', editable_combobox, ('real_value1':rendered_value1';...;'real_valueN':rendered_valueN'), 'default')} **Description:** Displays a dialog box that offers a drop-down menu with editable elements. The drop-down menu is populated with the given rendered_value values. Choosing such a value will return its associated real value (real_value) or the value inserted when you edit a list entry. **Note:** The list of ‘real_value’:‘rendered_value’ pairs can be computed using ${xpath_eval()}. **Note:** The ‘default’ parameter specifies the default-selected value and can match either a key or a value.
### Parameter

<table>
<thead>
<tr>
<th>Example:</th>
</tr>
</thead>
<tbody>
<tr>
<td>• ${\text{ask}('Operating System', \text{editable_combobox}, ('win': 'Microsoft Windows'; 'macos': 'macOS'; 'lnx': 'Linux/UNIX'), 'macos')} - The dialog box has the name 'Operating System'. The drop-down menu displays the three given operating systems and also allows you to edit the entry. The associated value will be returned based upon your selection or the text you input.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Example:</th>
</tr>
</thead>
<tbody>
<tr>
<td>• ${\text{ask}('Operating System', \text{editable_combobox}, ${\text{xpath_eval}(\text{for}\ $\text{pair}\ \text{in}\ ([\text{win}', 'Microsoft Windows'], [\text{macos}', 'macOS'], [\text{lnx}', 'Linux/UNIX'])\ \text{return} \ '' \</td>
<td></td>
</tr>
</tbody>
</table>

| Format: $\{\text{ask}('message', \text{radio}, ('\text{real\_value1}': '\text{rendered\_value1}';...;;'\text{real\_valueN}': '\text{rendered\_valueN}'), 'default')\} |

| Description: Displays a dialog box that offers a series of radio buttons. Each radio button displays a 'rendered value' and will return an associated real value. |

| Note: |
| The list of 'real\_value': 'rendered\_value' pairs can be computed using $\{\text{xpath\_eval}()\}. |

| Note: |
| The 'default' parameter specifies the default-selected value and can match either a key or a value. |

<table>
<thead>
<tr>
<th>Example:</th>
</tr>
</thead>
<tbody>
<tr>
<td>• ${\text{ask}('Operating System', \text{radio}, ('win': 'Microsoft Windows'; 'macos': 'macOS'; 'lnx': 'Linux/UNIX'), 'macos')} - The dialog box has the name 'Operating System'. The radio button group allows you to choose between the three operating systems.</td>
</tr>
</tbody>
</table>

| Note: |
| In this example, macOS is the default-selected value and if selected, it would return macos for the output. |

<table>
<thead>
<tr>
<th>Example:</th>
</tr>
</thead>
<tbody>
<tr>
<td>• ${\text{ask}('Operating System', \text{radio}, ${\text{xpath_eval}(\text{for}\ $\text{pair}\ \text{in}\ ([\text{win}', 'Microsoft Windows'], [\text{macos}', 'macOS'], [\text{lnx}', 'Linux/UNIX'])\ \text{return} \ '' \</td>
</tr>
</tbody>
</table>
• ${author.name} - Expands to the current author name that is set in the Review preferences page (on page 183).
• ${caret} - The position where the cursor is located. This variable can be used in a code template, in Author mode operations, or in a selection plugin.

**Note:**
The ${caret} editor variable is available only for parameters that take XML content as values. It is replaced with the $\text{UNIQUE_CARET_MARKER_FOR_AUTHOR}$ macro. The default Author operations process this macro and position the cursor at the designated offset.

**Note:**
The ${caret} editor variable can be used for setting a fixed cursor position inside an XML fragment. To set the cursor position depending on the fragment inserted in the document, you can use AuthorDocumentFilter and inside the insertFragment(AuthorDocumentFilterBypass, int, AuthorDocumentFragment) method, use the AuthorDocumentFragment.setSuggestedRelativeCaretOffset(int) API on the given fragment.

• ${cf} - Current file as file path, that is the absolute file path of the currently edited document.
• ${cfd} - Current file folder as file path, that is the path of the currently edited document up to the name of the parent folder.
• ${cfdu} - Current file folder as URL, that is the path of the currently edited document up to the name of the parent folder, represented as a URL.
• ${cfn} - Current file name without the extension and parent folder. The current file is the one currently open and selected.
• ${cfne} - Current file name with extension. The current file is the one currently open and selected.
• ${comma} - Used to display a comma when the actual comma symbol would be considered part of some sort of instruction or delimiter.
• ${configured.ditaot.dir} - The default directory of the DITA Open Toolkit distribution, as configured in the DITA preferences page (on page 269).
• ${cp} - Current page number. Used to display the current page number on each printed page in the Editor / Print Preferences page.
• ${currentFileURL} - Current file as URL, that is the absolute file path of the currently edited document represented as URL.
• ${date(pattern)} - Current date. The allowed patterns are equivalent to the ones in the Java SimpleDateFormat class. Example: yyyy-MM-dd.

**Note:**
This editor variable supports both the xs:date and xs:datetime parameters. For details about xs:date, go to: http://www.w3.org/TR/xmlschema-2/#date. For details about xs:datetime, go to: http://www.w3.org/TR/xmlschema-2/#dateTime.
• ${dbgXML} - The local file path to the XML document that is currently selected in the Debugger source combo box (for tools started from the XSLT/XQuery Debugger).

• ${dbgXSL} - The local file path to the XSL/XQuery document that is currently selected in the Debugger stylesheet combo box (for tools started from the XSLT/XQuery Debugger).

• ${dita.dir.url} - A special local contextual editor variable that gets expanded only in the Libraries dialog box that is accessible from the Advanced tab of DITA transformation scenarios. The Libraries dialog box allows you to specify additional libraries (JAR (on page 3220) files or additional class paths) to be used by the transformer. This ${dita.dir.url} editor variable gets expanded to the value of the dita.dir parameter from the Parameters tab of the DITA transformation scenario.

• ${ds} - The path of the detected schema as a local file path for the current validated XML document.

• ${dsu} - The path of the detected schema as a URL for the current validated XML document.

• ${env(VAR_NAME)} - Value of the VAR_NAME environment variable. The environment variables are managed by the operating system. If you are looking for Java System Properties, use the ${system(var.name)} editor variable.

• ${framework(fr_name)} - The path (as URL) of the fr_name framework.

• ${framework} - The path (as URL) of the current framework directory.

• ${frameworkDir(fr_name)} - The path (as file path) of the fr_name framework.

Note:
Since multiple frameworks might have the same name (although it is not recommended), for both ${framework(fr_name)} and ${frameworkDir(fr_name)} editor variables Oxygen XML Editor employs the following algorithm when searching for a given framework name:

◦ All frameworks are sorted, from high to low, according to their Priority (on page 140) setting from the Document Type configuration dialog box (on page 140). Only frameworks that have the Enabled checkbox selected are taken into account.

◦ Next, if the two or more frameworks have the same name and priority, a further sorting based on the Storage setting is made, in the exact following order:

  ▪ Frameworks stored in the internal Oxygen XML Editor options.
  ▪ Additional frameworks added in the Locations preferences page (on page 139).
  ▪ Frameworks installed using the add-ons support.
  ▪ Frameworks found in the main framework location (on page 139) (Default or Custom).

• ${frameworkDir} - The path (as file path) of the current framework directory.

• ${frameworks} - The path (as URL) of the frameworks directory. When used to define references inside a framework configuration, it expands to the parent folder of that specific framework folder. Otherwise, it expands to the main frameworks folder defined in the Document Type Association > Locations preferences page.

• ${frameworksDir} - The path (as file path) of the frameworks directory. When used to define references inside a framework configuration, it expands to the parent folder of that specific framework folder. Otherwise, it expands to the main frameworks folder defined in the Document Type Association > Locations preferences page.
• \${home} - The path (as URL) of the user home folder.
• \${homeDir} - The path (as file path) of the user home folder.
• \${i18n(key)} - Editor variable used only at framework-level to allow translating names and descriptions of Author mode actions in multiple actions. For more details, see Localizing Frameworks (on page 2229).
• \${id} - Application-level unique identifier. It is a short sequence of 10-12 letters and digits that is not guaranteed to be universally unique.

• \${makeRelative(base,location)} - Takes two URL-like paths as parameters and tries to return a relative path. A use-case would be to insert content references to a certain reusable component when defining code templates.

Example:

\${makeRelative(${currentFileURL}, ${dictionaryURL}#gogu)}

• \${oxygenHome} - Oxygen XML Editor installation folder as URL.
• \${oxygenInstallDir} - Oxygen XML Editor installation folder as file path.
• \${pd} - The file path to the folder that contains the current project file (.xpr).
• \${pdu} - The URL path to the folder that contains the current project file (.xpr).
• \${pluginDir(pluginID)} - Each plugin has an ID specified in its plugin.xml file. This editor variable expands to the file path of the folder that contains the plugin.xml file where that specific plugin ID is located.
• \${pluginDirURL(pluginID)} - Each plugin has an ID specified in its plugin.xml file. This editor variable expands to the URL path of the folder that contains the plugin.xml file where that specific plugin ID is located.
• \${pn} - Current project name.
• \${ps} - Path separator, which is the separator that can be used on the current platform (Windows, macOS, Linux) between library files specified in the class path.
• \${rootMapDir} - Will be expanded to the current root map parent directory file path.
• \${rootMapDirURL} - Will be expanded to the current root map parent directory URL.
• \${rootMapFile} - Will be expanded to the current root map file path.
• \${rootMapURL} - Will be expanded to the current root map URL. For example, if in the main DITA Map you define a key with a certain value:

```
<keydef keys="test">
  <topicmeta><keywords><keyword>ABC</keyword></keywords></topicmeta>
</keydef>
```

you can modify a DITA-OT publishing parameter to have the value: \${xpath_eval(doc('${rootMapURL}')/keydef[@keys='test']/keywords/keyword/text())}. It will be expanded to the value of that specified key name.

• \${selection} - The currently selected text content in the currently edited document. This variable can be used in a code template, in Author mode operations, or in a selection plugin.
• ${system(var.name)} - Value of the var.name Java System Property. The Java system properties can be specified in the command-line arguments of the Java runtime as -Dvar.name=var.value. If you are looking for operating system environment variables, use the ${env(VAR_NAME)} editor variable instead.

• ${timeStamp} - The timestamp, which is the current time in Unix format. For example, it can be used to save transformation results in multiple output files on each transformation.

• ${tp} - Total number of pages in the document. Used to display the total number of pages on each printed page in the Editor / Print Preferences page.

• ${tsf} - The transformation result file path. If the current opened file has an associated scenario that specifies a transformation output file, this variable expands to it.

• ${uuid} - Universally unique identifier, a unique sequence of 32 hexadecimal digits generated by the Java UUID class.

• ${xmlCatalogFilesList} - A list of file paths that point to all known XML catalog files, separated by semicolons (;).

• ${xpath_eval(expression)} - Evaluates an XPath expression. Depending on the context, the expression can be:

  ◦ static - When executed in a non-XML context. For example, you can use such static expressions to perform string operations on other editor variables for composing the name of the output file in a transformation scenario's Output tab.
  
  Example:

  ```
  ${xpath_eval(upper-case(substring('${cfn}', 1, 4)))}
  ```

  ◦ dynamic - When executed in an XML context. For example, you can use such dynamic expression in a code template or as a value of a parameter of an Author mode operation.

  Example:

  ```
  ${ask('Set new ID attribute', generic, '${xpath_eval(@id)})')}
  ```

Related information

- Code Templates (on page 536)
- Selection Plugin Extension (on page 2436)
- Installing and Updating Add-ons (on page 120)

Custom Editor Variables

An editor variable (on page 323) can be created and included in any user-defined expression where a built-in editor variable is also allowed. For example, a custom editor variable may be necessary for configuring the command line of an external tool, the working directory of a custom validator, the command line of a custom XSLT engine, or a custom FO processor.

You can create or configure custom editor variables in the Custom Editor Variables preferences page (on page 300). To create a custom editor variable, follow these steps:
1. Open the Preferences dialog box (Options > Preferences) (on page 124) and go to Custom Editor Variables.
2. Click the New button at the bottom of the table.
3. Use the subsequent dialog box to specify the Name, Value, and Description for the new editor variable.
4. Click OK to save your configuration.

Related information
Editor Variables (on page 323)

Custom System Properties

A variety of Java system properties can be set in the application to influence its behavior. For information about how to do this, see Setting a System Property (on page 340).

com.oxygenxml.disable.http.protocol.handlers

- **Allowed Values:** true or false
- **Default Value:** false
- **Purpose:** By default, Oxygen XML Editor uses the open source Apache HTTP Client software for HTTP(S) connections. If set to true, the default Java Sun HTTP(S) will be used instead. You will also lose WebDAV support and possibly other related features.

com.oxygenxml.present.license.reminders

- **Allowed Values:** true or false
- **Default Value:** true
- **Purpose:** When set to false, Oxygen XML Editor will not display the messages that remind you to renew your Support and Maintenance Pack that covers your current license.

com.oxygenxml.enable.content.reference.caching

- **Allowed Values:** true or false
- **Default Value:** true
- **Purpose:** Enables content reference caching.

com.oxygenxml.eclipse.remove.grid.editing.mode

- **Allowed Values:** true or false
- **Default Value:** false
- **Purpose:** When set to false, Oxygen XML Editor does not show the Grid editing mode when opening an XML document.

com.oxygenxml.default.java.accessibility
• **Allowed Values:** true or false
• **Default Value:** false
• **Purpose:** System property that can be set to true to force the default detection of Java accessibility. If `com.sun.java.accessibility.AccessBridge` cannot be loaded, Oxygen XML Editor forces the Java accessibility to be disabled.

**com.oxygenxml.floating.license.timeout**

• **Allowed Values:** An integer (minutes)
• **Default Value:** 120
• **Purpose:** Stores the time interval (in minutes) before floating licenses are released in case of application's inactivity.

**com.oxygenxml.language**

• **Allowed Values:** Language code (for example, en-us)
• **Default Value:** N/A
• **Purpose:** Property that holds the language code set during installation.

**com.oxygenxml.default.options**

• **Allowed Values:** A URL-type relative or absolute path.
• **Default Value:** N/A
• **Purpose:** Provides the path to an XML file containing default application options. For more details, see **Customizing Default Options (on page 309)**.

**com.oxygenxml.customOptionsDir**

• **Allowed Values:** A file system absolute path pointing to a folder.
• **Default Value:** N/A
• **Purpose:** Sets a folder to be used by the application to load and save preference files. The default location where the options are saved varies according to the operating system. For more details, see **Importing/Exporting/Resetting Global Options (on page 314)**.

**com.oxygenxml.ApplicationDataFolder** (Windows only)

• **Allowed Values:** A file system absolute path pointing to a folder.
• **Default Value:** %APPDATA%
• **Purpose:** When the application runs on Windows, you can set this property to change the location where the application considers that the APPDATA folder is located.

**com.oxygenxml.editor.frameworks.url**
• **Allowed Values**: A URL-type absolute path.
• **Default Value**: `OXYGEN_DIR/frameworks`
• **Purpose**: Changes the folder where the application considers that the main frameworks are installed. It has the same effect as changing the custom frameworks directory value in the Location preferences page (on page 139).

**com.oxygenxml.editor.plugins.dir**

• **Allowed Values**: The path can be specified with any of the following:
  - A URL or file path that is relative to the application's installation folder (for example: `-Dcom.oxygenxml.editor.plugins.dir=my-plugins`).
  - A system variable that specifies the file path (for example: `-Dcom.oxygenxml.editor.plugins.dir=${system(CONFIG)}/plugins`).
  - An environmental variable that specifies the file path (for example: `-Dcom.oxygenxml.editor.plugins.dir=${env(CONFIG)}/plugins`).
• **Default Value**: N/A
• **Purpose**: Specifies the directory where the application finds plugins to load.

**com.oxygenxml.MultipleInstances**

• **Allowed Values**: `true` or `false`
• **Default Value**: `false`
• **Purpose**: If set to `true`, multiple instances of the application are allowed to be started.

**com.oxygenxml.xep.location**

• **Allowed Values**: A file system absolute path pointing to a folder.
• **Default Value**: N/A
• **Purpose**: Points to a folder where RenderX XEP is installed. Has the same effect as configuring XEP in the FO Processors preferences page (on page 261).

**com.oxygenxml.additional.classpath**

• **Allowed Values**: A list of JAR (on page 3220)-type resources separated by a classpath separator.
• **Default Value**: N/A
• **Purpose**: An additional list of libraries to be used in the application's internal class loader in addition to the libraries specified in the `lib` folder.

**com.oxygenxml.user.home (Windows only)**

• **Allowed Values**: A file system absolute path pointing to a folder.
• **Default Value**: `USERPROFILE` folder
• **Purpose**: Overwrites the user home directory that was implicitly detected for the application.
**com.oxygenxml.use.late.delegation.for.author.extensions**

- **Allowed Values:** `true` or `false`
- **Default Value:** `true`
- **Purpose:** All Java extensions in a framework configuration are instantiated in a separate class loader. When `true`, the JAR libraries used in a certain document type will have priority to resolve classes before delegating to the parent class loader. When `false`, the parent class loader will take precedence.

**com.oxygenxml.stack.size.validation.threads**

- **Allowed Values:** The number of bytes used for validation threads.
- **Default Value:** `5*1024*1024`
- **Purpose:** Some parts of the application (validation, content completion) that use the Relax NG parser sometimes require a larger `Thread` stack size to parse complex schemas. The default value should be more than enough.

**com.oxygenxml.jing.skip.validation.xhtml.data.attrs**

- **Allowed Values:** `true` or `false`
- **Default Value:** `true`
- **Purpose:** By default, the Relax NG validation was configured to skip validation for XHTML attributes that start with "data-", which should be skipped from validation according to the XHTML 5 specification.

**com.oxygenxml.report.problems.url**

- **Allowed Values:** User-defined URL
- **Default Value:** N/A
- **Purpose:** The URL where a problem reported through the Report Problem dialog box is sent. The report is sent in XML format using the `report` parameter with the POST HTTP method.

**com.oxygenxml.parallel.title.computing.threads**

- **Allowed Values:** Integers
- **Default Value:** 4
- **Purpose:** The number of parallel threads that will be used to compute referenced topic titles. Increasing this value reduces the amount of time it takes to compute topic titles in the DITA Maps Manager view.

**com.oxygenxml.hidpi.scaling**
• **Allowed Values:** Numerical values between 1 and 2 (1, 1.5, and 2 have been tested, and for example, 1.5 is for 150% scaling)
• **Default Value:** N/A
• **Purpose:** Used to override the HiDPI scaling detection to force a specific scaling setting. This is helpful if you encounter scaling detection issues in Windows or Linux.

**com.oxygenxml.prefer.plugin.classloader.context.loader**

• **Allowed Values:** true or false
• **Default Value:** true
• **Purpose:** Used to instruct the application to use the plugin class loader when there is code that loads content (usually Xerces code) using the thread's class loader. For instance, if you have a plugin that specifies a certain Xerces version and you want to load that version instead of the one from Oxygen's lib directory.

**com.oxygenxml.classic.file.output.stream.save**

• **Allowed Values:** true or false
• **Default Value:** false
• **Purpose:** When set to true, the files are saved using a Java classic file output stream, which destroys the NTFS alternate data streams set on the file. However, this might prevent data loss in the rare occasions when Oxygen XML Editor saves empty file content over shared network drives.

**com.oxygenxml.disable.correct.over.ascii.chars**

• **Allowed Values:** true or false
• **Default Value:** false
• **Purpose:** By default, Oxygen XML Editor will escape non-ASCII characters (encode them with their hexadecimal equivalent) within URL paths. If set to true, non-ASCII characters in URL paths will not be escaped. This is helpful if you are using a non-Latin alphabet (such as Arab, Japanese, Chinese) since the URL paths will remain unchanged and will be more readable.

**com.oxygenxml.format.indent.files.parallel**

• **Allowed Values:** true or false
• **Default Value:** false
• **Purpose:** By default, when using the Format and Indent Files action from the Project view, only one thread will be used for the formatter. If the system property is enabled, up to four parallel threads are used by the operation, speeding up the processing when formatting very large or a large amount of documents.
Localizing the User Interface

Oxygen XML Editor comes with the following built-in languages: English, French, German, Japanese, Dutch, and Chinese. To change the interface language, go to Options > Preferences > Global preferences page, then choose the appropriate language from the Language drop-down menu.

You can also localize the interface in another language by creating an interface localization file.

How to Create an Interface Localization File

You can change the language of the Oxygen XML Editor user interface by creating an interface localization file:

1. Identify the code for the new language you want to translate the interface. It is composed from a language code (two or three lowercase letters that conform to the ISO 639 standard), followed by an underscore character, and a region code (two or three uppercase letters that conform to the ISO 3166 standard).
2. Write an email to the Oxygen XML Editor support team and ask them to send you the translation.xml sample file.
3. Open the translation.xml file in Oxygen XML Editor. The file contains all the interface messages that can be translated and is updated at every new release with the latest additions. Here is a small sample of its content:

```xml
<translation>
  <languageList>
    <language description="English" lang="en_US"/>
  </languageList>
  <key value="New">
    <comment>The File/New action. Creates a new document.</comment>
    <val lang="en_US">New</val>
  </key>
  <key value="New_folder">
    <comment>Creates a folder in the Project View.</comment>
    <val lang="en_US">New Folder</val>
  </key>
  .....
</translation>
```

4. Update the `language` element to reflect the new language. For example, set the @description attribute to Spanish and the @lang attribute to es_ES.
5. For each `<key>` element, translate the `<comment>` (optional) and `<val>` elements. For example, set the @lang attribute to es_ES.
Note:

After you are finished, the file should look like this:

```xml
<translation>
  <languageList>
    <language description="Español" lang="es_ES"/>
  </languageList>
  <key value="New">
    <comment>El Archivo / Nueva acción. Crea un nuevo documento.</comment>
    <val lang="es_ES">Nuevo</val>
  </key>
  <key value="New_folder">
    <comment>Crea una carpeta en la vista del proyecto.</comment>
    <val lang="es_ES">Nueva carpeta</val>
  </key>
  ..... 
</translation>
```

6. Open the Preferences dialog box (Options > Preferences) (on page 124), go to Global, and select the Other language option (on page 126). Browse for the translation.xml file.

7. Restart the application.

Adding New Languages to the Interface

Oxygen XML Editor provides a plugin extension is available in the Oxygen SDK that provides the ability to contribute a new translation language to the interface. By using this plugin extension, you can bundle the new language translation and that new language will be available in the Languages drop-down menu in the Options > Preferences > Global preferences page (on page 126).

Setting a Java Virtual Machine Parameter when Launching Oxygen XML Editor

You can set Java Virtual Machine parameters (for example, if you want to increase the maximum amount of memory available) for the Oxygen XML Editor application launchers (on page 339) or command-line scripts (on page 342). You can also create a custom startup parameters file (on page 342).

Setting Parameters for the Application Launchers

Increasing the Amount of Memory that Oxygen XML Editor Uses on Windows and Linux

For Windows and Linux installations of Oxygen XML Editor, the startup launchers for the application and its executable internal tools (Tree Editor, XML Schema Regular Expressions Builder, Large File Viewer, SVN Client, Compare Directories, and Compare Files) include a default .vmoptions file in the installation directory that contains some startup parameters (such the -Xmx parameter, which is used for allocating
memory for that particular application). If your installation contains these .vmoptions files, you can edit the parameters in them so that the applications will launch with your desired values. However, if you re-install the application, install an update for the application, or deploy it to other users or machines, those parameters will be reset to their default values.

To increase the memory available to the Oxygen XML Editor application on Windows:

1. Browse the installation directory of Oxygen XML Editor.
2. Locate the -Xmx parameter in the oxygen24.1.vmoptions file. If it is located in a directory where you do not have write access, copy the file to another folder (where you do have write access), modify it there, and then copy it back to the original location.

**Note:**
The parameters from the .vmoptions file are used when you start Oxygen XML Editor with the oxygen launcher (or with the desktop shortcut). If you use the command-line script (<oxygen.bat> or <oxygen.sh>) to launch Oxygen XML Editor, modify the -Xmx parameter in that script file.

**Tip:**
For 32-bit Windows, modify the parameter to -Xmx1024m or larger, but not over -Xmx1200m. Make sure you do not exceed your physical RAM. For 64-bit Windows modify the parameter to a larger value (for example, -Xmx2048m). It is recommended to not use more than half of your existing physical RAM.

3. Restart Oxygen XML Editor. Go to Help > About and verify the amount of memory that is actually available (see the JVM Memory Used in the last row in the Copyright tab). If Oxygen XML Editor does not start and you receive and error message saying that it could not start the JVM, decrease the -Xmx parameter and try again.

**Increasing the Amount of Memory that Oxygen XML Editor Uses on macOS**

To increase the memory available to Oxygen XML Editor on macOS:

1. Create a file named vmoptions.txt.
2. Add the -Xmx argument (or other Java VM arguments). For example:

   -Xmx=2048m

3. Make sure you save the file as plain text (in the TextEdit, go to Menu > Format > Make plain text) and copy the file to the contents folder within the application installation folder.

**Setting a System Property**

Depending on the operating system and type of installer, you can set a Java system property in multiple ways:
• **[Windows/Linux Installer]** When installing the application on Windows or Linux using the provided installation kit, you can create your own custom startup parameters file *(on page 342)* in the installation folder.

• **[macOS Installer]** Create a file named `vmoptions.txt` in the `Contents` folder within the application installation folder, similarly to this procedure *(on page 340)*. Add each system property (or Java VM argument) on a separate line. For example:

```
-DproertyName1=value1
-DproertyName2=value2
```

• **[Windows Linux/Mac Startup Scripts]** The application also contains startup scripts in the installation folder. If you are using such scripts to start the application, you can follow this procedure to set system properties for them: Setting Parameters in the Command-Line Scripts *(on page 342)*.

  **Note:**
  You can also set a system property through a parameter prefixed with `-Doxy` in the command line used to start the application:

  `oxygen20.1.exe "-Doxyproperty.name=value"`

  but this system property will be set immediately after the application starts and might not be available if it is needed sooner.

To check the value for a system property, you can select **Help > About** from the main menu and look in the **System properties** tab.

To view the list of Oxygen XML Editor system properties, go to Custom System Properties *(on page 333)*.

**Disabling DPI Scaling**

Some users may prefer the look of smaller icons in an HiDPI display. To achieve this, display scaling needs to be disabled for high DPI settings. To disable the DPI scaling, set the following property *(on page 340)*:

```
sun.java2d.dpiaware=false
```

**Setting Environment Variables**

When started, the application inherits and can access all environment variables set in the operating system. All processes started by the application (for example, publishing using the DITA Open Toolkit engine or starting external tools) also inherit the environment variables provided to the application. Depending on the operating system, environment variables can be set in various ways:

• **[Windows]** (Note: You will need Administrator permissions or to work with a system administrator):
  1. Go to **Start > Edit the system environment variables > Environment Variables**.
  2. Click **New** in the **System variables** section.
  3. Specify the variable name and value in the **Name** and **Value** fields.
4. Click OK.
5. Restart Windows.

- **[Linux]:**
  
  1. Append the following line to the `/etc/environment` file:

  ```
  ENV_VAR_NAME=VALUE
  ```

  2. Reboot the computer.

- **[macOS]:** There is no standard way to set an environment variable so that it is inherited by the applications regardless of the way they start.

To check the value for an environmental variable, you can select Help > About from the main menu and look in the System properties tab.

### Setting Parameters in the Command-Line Scripts

If you start Oxygen XML Editor with a command-line script (`oxygen.bat/oxygen.sh`), you have to add or modify parameters in the java command at the end of the script.

For example, to set the maximum amount of Java memory to 600 MB in **Windows**, modify the `-Xmx` parameter like this:

```java
java -Xmx600m -Dsun.java2d.noddraw=true ...
```

On **macOS**, the java command should look like this:

```java
java "-Xdock:name=Oxygen"
-Dcom.oxygenxml.editor.plugins.dir="$OXYGEN_HOME/plugins"
-Xmx600m
...
```

On **Linux**, the Java command should look like this:

```java
java -Xmx600m
"-Dcom.oxygenxml.editor.plugins.dir="$OXYGEN_HOME/plugins"
```

### Creating Custom Startup Parameters File

You can create your own custom `.vmoptions` file and the application and the executable tools will automatically include your custom parameters at startup. The following custom files are recognized by the application and the executable tools:

- **custom_commons.vmoptions** - The parameters and their values of this file will be included in all the startup launchers.
- **custom_<app name>.vmoptions** - The `<app name>` is the name of the executable application or tool (for example, `custom_diffFiles.vmoptions` for the Compare Files tool). The parameters and their values of this file will be included in the startup launcher for this particular executable.
For example: To specify a different language for all launchers you can use the custom `vmoptions` file called `custom_commons.vmoptions` and the content would look like this:

```bash
-Dcom.oxygenxml.language=French
```

For example: To increase the memory available for a specific tool, such as the Compare Files tool (diffFiles.exe), you can use a custom `vmoptions` file called `custom_diffFiles.vmoptions` and the content would look like this:

```bash
-Xmx1000m
```

To be recognized and included, these custom startup parameter files must be saved in the installation directory of Oxygen XML Editor.

**How to Increase the Amount of Available Memory**

Determining how to increase the amount of memory that is allocated to Oxygen XML Editor depends on how you launch the application.

- **Windows/Linux Application Launcher** - If you start Oxygen XML Editor using the default startup launcher that was created during a Windows or Linux installation, see Increasing the Amount of Memory that Oxygen XML Editor Uses on Windows and Linux (on page 339).

- **macOS Application Launcher** - If you start Oxygen XML Editor using the default startup launcher that was created during a macOS installation, see Increasing the Amount of Memory that Oxygen XML Editor Uses on macOS (on page 340).

- **Command-Line Script** - If you start Oxygen XML Editor using a command-line script, see Setting Parameters in the Command-Line Scripts (on page 342).

- **Custom Startup Parameters File** - You can also create your own custom startup parameters file and increase the memory using this file. For more information, see Creating Custom Startup Parameters File (on page 342).
5. Perspectives

An Oxygen XML Editor perspective (on page 3221) is a layout geared towards a specific use. The Oxygen XML Editor interface uses standard interface conventions and components to provide a familiar and intuitive editing environment across all operating systems. There are several perspectives that you can use to work with documents in Oxygen XML Editor. You can change the perspective by selecting the respective icon ( ) in the top-right corner of Oxygen XML Editor or by selecting the perspective from the Window > Open Perspective menu.

Editor Perspective

The Editor perspective (on page 3221) is the most commonly used perspective and it is the default perspective when you start Oxygen XML Editor for the first time. It is the perspective that you will use to edit the content of your XML documents.

To switch the focus to this perspective, select the Editor button in the top-right corner of Oxygen XML Editor (or select Editor from the Window > Open perspective menu).

The layout of this perspective is composed of the following components:

- **Menus**
  Provides menu driven access to all the features and functions available in Oxygen XML Editor. Most of the menus are common for all types of documents. However, Oxygen XML Editor also includes some context-sensitive and framework (on page 3219)-specific menus that are only available for a specific context or type of document.

- **Toolbars**
  Provides easy access to common and frequently used functions. Each icon is a button that acts as a shortcut to a related function. Most of the toolbars are common for all types of documents. However, Author mode also includes framework (on page 3219)-specific toolbars, depending on the type of document that is being edited (for example, if you are editing a DITA document, a DITA Author Custom Actions toolbar is available that includes operations that are specific to DITA documents). The toolbars can be configured (on page 365) to suit your specific needs.

- **Editor Pane**
  The main editing pane where you spend most of your time reading, editing, applying markup, and validating your documents.

- **Views**
Oxygen XML Editor includes a large variety of *dockable* views to assist you with editing, viewing, searching, validating, transforming, and organizing your documents. The most commonly used views are displayed by default and you can choose to display others by selecting them from the **Window > Show View** menu. The layout of the views can also be configured according to your preferences.

When two or more views are displayed, the application provides divider bars. Divider bars can be dragged to a new position increasing the space occupied by one panel while decreasing it for the other.

As the majority of the work process centers around the Editor area, other views can be hidden using the toggle controls located on the top corner of the view (⌘ on macOS).

Some of the most helpful views in the **Editor perspective** include the following:

- **Project view (on page 402)** - Enables the definition of projects and logical management of the documents they contain.
- **DITA Maps Manager view (on page 2897)** - For DITA document types, this view helps you organize, manage, and edit DITA topics and maps.
- **Open/Find Resource view (on page 422)** - Designed to offer advanced search capabilities in various scopes.
- **Outline view (on page 539)** - It provides an XML tag overview and offers a variety of functions, such as modifications follow-up, document structure change, document tag selection, and elements filtering.
- **Results view (on page 548)** - Displays the messages generated as a result of user actions such as validations (on page 772), transformation scenarios (on page 1432), spell checking in multiple files (on page 458), search operations, and others. Each message is a link to the location related to the event that triggered the message.
- **Attributes view (on page 542)** - Presents all possible attributes of the current element and allows you to edit attribute values. You can also use this view to insert attributes in **Text** mode. **Author** mode also includes an in-place attribute editor (on page 630).
- **Model view (on page 545)** - Presents the currently edited element structure model and additional documentation as defined in the schema.
- **Elements view (on page 546)** - Presents a list of all defined elements that you can insert at the current cursor position according to the document's schema. In **Author mode this view (on page 633)** includes tabs that present additional information relative to the cursor location.
- **Entities view (on page 547)** - Displays a list with all entities declared in the current document as well as built-in ones.
- **Transformation Scenarios view (on page 1555)** - Displays a list with all currently configured transformation scenarios.
• **XPath/XQuery Builder view (on page 2002)** - Displays the results from running an XPath expression.

• **WSDL SOAP Analyzer view (on page 1072)** - Provides a tool that helps you test if the messages defined in a Web Service Descriptor (WSDL) are accepted by a Web Services server.

### DITA Perspective

The DITA perspective (on page 3221) provides an editing environment with default side-views and other interface components that are optimal for working with DITA projects. To switch the focus to this perspective, select the DITA button in the top-right corner of Oxygen XML Editor or select DITA from the Window > Open perspective menu. If you open a DITA resource from the DITA Maps Manager (on page 2897) while in another perspective, a message will appear asking if you want to switch to the DITA perspective.

The layout of this perspective is composed of the following components:

**Menus**

Most of the menus are common for all types of documents, but this perspective also include DITA-specific actions in the DITA Maps and DITA menus.

**Toolbars**

Many of the toolbar buttons are common for all types of documents, but Author mode also includes DITA-specific toolbar actions. The toolbars can be configured (on page 365) to suit your specific needs.

**Editor Pane**

The main editing pane where you spend most of your time reading, editing, applying markup, and validating your documents.

**Views**

Oxygen XML Editor includes a large variety of dockable (on page 3218) views to assist you with editing, viewing, searching, validating, transforming, and organizing your documents. By default, this perspective displays the most commonly used views for DITA users and you can choose to display others by selecting them from the Window > Show View menu. The layout of the views can also be configured (on page 360) according to your preferences.

Some of the most helpful views in the DITA perspective include the following:
• **DITA Maps Manager view (on page 2897)** - This view helps you organize, manage, and edit DITA maps.

• **DITA Reusable Components view (on page 3061)** - This view is helpful if you use a large amount of keys or reusable components in your DITA project. It collects all of the keys and reusable components that are defined in the root map (on page 3223) and presents them in several tabs. It includes various features to make it easy to locate and insert references to the reusable content.

• **DITA Resource Hierarchy/Dependencies view (on page 3183)** - This view displays the hierarchy or dependencies for resources that are directly referenced in the DITA topic.

• **Project view (on page 402)** - Enables the definition of projects and logical management of the documents they contain.

• **Open/Find Resource view (on page 422)** - Designed to offer advanced search capabilities in various scopes.

• **Outline view (on page 539)** - It provides an XML tag overview and offers a variety of functions, such as modifications follow-up, document structure change, document tag selection, and elements filtering.

• **Results view (on page 548)** - Displays the messages generated as a result of user actions such as validations (on page 772), transformation scenarios (on page 1432), spell checking in multiple files (on page 458), search operations, and others. Each message is a link to the location related to the event that triggered the message.

• **Attributes view (on page 542)** - Presents all possible attributes of the current element and allows you to edit attribute values. You can also use this view to insert attributes in Text mode. Author mode also includes an in-place attribute editor (on page 630).

• **Elements view (on page 546)** - Presents a list of all defined elements that you can insert at the current cursor position according to the document's schema. In Author mode, this view (on page 633) includes tabs that present additional information relative to the cursor location.

### XSLT Debugger Perspective

The XSLT Debugger perspective (on page 3221) allows you to detect problems in an XSLT transformation by executing the process step by step in a controlled environment. To switch the focus to this perspective, select the XSLT Debugger button in the top-right corner of the interface or Window > Open perspective > XSLT Debugger.

The workspace in this perspective is organized as an editing area assisted by special helper views. The editing area contains editor panels that you can split horizontally or vertically (on page 256) in a stack of XML editor panels and a stack of XSLT editor panels. The XML files and XSL files can be edited in Text mode (on page 353) only.

The layout of this perspective is composed of the following components:

**Menus**

Provides menu driven access to all the features and functions available in the XSLT Debugger.
Toolbars

Contains all actions needed to configure and control the debugging process. The toolbars can be configured (on page 365) to suit your specific needs.

XML Source Pane

The editing pane where you can display and edit data or document-oriented XML documents.

XSL Source Pane

The editing pane where you can display and edit XSL stylesheets.

Output View

Displays the transformed output that results from the input of a selected document (XML) and selected stylesheet (XSL) to the transformer. The result of transformation is dynamically written as the transformation is processed. There are three types of views for the output: a text view (with XML syntax highlight), an XHTML view, and one text view for each `<xsl:result-document>` element used in the stylesheet (if it is an XSLT 2.0 / 3.0 stylesheet).

Debugging Information Views (on page 2108)

Presented in two panes, they display various types of information that can be used to understand the transformation process. For each information type there is a corresponding tab. While running a transformation, relevant events are displayed in the various information views. This allows you to obtain a clear view of the transformation progress. See the Debugging Information Views (on page 2108) topic for a list of all the information views (and links to more details on each view).

Note:

You can add XPath expression automatically in the XWatch view using the Watch expression action from the contextual menu. In case you select an expression or a fragment of it and then click Watch expression in the contextual menu, the entire selection is presented in the XWatch view. Using Watch expression without selecting an expression displays the value of the attribute from the cursor position in the XWatch view. Variables detected at the cursor position are also displayed. Expressions displayed in the XWatch view are normalized (unnecessary white spaces are removed from the expression).

For more information about the XSLT debugging capabilities in Oxygen XML Editor, watch our video demonstration:

https://www.youtube.com/embed/m9d8c4V-LJw

Related information

Debugging XSLT Stylesheets and XQuery Documents (on page 2102)
XQuery Debugger Perspective (on page 349)
XQuery Debugger Perspective

The XQuery Debugger perspective (on page 3221) allows you to detect problems in an XQuery transformation process by executing the process step by step in a controlled environment and inspecting the information provided in the special views. To switch the focus to this perspective, select the XQuery Debugger button in the top-right corner of the interface or Window > Open perspective > XQuery Debugger.

The workspace in this perspective is organized as an editing area assisted by special helper views. The editing area contains editor panels that you can split horizontally or vertically (on page 256) in a stack of XML editor panels and a stack of XQuery editor panels. The XML files and XQuery files can be edited in Text mode (on page 353) only.

The layout of this perspective is composed of the following components:

- **Menus**
  Provides menu driven access to all the features and functions available in the XQuery Debugger.

- **Toolbars**
  Contains all actions needed to configure and control the debugging process. The toolbars can be configured (on page 365) to suit your specific needs.

- **XML Source Pane**
  The editing pane where you can display and edit data or document-oriented XML documents.

- **XQuery Source Pane**
  The editing pane where you can display and edit XQuery files.

- **Output View**
  Displays the transformed output that results from the input of a selected document (XML) and selected XQuery document to the XQuery transformer. The result of transformation is dynamically written as the transformation is processed. There are two types of views for the output: a text view (with XML syntax highlight) and an XHTML view.

- **Debugging Information Views (on page 2108)**
  Presented in two panes, they display various types of information that can be used to understand the transformation process. For each information type there is a corresponding tab. While running a transformation, relevant events are displayed in the various information views. This allows you to obtain a clear view of the transformation progress. See the Debugging Information Views (on page 2108) topic for a list of all the information views (and links to more details on each view).

**Note:**
You can add XPath expression automatically in the XWatch view using the Watch expression action from the contextual menu. If you select an expression, or a fragment of it, and then click Watch expression in the contextual menu, the entire selection is
presented in the **XWatch** view. Expressions displayed in the **XWatch** view are normalized (unnecessary white spaces are removed from the expression).

For more information about the XQuery debugging capabilities in Oxygen XML Editor, watch our video demonstration:

https://www.youtube.com/embed/o5_M2kbyipU

**Related information**
- Debugging XSLT Stylesheets and XQuery Documents *on page 2102*
- XSLT Debugger Perspective *on page 347*

## Database Perspective

The **Database perspective** *on page 3221* allows you to manage databases. To switch the focus to this perspective, select the ![Database](database_icon.png) button in the top-right corner of Oxygen XML Editor or **Window > Open perspective > Database** from the **Window > Open perspective** menu.

The **Database perspective** offers various helpful features, including:

- Support for browsing multiple connections at the same time.
- Support for both *Relational* and *Native XML* databases.
- Browsing the structure of databases.
- Viewing tables from databases.
- Inspecting or modifying data.
- Specifying XML Schemas for XML fields.
- SQL execution.
- XQuery execution.
- Data export to XML.

### Supported Databases

Oxygen XML Editor supports numerous types of databases, including:

- eXist XML Database
- IBM DB2 (Enterprise edition only)
- JDBC-ODBC Bridge
- MarkLogic (Enterprise edition only)
- MySQL
- Oracle 11g (Enterprise edition only)
- PostgreSQL (Enterprise edition only)
- SharePoint (CMS)
- Oracle Berkeley DB XML Database *(Deprecated)*
- Microsoft SQL Server 2005 and Microsoft SQL Server 2008 (Enterprise edition only) *(Deprecated)*
Note:
For the databases marked with "Enterprise edition only", the XML capabilities are only available in the Enterprise edition of Oxygen XML Editor. For a detailed feature matrix that compares the Academic, Professional, and Enterprise editions of Oxygen XML Editor go to the Oxygen XML Editor website.

Supported Capabilities

The supported non-XML capabilities are as follows:

• Browsing the structure of the database instance.
• Opening a database table in the Table Explorer view (on page 2017).
• Handling the values from XML Type columns as String values.

Note:
The non-XML capabilities are available in the Enterprise, Academic, and Professional editions of Oxygen XML Editor by registering the database driver as a Generic JDBC type driver when defining the data source for accessing the database. For more information, see Database Connection Support (on page 2020).

The supported XML capabilities are as follows:

• Displaying an XML Schema node in the tree of the database structure (for databases with an XML-specific structure) with actions for opening, editing, and validating the schemas in an Oxygen XML Editor editor panel.
• Handling the values from XML Type columns as XML instance documents that can be opened and edited in an Oxygen XML Editor editor panel.
• Validating an XML instance document added to an XML Type (column of a table, etc.)

Tip:
Connections configured on relational data sources can be used to import data to XML or to generate XML schemas.

Layout of the Database Perspective

The layout of this perspective is composed of the following components:

Menus

Provides menu driven access to all the features and functions available in the perspective.

Toolbars

Contains all actions needed to configure and control the debugging process. The toolbars can be configured (on page 365) to suit your specific needs.

Editor Pane
The main editing pane where you spend most of your time reading, editing, applying markup, and validating your documents.

**Data Source Explorer View (on page 2015)**

Provides browsing support for the configured connections.

**Table Explorer View (on page 2017)**

Provides table content editing support for inserting new rows, deleting table rows, editing cell values, exporting to an XML file, and more.

**Related information**

- Working with Databases (on page 2015)
- Data Source Explorer View (on page 2015)
- Table Explorer View (on page 2017)
6. Editing Modes

The main editing area in Oxygen XML Editor includes several editing modes to suit the type of editing that you want to perform. You can easily switch between modes by clicking on the desired mode at the bottom of the main editing pane. Oxygen XML Editor offers the following editing modes:

- **Text** *(on page 353)* - This mode presents the source of an XML document.
- **Grid** *(on page 354)* - This mode displays an XML document as a structured grid of nested tables.
- **Author** *(on page 354)* - This mode enables you to edit in a WYSIWYG-like editor.
- **Design** *(on page 355)* - This mode is found in the schema editor and represents the schema as a diagram.

The default editing mode that will be initially opened for each type of document can be set in two ways:

- If the *Allow Document Type specific edit mode setting to override the general mode setting* option *(on page 171)* is selected in the Edit Modes preferences page, then the edit mode specified in the *Document Type* configuration dialog box *(on page 140)* is used when that particular type of document is initially opened.
- If the *Allow Document Type specific edit mode setting to override the general mode setting* option *(on page 171)* is not selected, then the edit mode specified in the table in the Edit Modes preferences page *(on page 171)* is used when that particular type of document is initially opened.

**Text Editing Mode**

The **Text** mode editor in Oxygen XML Editor is designed to be a simple, yet powerful, XML source editor. It provides support to help you edit, transform, and debug XML-based documents. It is similar to other common text editors, but Oxygen XML Editor also includes specialized editing actions, a powerful *Content Completion Assistant* *(on page 532)*, and many other unique features.

To switch to this mode, select **Text** at the bottom of the editing area.

For more information about working with XML documents in **Text** mode and all of the details about its features, see the *Editing XML Documents in Text Mode* section *(on page 517)*.

**Related information**

- *Editing XML Documents in Text Mode* *(on page 517)*
Grid Editing Mode

The Oxygen XML Editor Grid editing mode displays the XML document as a structured grid of nested tables where the text content can be modified without directly interacting with the XML markup. This is helpful for non-technical users who want to edit text content without modifying the XML markup. You can easily expand or collapse elements within the table and the document structure can be changed with simple drag/drop or copy/paste operations.

To switch to this mode, select Grid at the bottom of the editing area.

Figure 50. Grid Editing Mode

For more information about working with XML documents in Grid mode and all of the details about its features, see the Editing XML Documents in Grid Mode section (on page 578).

Related information

Editing XML Documents in Grid Mode (on page 578)

Author Editing Mode

The Author editing mode in Oxygen XML Editor allows you to visually edit XML documents in a user-friendly interface that is similar to a WYSIWYG word processor. Oxygen XML Editor provides support for visually editing the most commonly used XML vocabularies in Author mode, including DITA, DocBook, TEI, and XHTML. Adding text content is as simple as doing so in a standard text editor but the content is rendered similar to how you will see it in the output. Tables, images, and media objects (such as videos) are also rendered comparable to the output.

To switch to this mode, click the Author button at the bottom of the editing area.

For more information about working with XML documents in Author mode and all of the details about its features, see the Editing XML Documents in Author Mode section (on page 588).
Design Editing Mode (Schema Diagram Editor)

Schemas allow document designers to specify the allowed structure and content of a document and to check if it is valid. Oxygen XML Editor provides a simple and expressive schema diagram editor (Design mode) for editing XML schemas or JSON schemas. The schema diagram helps both the content authors who want to understand a schema and schema designers who develop complex schemas.

The Design mode offers a diagram view of the schema document by rendering all the schema components. You can edit component features directly within the diagram (for instance, the component name, its type, etc.), you can quickly navigate to the referenced definitions (elements, attributes, types, groups, etc.), and you can use drag-and-drop operations to move, copy, or make references. It also features some specialized helper views (such as the Palette view (on page 949) and Facets view (on page 951)) to further enhance the diagram editor, various contextual menu actions, validation support, and much more.

To switch to this mode, select Design at the bottom of the editing area.
For more information about designing and editing XML schemas, and all the details about the features that are available in the Oxygen XML Editor Design mode for XML schema documents, see the Editing XML Schemas section (on page 947) and the XML Schema Design Mode (XML Schema Diagram Editor) subsection (on page 948).

JSON Schema Resources

For more information about designing and editing JSON schemas, and all the details about the features that are available in the Oxygen XML Editor Design mode for JSON schema documents, see the Editing JSON
Schemas section (on page 1146) and the JSON Schema Design Mode (XML Schema Diagram Editor) subsection (on page 1149).

Related information

Editing XML Schemas (on page 947)
XML Schema Design Mode (XML Schema Diagram Editor) (on page 948)
Editing JSON Schema Documents (on page 1146)
JSON Schema Design Mode (JSON Schema Diagram Editor) (on page 1149)
7. Working With Any Type of Document

Oxygen XML Editor includes a variety of general features that can be used when working with any type of document. More specialized features are available when working with specific type of documents, such as the various types of XML documents, CSS, JavaScript, Markdown, and more. For details about those specialized features for specific types of documents, see Editing Documents (on page 516).

This chapter includes information about how to create and work with documents, working with projects, and various editing features that are provided in Oxygen XML Editor for all document types. This chapter also includes information about some of the special tools that are included in Oxygen XML Editor, such as the file and directory comparison tools.

Getting Familiar with the Interface

Oxygen XML Editor includes several perspectives (on page 3221) and editing modes (on page 353) to help you accomplish a wide range of tasks. Each perspective and editing mode also includes a large variety of helper views, menu actions, toolbars, and contextual menu functions.

There are various ways that you can configure the layout of the views or editors (on page 360), and you can customize the toolbars (on page 365).

Regardless of the perspective (on page 3221) or editing mode (on page 353) that you are working with, the default layout consists of the following areas:

**Menus**

Menu-driven access to all the features and functions available in Oxygen XML Editor. Most of the menus are common for all types of documents, but Oxygen XML Editor also includes some context-sensitive and framework-specific menus and actions that are only available for a specific context or type of document.

**Toolbars**

Easy access to common and frequently used functions. Each icon is a button that acts as a shortcut to a related function. Some of the toolbars are common for all perspectives, editing modes, and types of documents, while others are specific to the particular perspective or mode. Some toolbars are also framework-specific, depending on the type of document that is being edited. All the toolbars can be configured (on page 365) to suit your specific needs.

**Helper Views**

Oxygen XML Editor includes a large variety of dockable (on page 3218) views to assist you with editing, viewing, searching, validating, transforming, and organizing your documents. Many of the views also contain useful contextual menu actions, toolbar buttons, or menus. The most
commonly used views for each *perspective* and editing mode are displayed by default and you can choose to display others to suit your specific needs. The layout of the views can also be configured (on page 360) according to your preferences.

**Editor Pane**

The main editing area in the center of the application. Each *editing mode* (on page 353) provides a main editor pane where you spend most of your time reading, editing, applying markup, and validating your documents. The editor pane in each *editing mode* (on page 353) also includes various contextual menu actions and other features to help streamline your editing tasks. Each file that has been opened has a tab at the top of the editing pane and there are several ways to switch between tabs or move them (on page 393).

**Perspectives**

Oxygen XML Editor includes several different perspectives (on page 344) that you can use to work with your documents. The Editor perspective is the most commonly used perspective used for displaying and editing the content of your XML documents, and it is the default perspective when you start Oxygen XML Editor for the first time. Oxygen XML Editor also includes a Database perspective that allows you to manage databases and their connections and a few debugging perspectives that allow you to detect problems in XSLT or XQuery transformations.

**Status Bar**

The status bar at the bottom of the application contains some useful information when you are working with documents. It includes the following information, in the order it is displayed from left to right:

- The path of the current document.
- The *Unicode value* (on page 463) for the character directly to the right of the current cursor position.
- The status of the current document. The status of Modified is displayed for documents that have not yet been saved. Otherwise, this section is left blank.
- In Text editing mode (on page 353), the current line and character position is displayed.
- If the **Check for notifications** option (on page 126) is selected, this section will show you when new messages have been received. The types of messages include the addition of new videos on the Oxygen XML Editor website, the announcement of upcoming webinars and conferences where the Oxygen XML Editor team will participate, and more.
- The memory consumption, including the memory used by the application and the maximum amount that is allocated to the application.
- If the **Show memory status** option (on page 128) is selected, a Free unused memory icon is displayed in the bottom-right corner and you can use this icon to free up unused memory.
Configuring the Layout of the Views and Editors

All of the side-views available in Oxygen XML Editor are dockable (on page 3218) and there are various ways to configure and arrange the layout of the views and editing panes. You can also configure the layout of the toolbars (on page 365).

To open a view, select it from the Window > Show View menu. You can hide a view by closing it with the X button at the top-right corner of the view, or with the Window > Hide current view action.

Arranging the Layout

You can drag any view to any margin of another view or editor inside the Oxygen XML Editor window. Once you create a layout that suits your needs, you can save it from Window > Export Layout. Oxygen XML Editor creates a layout file containing the preferences of the saved layout. To load a layout, go to Window > Load Layout. To reset it, select Window > Reset Layout.

Note:
The Load Layout menu lets you select between the default layout, a predefined layout, or a custom layout. The changes you make using the Load Layout menu are also reflected in the Application Layout preferences page.

The changes you make to any layout are preserved between working sessions. The predefined layout files are saved in the preferences directory (on page 125) of Oxygen XML Editor.

You can drag the editors and arrange them in any order, both horizontally and vertically.

The following image presents two editors arranged as horizontal tiles. To arrange them vertically, drag one of them on top of the other. In the following example, the personal.xml file was dragged over the personal-schema.xml file:
Hide or Float Views

Hide

To gain more editing space in the Oxygen XML Editor window, click the **Toggle auto-hide** in any view. This button sets the view in the *auto-hide* state, making it visible only as a vertical tab, at the margins of the Oxygen XML Editor window. To display a view in the *auto-hide* state, hover its side-tab with your cursor, or click it to keep the view visible until you click elsewhere.

Float

A view can also be set to a floating state by using the **Toggle floating** action, making it independent from the rest of the Oxygen XML Editor window.

Maximize the Editing Environment

You can configure the interface to maximize the editing area, leaving more vertical screen space available for the main editing pane. This is, for example, useful for presentations on low-resolution screens or for laptops.
with small screen space. You can use the following two actions that are available in the Window menu to create a near full-screen editing environment:

Maximize Editor Area
If toggled on, all side views are minimized to give you more horizontal space in the editing pane.

Hide All Toolbars
If toggled on, all toolbar buttons are hidden to give you more vertical space in the interface.

Tile/Stack Editor Actions
You can also tile or stack all open editors using the following actions from the toolbar or Window menu:

Tile Editors Horizontally
Splits the editing area into horizontal tiles, one for each open file.

Tile Editors Vertically
Splits the editing area into vertical tiles, one for each open file.

Stack Editors
The reverse of the Tile Editors Horizontally/Vertically actions. Stacks all open editors.

Synchronous Scrolling
Select this action to scroll through the tiled editors at the same time.

Note:
When tiled, you can still drag and drop the editors, but note that they are docked in the same way as a window/view (instead of just tabs). You are actually rearranging the editor windows, so drag the editor tab and drop it to one of the sides of an editor (left/right/top/bottom). While dragging, you will see the dark gray rectangle aligned to one of the sides of the editor, or around the entire editor window. If you drop it to one of the sides, it will dock to that side of the editor. If you drop it when the rectangle is around the entire window of the editor, it will get stacked on top of that editor. You can also grab one of the stacked editors and tile it to one of the sides.

Split Editor Actions
You can divide the editing area vertically and horizontally using the following actions available in the toolbar and Window menu:

- Split Editor Horizontally - Splits the editor horizontally so that two editor panes are displayed with one on top of the other. This is useful for comparing and merging content between two documents.
- Split Editor Vertically - Splits the editor vertically so that two editor panes are displayed side by side. This is useful for comparing and merging content between two documents.
- Unsplit Editor - Removes a split action on the editing area.
To maximize or restore the editors, go to Window > Maximize Editing Area.

Switch, Move, or Hide Editor Tabs

Each file that has been opened has a tab at the top of the editing pane and there are several ways to switch between tabs or move them, and you can even hide the tabs to only show the currently open file.

**Note:**
If multiple file tabs are left open when you close the application, upon startup, Oxygen XML Editor will not load the file content until you switch to the corresponding file tab. The tabs remain visible as placeholders until the focus is switched to them. This helps to improve the application's startup time. If you want to disable this feature (meaning that the previously open files will all be re-loaded at startup), deselect the **Load file content only when switching to its corresponding editor tab** option in the Global preferences page (on page 127).

Switching Editor Tabs

You can switch between editor tabs by using any of the following methods:

**Mouse and Scroll Wheel**

Of course, you can switch to a different editor tab by left-clicking the tab with your mouse, but when there are too many open tabs to fit on the screen, you can hover over the tab stripe and use the scroll wheel on your mouse to scroll to the left or right (same as using the two arrows on the far-right of the tab stripe).

**Buttons on the Far-Right of the Tab Stripe (стрелки)**

You can use the arrow buttons (стрелки) on the right side of the tab stripe to scroll to the left or right and the **Show List** button opens a pop-up window that displays all the open file tabs and allows you to select and switch to a specific open file.

**Ctrl + Tab (Command + Tab on macOS)** [NOTE: Ctrl + Page Down (Ctrl + Alt + Right Arrow on macOS) does the same]

Switches to the next open tab in the order specified in the Order of switching between editor tabs option (on page 128).

**Ctrl + Shift + Tab (Command + Shift + Tab on macOS)** [NOTE: Ctrl + Page Up (Ctrl + Alt + Left Arrow on macOS) does the same]

Switches to the previous open tab in the order specified in the Order of switching between editor tabs option (on page 128).

**Window > Switch editor tab (Ctrl + F9 (Command + F9 on macOS))**

This action opens a dialog box that allows you to switch to a particular editor tab by selecting it from a filterable list. This is especially helpful when you have a large amount of open file tabs and you want to switch to a certain tab this is not shown on the screen. It includes a search filter field and several options to help you find specific open file tabs.
The **Switch Editor Tab** dialog box contains the following options and features:

**Search Filter**

You can enter text in the filter field at the top of the dialog box to filter the list and search for specific open files. You can enter any number of terms, separated by space, and wildcards are allowed (for example, * to match any sequence of characters, or ? to match a single character). This field also has a history drop-down that allows you to select previously used search terms.

**Match all terms**

If this option is selected, only the files that match all of your search terms will be displayed. If you use a wildcard in the search filter, this option is automatically disabled.

**Include file paths**

If this option is selected, the search is expanded to include file paths, and also the paths are displayed in this dialog box.

**Case sensitive**

If this option is selected, the search operation will be case-sensitive.

**List of Open File Tabs**
All files that are currently open are displayed in the upper part of the main pane of the dialog box, followed by recently closed files. Files that have been modified but not yet saved are prefixed by an asterisk. To switch to a particular file tab, double-click the file or select it and click OK.

### Moving Editor Tabs

You can move editor tabs by using any of the following methods:

**Mouse Drag**

You can use your mouse to drag editor tabs to a new location on the tab stripe.

**Ctrl + Alt + Comma**

Moves the current file tab one position to the left.

**Ctrl + Alt + Period**

Moves the current file tab one position to the right.

### Hiding Editor Tabs

If you want to hide all the file tabs and only show the currently open file, select Hide editor tabs from the Window menu. This does not close the other tabs, just hides them. You can still navigate between tabs using keyboard shortcuts (Ctrl + Tab, Ctrl + Shift + Tab, Ctrl + F6, Ctrl + Shift + F6) or by selecting Next editor or Previous editor from the Window menu.

For more information about configuring the interface of Oxygen XML Editor, watch our video demonstration:

https://www.youtube.com/embed/anwjepfAdEk

---

**Tip:**

To get more ideas for more advanced customization possibilities, watch our Webinar: [Working with DITA in Oxygen - Customizing the Editing Experience](https://www.youtube.com/embed/anwjepfAdEk). It offers a visual demonstration of how to customize actions, document validation, content completion, new document templates, Author mode rendering, and more.

---

**Related information**

Configuring Toolbars (on page 365)

---

### Configuring Toolbars

You can configure the toolbars in Oxygen XML Editor to personalize the interface for your specific needs. You can choose which toolbars to show or hide in the current editor mode (Text, Author, Design, or Grid) and in the current perspective (on page 3221) (Editor, XSLT Debugger, XQuery Debugger, or Database). You can also choose which actions to display in each toolbar, add actions to toolbars, and customize the layout of the toolbars.
To configure the toolbars, open the **Configure Toolbars** dialog box by doing one of the following:

- Right-click any toolbar and select **Configure Toolbars**.
- Select **Configure Toolbars** from the **Window** menu.

**Figure 54. Configure Toolbars Dialog Box**

The **Configure Toolbars** dialog box provides the following features:

**Filter Text Box**

You can use the filter text box at the top of the dialog box to search for a specific toolbar or action.

**Show or Hide Toolbars**

You can choose whether to show or hide a toolbar by using the checkbox next to the toolbar name. This checkbox is only available for toolbars that are available for the current **perspective (on page 3221)** and editing mode.

**Show or Hide Actions in a Toolbar**

To show or hide actions in a toolbar, expand it by clicking the arrow next to the toolbar name, then use the checkbox to select or deselect the appropriate actions. The toolbar configuration changes in the **Preview** column according to your changes.

**Add Actions to a Toolbar**

Use the **Add Actions** button to open the **Add Actions** dialog box that displays all the actions that can be added to any of the toolbars, with the exception of those that are contributed from **frameworks (on page 3219)** or 3rd party **plugins (on page 3222)**.

**Remove Actions from a Toolbar**
You can remove actions that you have previously added to toolbars by using the **Remove Action** button.

**Move Actions in a Toolbar**

Use the **Move Up** and **Move Down** actions to change the order of the actions in a toolbar.

The **Configure Toolbars** dialog box also provides a variety of other ways to customize the layout in Oxygen XML Editor.

**Customize My Toolbar**

You can customize the **My Toolbar** to include your most commonly used actions. By default, this toolbar is listed first. Also, it is hidden until you add actions to it and you can easily hide it with the **Hide "My Toolbar" Toolbar** action that is available when you right-click anywhere in the toolbar area.

**Drop-down Menu Actions**

Composite actions that are usually displayed as a drop-down menu can only be selected in one toolbar at a time. These actions are displayed in the **Configure Toolbars** dialog box with the name in brackets.

![Checkmark][1] [Validate]

**Configure External Tools Action**

There is a **Configure external tools** composite action that appears in the toolbar called **Tools**. It is a drop-down menu that contains any external tools that are configured in the **External Tools** preferences page.

![Note:][2]

If no external tools are configured, this drop-down menu is not shown in the toolbar.

Additional actions are available from the **Window** menu or contextual menu when invoked from a toolbar that allows you to further customize your layout. These actions include:

**Reset Toolbars**

To reset the layout of toolbars to the default setting, select the **Reset Toolbars** action from the contextual menu or **Window** menu.

**Reset Layout**

To reset the entire layout (including toolbars, editing modes, views, etc.) to the default setting, select **Reset Layout** from the contextual menu or **Window** menu.

**Export Layout**

You can use the **Export Layout** action that is available in the **Window** menu to export the entire layout of the application to share it with other users.
Hide Toolbars

You can use the **Hide Toolbar** action from the contextual menu to easily hide a displayed toolbar. When you right-click a toolbar it will be highlighted to show you which actions are included in that toolbar.

**Related information**

* Configuring the Layout of the Views and Editors ([on page 360](#))

**Creating, Opening, Saving, and Closing Documents**

Oxygen XML Editor includes various features, actions, and wizards to assist you with creating new files and working with existing files. This section explains many of these features, including information on creating new documents, opening, saving, and closing existing files, searching documents, viewing file properties, and more.

**Creating New Documents and Templates**

Oxygen XML Editor includes a helpful **New Document** wizard that allows you to customize and create new files from a large list of document types and built-in templates. You can also create your own templates ([on page 375](#)) and share them with others ([on page 381](#)).

To quickly create a new document:

1. Click the **New** button on the toolbar or select **File > New**.
2. Select the type of document that you want to create.

   **Tip:**
   
   You can use the text filter field at the top of the dialog box to search for a specific template.

3. Click the **Create** button at the bottom of the dialog box.

**New Document Wizard**

Oxygen XML Editor supports a wide range of document types. The **New Document** wizard presents the default associations between a file extension and the type of editor that opens the file. To customize these default associations, open the **Preferences** dialog box ([Options > Preferences](#)) and go to **File Types** ([on page 297](#)).

The **New Document** wizard creates a skeleton document that may contain a root element, the document prolog, and possibly other child elements depending on options that are specific for each schema type. You can also create your own custom document templates ([on page 375](#)) and select them from this wizard.
1. Click the **New** button on the toolbar or select **File > New**.

**Result:** The **New Document** wizard is displayed:

![New Document Wizard](image)

The first page of the wizard displays the supported document types and groups them in the following categories:

**Recently Used**

Contains the list of the most recently used file types. To clear the history of this folder, right-click an entry and select **Remove all** (or select an entry and press **Ctrl + Delete** on your keyboard). To remove a single entry, right-click and select **Remove** (or select the entry and press **Delete** on your keyboard).

**User-defined template directory**

You can add your own custom templates by creating template files *(on page 375)* in a directory and then adding that directory to the list of template directories that Oxygen XML Editor uses in the **Document Templates preferences page** *(on page 167)*. This user-defined directory will also appear in the **New Document** wizard.

**Popular**

Contains a list of popular framework templates. Each of these templates are marked as popular using a properties file that contains **popular** as one of the values for the **tags** property. For example, for the `OXYGEN_INSTALL_DIR/frameworks/dita/`
templates/topic/Topic.dita template, there is a Topic.properties sibling file that contains tags=popular. Other document templates can also be added to the Popular category by customizing them using a properties file for each one (on page 377).

**New Document**

Contains the list of all supported document types. This list includes XML, XSL, XML Schema, Document Type Definition, Relax NG Schema, XQuery, Web Services Definition Language, Schematron Schema, CSS, Text, PHP, JavaScript, Java, C, C++, Batch, Shell, Properties, SQL, XML Catalog, PERL, JSON, and more.

**Global Templates**

Contains the list of built-in templates as well as user-defined custom templates. You can create your own custom document templates (on page 375) and add them to the templates folder of the Oxygen XML Editor installation directory.

**Framework Templates**

Contains the list of templates defined in the Document Type configuration dialog box (Templates tab) (on page 163) for each framework.

2. Select the type of document that you want to create.

**Tip:**
You can use the text filter field at the top of the dialog box to search for a specific template.

3. If you want to change the default name and path of the file, select the **Save as** option and specify the file path (the Show "Save as" option to save newly created documents in the "New" document wizard option (on page 201) must be selected in the Save preferences page). Otherwise, the file will be opened in a new tab with a default untitled name and the document path will not yet exist until you save it.

**Note:**
For DITA documents, the dialog box includes some additional options for generating a title, file name, and root ID attribute. For more information, see Creating a New DITA Topic (on page 2961).

4. If you want to use the default settings in the creation process, select **Create** at the bottom of the dialog box.

**Result:** The document is created using the default settings and the new file is opened in the appropriate editor.
5. If you want to configure properties before creating the file, select **Customize**. This action is available for XML, XML Schema, Schematron, and XSL documents.

**Result:** A new file configuration dialog box is opened that allows you to customize various options, depending on the document type you selected. After configuring the options in this wizard, click **Create** to create the file and open it in the appropriate editor.

**XML Document Configuration Page**

Figure 56. New XML Document Configuration Wizard Page

If you selected **XML Document** for the type of file you want to create and selected the **Customize** option, the configuration dialog box will include the following options:

**Schema URL**

Specifies the path to the schema file. When you select a file, Oxygen XML Editor analyzes its content and tries to fill in the rest of the dialog box.

**Schema Type**

Allows you to select the schema type. The following options are available: XML Schema, DTD, RelaxNG XML syntax, RelaxNG compact syntax, and NVDL.

**Public ID**
Specifies the PUBLIC identifier declared in the document prolog.

**Namespace**

Specifies the document namespace.

**Prefix**

Specifies the prefix for the namespace of the document root.

**Root Element**

Populated with elements defined in the specified schema, enables selection of the element used as document root.

**Description**

A small description of the selected document root.

**Add Optional Content**

If you select this option, the elements and attributes defined in the XML Schema as optional are generated in the skeleton XML document.

**Add First Choice Particle**

If you select this option, Oxygen XML Editor generates the first element of an `<xs:choice>` schema element in the skeleton XML document. Oxygen XML Editor creates this document in a new editor panel when you click OK.

**XSLT Document Configuration Page**

*Figure 57. New XSLT Stylesheet Configuration Wizard Page*

If you selected XSLT Stylesheet for the type of file you want to create and selected the Customize option, the configuration dialog box will include the following options:

**Stylesheet version**
Allows you to select the **Stylesheet version** number. You can select from: 1.0, 2.0, and 3.0.

**Add documentation annotations**

Select this option to generate the stylesheet annotation documentation.

### XML Schema Document Configuration Page

**Figure 58. New XML Schema Configuration Wizard Page**

If you selected **XML Schema** for the type of file you want to create and selected the **Customize** option, the configuration dialog box will include the following options:

**Default XML Schema version**

Uses the XML Schema version defined in the **XML Schema** preferences page (on page 238).

**XML Schema 1.0**

Sets the `@minVersion` attribute to 1.0 and the `@maxVersion` attribute to 1.1.

**XML Schema 1.1**

Sets the `@minVersion` attribute to 1.1.

**Target namespace**

Allows you to specify the schema target namespace.
Namespace prefix declaration table

This table contains namespace prefix declarations. Table information can be managed using the + New and × Delete buttons.

Tip:
For further details on how you can set the version of an XML Schema, go to Setting the XML Schema Version (on page 1032).

Schematron Document Configuration Page

Figure 59. New Schematron Configuration Wizard Page

If you selected Schematron for the type of file you want to create and selected the Customize option, the configuration dialog box will include the following option:

Schematron version

Specifies the Schematron version. Possible options: 1.5 (deprecated) and ISO.

Note:
Starting with version 16.0 of Oxygen XML Editor, the support for Schematron 1.5 is deprecated. It is recommended to use ISO Schematron instead.
If you select JSON for the type of file you want to create and select the Customize option, the configuration dialog box will include the following options:

**Schema URL**

Specifies the path to a JSON Schema file that will be used to generate key-value pairs.

**Associate Schema in the Document**

If you select this option, the JSON instance will be generated with the JSON Schema associated directly in the document.

**Generate Optional Properties**

If you select this option, the JSON instance will be generated with optional properties that are defined in the JSON schema. Otherwise, only the required properties will be generated.

**Generate Additional Content**

If you select this option, the JSON instance will be generated with additional properties that are defined in the JSON schema as `additionalProperties` and additional items that are defined as `additionalItems` (in the case of an Array).

### Creating New Document Templates

Oxygen XML Editor allows you to create your own custom document templates and they will appear in the **New document wizard (on page 368)**.

### Creating a New Document Template

To create your own custom document template and have it appear in the new document wizard, follow these steps:
1. Create a new file (whatever type of document you need) and customize it to become a starting point for creating new files of this type.

Tip:
You can use editor variables (on page 323) in the template file content and they will be expanded when the files are opened. Also, see Customizing Document Templates (on page 377) for other template customization tips (for example, you could add placeholders or hints (on page 380) to assist authors).

2. Save the new document template and reference that location in Oxygen XML Editor. There are several options for doing this:

- **Saving the new template in a specific framework's directory** - Save the new template in a directory (for example, called templates) within that specific framework directory (usually a custom framework (on page 2133)). Then open the Document Type configuration dialog box (on page 140) for that specific framework, go to the Templates tab (on page 163), and click the + button in the bottom-right corner to add your new directory to the list. It is recommended that the reference be made relative to the framework directory (for example, ${frameworkDir}/templates). You can also remove any existing entries in the list that aren't applicable or won't be used in your custom framework. Click OK to close the configuration dialog box and then OK or Apply to save your changes.

- **Saving the new template in the Oxygen installation directory** - Save the new template in the templates directory of the Oxygen XML Editor installation directory ([OXYGEN_INSTALL_DIR]/templates). Document templates saved in this directory will appear in the Global templates category in the New document wizard (on page 368).

- **Saving the new template in a custom directory** - Save the new template in any directory of your choice and then add that directory to the list of templates in the Document Templates preferences page (on page 167). This user-defined directory will appear in the New document wizard (on page 368) along with all the new document templates that you save inside it. Click OK or Apply to save your changes.

Tip:
If you want to create a new template for a binary file (e.g. a zip archive), you need to add .bin to the end of the file name (for example, *.zip.bin or *.epub.bin). Otherwise, the files will be treated as XML/text documents and you will be prompted to choose the editor type.

Attention:
The name that you use to save the template will be the name that appears in the new document wizard, including capitalization, space, and characters (for example, My Custom Template1.xml will appear in the new file wizard as My Custom Template1). You can also configure the displayed name in a properties file by following the procedure found in the Configure the Displayed Names for Document Templates (on page 379) section.
3. Open the new document wizard (New toolbar button or File > New) and you should see your custom template in the appropriate folder.

**Note:**
For DITA templates, they will also appear in the dialog box for creating new DITA topics from the DITA Maps Manager, but if you customize the template (on page 377), you need to set the type property to dita in the corresponding properties file.

Related information
- Customizing Document Templates (on page 377)
- Sharing Custom Document Templates (on page 381)

### Customizing Document Templates

Oxygen XML Editor allows you to customize certain aspects of built-in or custom document templates. For example, you can customize the icons or specify a prefix/suffix that will be used for the proposed file name in the New document wizard (on page 368).

### Customizing the Icons for a Document Template

If you want to customize the icons to be used for document templates, use a properties file to specify the icons using the following procedure:

1. Create a new properties file or edit an existing one following these guidelines:
   a. If you want to create a new properties file, you can use the Properties template found in the New Document folder in the New document wizard (on page 368). If you want to edit an existing template, you can find them within the subfolders in the templates folder for each framework (for example, the DITA topic properties file is located in: OXYGEN_INSTALL_DIR/frameworks/dita/templates/topic/topic.properties).
   b. Use the same name as your custom template file except with a .properties extension (for example, MyTemplate.properties).
   c. In this properties file, specify the paths to the icons that will be used in the new file wizard. The properties file should look like this:

```properties
type=general
smallIcon=../icons/Article_16.png
bigIcon=../icons/Article_48.png
```

**Tip:**
For DITA files, the type property must be set to dita. Otherwise, the template will not appear in the dialog box for creating new DITA topics from the DITA Maps Manager (on page 2897). For all other types of files, set it to general. The icons specified in this
properties file will only be used for the new file wizards and not in any other part of the interface.

**Important:**
If you created a new template and chose to use a custom directory for the new template (in step 2 of the new template procedure (on page 376)), make sure that the path to the icons is relative to that directory.

2. Save the properties file in the same directory as your custom template.
3. Open the new file wizard (File > New) and you should see your custom icons next to the document template in the appropriate folder.

### Add a Prefix or Suffix to File Names for a Document Template

You can use a properties file for each document template to add a prefix or suffix to the file name that is proposed in certain dialog boxes when you create a new file from that template. This applies to the following new document dialog boxes:

- The new document dialog box that appears when you click the ✉️ New button on the toolbar (or File > New). The prefix or suffix is added to the name of the file in the **Save as** field.
- The new document dialog box that appears when you select **New > File** from the contextual menu in the Project view (on page 402). The prefix or suffix is added to the name of the file in the **File name** field.
- For DITA files, it also applies to the new document dialog box that appears when you select **Append Child > New**, **Insert Before > New**, or **Insert After > New** from the DITA Maps Manager (on page 2897). The prefix or suffix is added to the name of the file in the **Save as** field.
- For DITA files, it also applies to the **Fast Create Topics** dialog box (on page 2964) that you can use to create multiple skeleton topics at once.

To add a prefix or suffix to the file names for a document template, follow these steps:

1. Create a new properties file or edit an existing one.
   - If you create a new properties file, use the same name as the template file except with a `.properties` extension (for example, `MyTemplate.properties`). This properties file specifies the prefix/suffix that will be used to propose the file name in the new file wizards.

   When defining the prefix/suffix, the properties file should look something like this:

   ```xml
   type=general
   filenamePrefix=prod_
   filenameSuffix=_test
   ```
Important:
For DITA files, the type property must be set to dita. For all other types of files, set it to general.

- If you edit an existing template, simply define the prefix/suffix as specified above (on page 378).

2. Save the properties file in the same directory as the document template.
3. Open the new document wizard (using the methods described above (on page 378)) and when you select the appropriate template, you should see your prefix or suffix in the file name that is proposed in that dialog box.

Note:
The filenamePrefix and filenameSuffix properties can also have editor variables (on page 323) that do not require user interaction (i.e. editor variables that have $\{ask()\}$ and $\{answer()\}$ as values cannot be used).

Configure the Displayed Names for Document Templates

To change the name that is displayed for a document template, use the following procedure:

1. Create a new properties file or edit an existing one. If you create a new properties file, use the same name as the template file except with a .properties extension (for example, MyTemplate.properties).

2. Add a displayName property in the properties file:

   ```
   displayName=My Template Name
   ```

   Tip:
The names for framework (on page 3219)-specific document templates (such as DITA Topic or DocBook Article, as you would see in the Framework templates folder in the New file wizard) can be translated via the internationalization support. In this case, the properties file should contain something like:

   ```
   displayName=$\{i18n(tag)\}$
   ```

   where tag refers to an entry in the translation.xml file for that specific framework (for example, OXYGEN_INSTALL_DIR/frameworks/dita/i18n/translation.xml for DITA).

3. Save the properties file in the same directory as the document template.
4. Open the new file wizard (File > New) and you should see the new name for the template.
Adding Placeholders or Hints in a Document Template

If a document template contains empty elements, it may not be clear to the Author what should be inserted in them. You can define placeholders in document templates that provide hints for Authors to help them understand what type of content should be added in any particular empty element within the document. The placeholder text is specified using a processing instruction and the placeholders are removed when the Author inserts content in the corresponding element.

To define placeholders in a document template to provide authors with hints, follow this procedure:

1. Edit the document template.
2. Add placeholders in the form of processing instructions within the elements where you want hints to be displayed when an Author creates a document from the template. For example:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE topic PUBLIC "-//OASIS//DTD DITA Topic//EN" "topic.dtd">
<topic id="pi">
  <title><?oxy-placeholder content="Enter a title"?></title>
  <shortdesc><?oxy-placeholder content="Writing short descriptions induces the writer to clarify the main thesis of the topic. We recommended a 50 word limit."?></shortdesc>
  <body>
    <p><?oxy-placeholder content="A paragraph element should be a self-contained unit dealing with one idea or point."?></p>
  </body>
</topic>
```

Important:
The elements that contain the placeholder processing instructions cannot contain other content/text, not even whitespace used for indentation. Otherwise, the placeholder will not be rendered properly.

3. Save the template file.
4. Use the New document wizard (on page 368) to create a new document using your customized template and you should see the hints in the open document.

Resources

To see a visual demonstration of how to customize document templates and to get more ideas for other advanced customization possibilities, watch our Webinar: Working with DITA in Oxygen - Customizing the Editing Experience.
Sharing Custom Document Templates

Your custom document templates (on page 375) can be shared with the other members of your team so that they all have access to the templates in the New document wizard (on page 368). The best way to share them is by integrating them in an extended framework (on page 3219) (document type) configuration and then sharing the whole framework with the other users.

Sharing Custom Document Templates

To share custom document templates with other members of your team:

1. Create a custom framework by extending an existing one (on page 2133), if you have not already done so.
2. Create the new document template (on page 375), if you haven't already done so.
3. Save the new template in a directory (for example, called templates) within your custom framework directory. Then open the Document Type configuration dialog box (on page 140) for that specific framework, go to the Templates tab (on page 163), and click the + button in the bottom-right corner to add your new directory to the list. It is recommended that the reference be made relative to the framework directory (for example, ${frameworkDir}/templates). You can also remove any existing entries in the list that aren't applicable or won't be used in your custom framework.
4. Click OK to close the configuration dialog box and then OK or Apply to save your changes.
5. All that remains is to share the entire framework with anyone who needs to have access to the custom templates. There are several methods for sharing frameworks and you can find details here: Sharing a Framework (on page 2288).

Opening Documents

To open a document in Oxygen XML Editor, do one of the following:

• Go to File > Open (Ctrl + O (Command + O on macOS)) or click the Open toolbar button to display the Open File dialog box. The start folder of this dialog box can be either the last folder it visited or the folder of the currently selected file. This can be configured in the Global preferences page. (on page 126)
• Go to **File > Open URL** or click the **Open URL** toolbar button to display a dialog box where you can specify a URL (defined by a protocol, host, resource path, and an optional port) or use the browsing actions in the **Browse for remote file** drop-down menu.
• Click the **Open/Find Resource** toolbar button to search for a file to open.
• Go to **File > Reload** to load the last saved file content. All unsaved modifications are lost.
• Go to **File > Reopen** to reopen one of the recently opened document files. The list containing recently opened files can be emptied by invoking the **Clear history** action.
• Select the **Open** or **Open with** action from the contextual menu of the **Project view (on page 402)**.

### Related information

**Opening Local Files at Start-up (on page 382)**

### Opening the Current Document in a System Application

To open the currently edited document in the associated system application, use the **View in Browser/ System Application** action that is available in the **File** menu and on the **File** toolbar. If you want to open XML files in a specific internet browser, instead of the associated system application, you can specify the internet browser to be used. To do so, open the **Preferences** dialog box (**Options > Preferences**) (on page 124), go to **Global**, and set it in the **Default Internet browser** field. This will take precedence over the default system application settings.

### Opening Local Files at Start-up

There are two possibilities for opening local files at startup from a command line by adding their file paths as parameters:

• **scriptName [pathToXMLFile1] [pathToXMLFile2]**
  - **scriptName** is the name of the startup script for your platform (**oxygen.bat** on Windows, **oxygen.sh** on macOS and Linux).
  - **pathToXMLFileN** is the name of a local XML file.

• An XML file and a schema file to be associated automatically to the file and used for validation and content completion:

  ```
  scriptName -instance pathToXMLFile -schema pathToSchemaFile -schemaType XML_SCHEMA|DTD_SCHEMA|RNG_SCHEMA|RNC_SCHEMA -dtName documentTypeName
  ```
  - **scriptName** is the name of the startup script for your platform (**oxygen.bat** on Windows, or **oxygen.sh** on macOS and Linux).
  - **pathToXMLFile** is the name of a local XML file.
  - **pathToSchemaFile** is the name of the schema that you want to associate to the XML file, the four constants (XML_SCHEMA, DTD_SCHEMA, RNG_SCHEMA, RNC_SCHEMA) are the possible
schema types (XML Schema, DTD, Relax NG schema in full syntax, Relax NG schema in compact syntax).

- **documentTypeName** specifies the name of the document type that has the schema defined. If the document type is already set in preferences, its schema and type are updated.

**Tip:**
You can use the `-h` or `--help` parameters to see more detailed information about possible values.

---

**Related information**

Opening a Document at a Specific Location Using a Command-Line Interface *(on page 383)*

---

**Opening a Document at a Specific Location Using a Command-Line Interface**

Oxygen XML Editor offers support for opening a file at a specific position using a command-line interface to transmit parameters to the Oxygen XML Editor application launching script file (`oxygen.bat`/`oxygen.sh`). The following methods are available, depending on how you identify the position that is needed:

1. **Specific position values (line and column number, or character offset)**

   Oxygen XML Editor supports the following position parameters:
   - **line** - The line number.
   - **column** - The column number (has meaning if the `line` parameter is also defined).
   - **char** - The character offset.

   **Examples for Windows:**

   The following examples show how you can open an XML document in Oxygen XML Editor from a Windows command-line interface:

   ```
   oxygen.bat file:samples/personal.xml#line=4
   oxygen.bat file:samples/personal.xml#line=4;column=5
   oxygen.bat file:samples/personal.xml#line=4;column=5
   oxygen.bat file:samples/personal.xml#char=334
   ```

2. **Simplified XPath index path**

   Oxygen XML Editor will open an XML file and select one of its elements identified by a simplified XPath index path. For example, an index path of the form 1/5/7 identifies the seventh child of the fifth child of the root element.

   **Restriction:**
   Oxygen XML Editor will display a selection that starts with the first character of the content of the identified element and spans until the end of the line.
Examples for Windows:

The following example shows how you can open an XML document in Oxygen XML Editor and select the third child of the root element using a Windows command-line interface:

```
oxygen.bat file:samples/personal.xml#element(1/3)
```

3. **Anchors identified by ID attribute values**

Oxygen XML Editor will open an XML file and select the element whose @id attribute value is an exact match of the anchor *(on page 3216)* attached to a command-line instruction.

Examples for Windows:

The following example shows how you can open an XML document in Oxygen XML Editor and select the element that has the @id set to titleID using a Windows command-line interface:

```
oxygen.bat file:samples/personal.xml#titleID
```

Related information

Opening Local Files at Start-up *(on page 382)*

**Saving Documents**

You can save the document you are editing with one of the following actions:

- **File > Save.**
- ![Save](image) **Save** toolbar button - If the document was not yet saved, it displays the **Save As** dialog box.
- **File > Save As** - Displays the **Save As** dialog box, used either to name and save an open document to a file or to save an existing file with a new name.
- **File > Save To URL** - Displays a **Save to URL** dialog box that can be used to save a file identified by its URL (defined by a protocol, host, resource path, and an optional port). You can also use the browsing actions in the ![Browse](image) **Browse for remote file** drop-down menu.
- **File > Save All** - Saves all open documents. If any document does not have a file, displays the **Save As** dialog box.

**Auto Recover Documents**

Oxygen XML Editor includes an **Auto Recover** feature to help prevent losing unsaved content if you encounter an application or system crash. The feature is enabled by default and it automatically saves documents you are working on to a specified auto-recover file location. At every specified interval, all documents that have been modified since the last auto-save are saved to the specified location.

This feature is controlled by two options in the **Save** preferences page. You can disable it, or configure how often content is saved by selecting the desired value in the drop-down list of the **Save auto-recover**
information every option (on page 202), and you can specify the location of the saved documents in the Auto-recover file location option (on page 202).

In the event of an application or system crash, once you restart the application, Oxygen XML Editor looks for an auto-recover file for each document that is either automatically or manually reopened. If an auto-recover file is found, a dialog box is displayed with options for how to handle the recovered information.

![Auto Recover Dialog Box](image)

The dialog box offers the following choices:

- **Open recovered content in a new tab** - Opens the recovered document in a new tab.

  **Tip:**
  You can use the Compare Files tool (available in the Tools menu) to compare the recovered content with the last saved version of the document.

- **Replace current file content with recovered content** - Replaces the content of the last saved version of the document with the content of the recovered version of the document and removes the auto-recover file from disk.

- **Use current file content and discard recovered content** - Discards the recovered document and retains the last saved version of the document.

**Notes About the Auto-Recover Feature:**

- The Auto Recover feature works for both local and remote files.
- For DITA projects, the Auto Recover feature also works for DITA maps opened in the DITA Maps Manager.
- The Auto Recover feature does NOT work if there is not enough space available on the disk where the auto-recover file location is specified (on page 202).
- The Auto Recover feature does NOT work on files opened in the huge file editor (on page 471) (if you select the Optimize loading for huge files option when opening large documents (on page 469)).
Closing Documents

To close open documents, you can simply click the close icon (×) for the particular editor tab or use one of the following actions that are available by right-clicking the current editor tab (or from the File menu):

**Close (Ctrl + W (Command + W on macOS))**

Closes the currently selected editor.

**Close Other Files**

If multiple files are opened, this action is available to close all opened editors in the current group/stack of tabs except for the one you are currently viewing. If this action is selected from the File menu, it closes all opened editors in all groups/stacks of tabs except for the current one.

**Close Files to the Right**

Available only from the contextual menu of the current editor tab and it closes all opened editors to the right of the currently selected editor.

**Close All**

If multiple files are opened, this action is available to close all opened editors in the current group/stack of tabs. If this action is selected from the File menu, it closes all opened editors in all groups/stacks of tabs.

Working with Remote Documents

Oxygen XML Editor supports editing remote files, using the FTP, SFTP, WebDAV, SharePoint, and SharePoint Online for Office 365 protocols. You can edit remote files in the same way you edit local files. For example, you can add remote files to a project, or use them in XSL and FO transformations.

You can open one or more remote files in the Open URL dialog box (on page 387).

A WebDAV resource can be locked when it is opened in Oxygen XML Editor by selecting the Lock WebDAV files on open option (on page 304) to prevent other users to modify it concurrently on the server. If a user tries to edit a locked file, Oxygen XML Editor displays an error message that contains the lock owner's name. The lock is released automatically when the editor for that resource is closed in Oxygen XML Editor.

To avoid conflicts with other users when you edit a resource stored on a SharePoint server, you can Check Out the resource.

To improve the transfer speed, the content exchanged between Oxygen XML Editor and the HTTP / WebDAV server is compressed using the GZIP algorithm.

The current WebDAV Connection (on page 2068) details can be saved by switching to the Database perspective (on page 3221) and then you can browse and manage the connection in the Data Source Explorer view (on page 2015).
Open URL

To open this dialog box, go to File > Open URL (or click the Open URL toolbar button), then choose the Browse for remote file option from the drop-down action list.

Figure 62. Open URL Dialog Box

The displayed dialog box is composed of the following:

**Server Type**

Specifies the type of server. You can choose between:

- **WebDAV FTP** - For generic HTTP/FTP/WebDAV and other servers.
- **SharePoint Online** - For SharePoint Online servers.
- **SharePoint On-Premises** - For SharePoint (older version) servers.

**Server URL**

Specifies the protocol (HTTP, HTTPS or FTP) and the host name or IP of the server.

**Tip:**

When specifying a URL, follow these rules:
• To access an FTP server, write the protocol, host, and port (if using a non-standard one). For example, ftp://server.com or ftp://server.com:7800/.
• To access a WebDAV server, write the path to the directory of the WebDAV repository along with the protocol and the host name. For example, https://www.some-webdav-server.com:443/webdav-repository/.

Important:
Make sure that the repository directory ends in a slash "/". For example, https://www.some-webdav-server.com:443/webdav-repository/

Autoconnect
If selected, the browse action is performed every time when you open the dialog box.

User and Password
To browse for a file on a server, you have to specify the user and password for the server. This information is bound to the selected URL displayed in the File URL combo box, and used further in opening/saving the file.

Note:
Your password is well protected. If the options file is used on another machine by a user with a different username, the password will become unreadable since the encryption is dependent on the username. This is also true if you add URLs that contain a username and password to your project.

Save
If selected, the user and password are saved between editing sessions. The password is kept encrypted in the options file.

Browse
When you click this button, the directory listing will be shown in the main section of the dialog box. If the selected URL points to a SharePoint server, a dedicated SharePoint browsing component is presented.

Browser view
• If you are browsing a WebDAV or FTP repository, the items are presented in a tree-like fashion. You can browse the directories, and make multiple selections. Additionally, you may use the Rename, Delete, and New Folder actions to manage the file repository.

Note:
The file names are sorted in a case-insensitive way.
• When you browse a SharePoint repository, a specialized component renders the SharePoint site content.

The left side navigation area presents the SharePoint site structure in a tree-like fashion with various node types (such as sites, libraries, and folders).

Depending on the type of node, a contextual menu offers customized actions that can be performed on that node. The contextual menu of a folder allows you to create new folders and documents, import folders and files, and to rename and delete the folder.

**Note:**

The rename and delete actions are not available for library root folders (folders located at first level in a SharePoint library).

Each library node displays a drop-down menu next to its name where you can select what you want to display for the current library node. This functionality is also available on the contextual menu of the node.

![Figure 63. Drop-Down Menu to Select Which Items to Display](image)

The content of a folder is displayed in a tabular form, where each row represents the properties of a folder or document. The list of columns and the way the documents and folders are organized depends on the currently selected view of the parent library.

You can filter and sort the displayed items. To display the available filters of a column, click the filter widget located on the column header. You can apply multiple filters at the same time.

![Figure 64. Column Filter](image)

**File URL**

You can use this combo box to directly specify the URL to be opened or saved. You can type a URL such as `http://some.site/test.xml` (if the file is accessible through normal HTTP
protocol), or ftp://anonymous@some.site/home/test.xml (if the file is accessible through anonymous FTP).

This combo box also displays the current selection when the user changes selection by browsing the tree of folders and files on the server.

Changing File Permissions on a Remote FTP Server

Some FTP servers allow the modification of permissions of the files served over the FTP protocol. This protocol feature is accessible directly in the FTP/WebDAV file browser dialog box by right-clicking a tree node and selecting the Change permissions menu item.

In this dialog box, the usual Unix file permissions Read, Write, and Execute are granted or denied for the file owner, owner group, and the rest of the users. The aggregate number of permissions is updated in the Permissions text field when it is modified with one of the checkboxes.

WebDAV over HTTPS

If you want to access a WebDAV repository across a non-secure network, Oxygen XML Editor allows you to load and save the documents over the HTTPS protocol (if the server understands this protocol) so that any data exchange with the WebDAV server is encrypted.

When a WebDAV repository is first accessed over HTTPS, the server hosting the repository will present a security certificate as part of the HTTPS protocol, without any user intervention. Oxygen XML Editor will use this certificate to decrypt any data stream received from the server. For the authentication to succeed you should make sure the security certificate of the server hosting the repository can be read by Oxygen XML Editor. This means that Oxygen XML Editor can find the certificate in the key store of the Java Runtime Environment where it runs. You know the server certificate is not in the JRE key store if you get the error No trusted certificate found when trying to access the WebDAV repository.

Troubleshooting HTTPS

If Oxygen XML Editor cannot connect to an HTTPS-capable server and an error message mentions that it is "unable to find a valid certification path to the requested target", the HTTPS server is most likely configured to use a self-signed certificate or uses a certificate issued by an unknown authority that the Java Runtime Environment (JRE) that is used by Oxygen XML Editor does not trust.

Tip:
To make Oxygen XML Editor accept a certificate even if it is invalid, open the Preferences dialog box (Options > Preferences) (on page 124), go to Connection settings > HTTP(S)/WebDAV, and select the Automatically accept a security certificate, even if invalid option.

To trust a certificate, follow this procedure:
1. Export a certificate into a local file using any HTTPS-capable web browser:

**Chrome or Edge**

a. Navigate to the page that uses the certificate.
b. Right-click the page and select **Inspect**.
c. Select the **Security** tab.
d. Click **View Certificate**. **Step Result:** A **Certificate** dialog box is displayed.
e. Select the **Details** tab of the **Certificate** dialog box.
f. Click the **Copy to File** button. **Step Result:** The **Certificate Export Wizard** is started.
g. Follow the instructions in the wizard for the DER encoded binary X.509 certificate. Save the certificate to the local file **server.cer**.

**Safari**

a. Navigate to the page that uses the certificate.
b. If there is a "This connection is not private" message, click **Show Details** and in the expanded panel, click **view the certificate**.
c. Otherwise, in the address bar, click the **padlock icon** on the left side of the website name and in the displayed pop-up, click **Show Certificate**.
d. Another pop-up box is displayed showing information about the **certificate**. Drag the large **certificate** icon to a **Finder** window. A **.cer** file will be created in the indicated folder from **Finder**.

2. Import the local file into the JRE running Oxygen XML Editor:

a. Open a text-mode console with administrative rights. If Oxygen XML Editor has been installed in a user’s home directory and includes a bundled JRE, administrative rights are not required. In all other cases, administrative rights will be required.
b. Go to the **lib/security** directory of the JRE running Oxygen XML Editor. You can find the home directory of the JRE in the **java.home** property that is displayed in the **About** dialog box (**System properties** tab).

c. Run the following command:

```
..\..\bin\keytool -import -trustcacerts -file server.cer -keystore cacerts
```

The **server.cer** file contains the server certificate, created during the previous step. The **keytool** requires a password before adding the certificate to the JRE **keystore** (**on page 3220**). The default password is **changeit**. If someone changed the default password, then that person is the only one who can perform the import.
Tip:
If you need to import multiple certificates, you need to specify a different alias for each additional imported certificate with the `-alias` command-line argument, as in the following example:

```bash
..\..\bin\keytool -import -alias myalias1 -trustcacerts -file server1.cer -keystore cacerts
```

```bash
..\..\bin\keytool -import -alias myalias2 -trustcacerts -file server2.cer -keystore cacerts
```

3. Restart Oxygen XML Editor.

Related information
HTTP(S)/WebDAV Preferences (on page 303)

HTTP Authentication Schemes

Oxygen XML Editor supports the following HTTP authentication schemes:

- **Basic** - The *basic* authentication scheme defined in the RFC2617 specifications.
- **Digest** - The *digest* authentication scheme defined in the RFC2617 specifications.
- **NTLM** - The *NTLM* scheme is a proprietary Microsoft Windows Authentication protocol (considered to be the most secure among currently supported authentication schemes).

Note:
For NTLM authentication, the user name must be preceded by the name of the domain it belongs to, as in the following example:

```
domain\username
```

- **Kerberos (on page 392)** - An authentication protocol that works on the basis of *tickets* to allow nodes communicating over a non-secure network to prove their identity to one another in a secure manner.

Single Sign-on

Oxygen XML Editor implements the *Single sign-on* property (meaning that you can log on once and gain access to multiple services without being prompted to log on for each of them), based on the *Kerberos* protocol and relies on a *ticket-granting ticket (TGT)* that Oxygen XML Editor obtains from the operating system.
To turn on the Kerberos-based authentication, you need to add the following system property in the .vmoptions configuration file or start-up script:

-Djavax.security.auth.useSubjectCredsOnly=false

Switching, Moving, or Hiding Editor Tabs

Each file that has been opened has a tab at the top of the editing pane and there are several ways to switch between tabs or move them, and you can even hide the tabs to only show the currently open file.

Note:

If multiple file tabs are left open when you close the application, upon startup, Oxygen XML Editor will not load the file content until you switch to the corresponding file tab. The tabs remain visible as placeholders until the focus is switched to them. This helps to improve the application's startup time. If you want to disable this feature (meaning that the previously open files will all be re-loaded at startup), deselect the Load file content only when switching to its corresponding editor tab option in the Global preferences page (on page 127).

Switching Editor Tabs

You can switch between editor tabs by using any of the following methods:

Mouse and Scroll Wheel

Of course, you can switch to a different editor tab by left-clicking the tab with your mouse, but when there are too many open tabs to fit on the screen, you can hover over the tab stripe and use the scroll wheel on your mouse to scroll to the left or right (same as using the two arrows on the far-right of the tab stripe).

Buttons on the Far-Right of the Tab Stripe (➡️)

You can use the arrow buttons (⬅️➡️) on the right side of the tab stripe to scroll to the left or right and the Show List button opens a pop-up window that displays all the open file tabs and allows you to select and switch to a specific open file.

Ctrl + Tab (Command + Tab on macOS) [NOTE: Ctrl + Page Down (Ctrl + Alt + Right Arrow on macOS) does the same]

Switches to the next open tab in the order specified in the Order of switching between editor tabs option (on page 128).
Ctrl + Shift + Tab (Command + Shift + Tab on macOS) [NOTE: Ctrl + Page Up (Ctrl + Alt + Left Arrow on macOS) does the same]

Switches to the previous open tab in the order specified in the Order of switching between editor tabs option (on page 128).

Window > Switch editor tab (Ctrl + F9 (Command + F9 on macOS))

This action opens a dialog box that allows you to switch to a particular editor tab by selecting it from a filterable list. This is especially helpful when you have a large amount of open file tabs and you want to switch to a certain tab this is not shown on the screen. It includes a search filter field and several options to help you find specific open file tabs.

![Switch Editor Tab Dialog Box](image)

The **Switch Editor Tab** dialog box contains the following options and features:

**Search Filter**

You can enter text in the filter field at the top of the dialog box to filter the list and search for specific open files. You can enter any number of terms, separated by space, and wildcards are allowed (for example, * to match any sequence of characters, or ? to match a single character). This field also has a history drop-down that allows you to select previously used search terms.

**Match all terms**
If this option is selected, only the files that match all of your search terms will be displayed. If you use a wildcard in the search filter, this option is automatically disabled.

**Include file paths**

If this option is selected, the search is expanded to include file paths, and also the paths are displayed in this dialog box.

**Case sensitive**

If this option is selected, the search operation will be case-sensitive.

**List of Open File Tabs**

All files that are currently open are displayed in the upper part of the main pane of the dialog box, followed by recently closed files. Files that have been modified but not yet saved are prefixed by an asterisk. To switch to a particular file tab, double-click the file or select it and click **OK**.

**Moving Editor Tabs**

You can move editor tabs by using any of the following methods:

**Mouse Drag**

You can use your mouse to drag editor tabs to a new location on the tab stripe.

**Ctrl + Alt + Comma**

Moves the current file tab one position to the left.

**Ctrl + Alt + Period**

Moves the current file tab one position to the right.

**Hiding Editor Tabs**

If you want to hide all the file tabs and only show the currently open file, select **Hide editor tabs** from the **Window** menu. This does not close the other tabs, just hides them. You can still navigate between tabs using keyboard shortcuts (Ctrl + Tab, Ctrl + Shift + Tab, Ctrl + F6, Ctrl + Shift + F6) or by selecting **Next editor** or **Previous editor** from the **Window** menu.

**Contextual Menu of the Current Editor Tab**

A contextual menu is available when you right-click the current editor tab label.

The actions that are available depend on the context and the number of files that are opened. The menu includes the following actions:

**Close (Ctrl + W (Command + W on macOS))**
Closes the currently selected editor.

**Close Other Files**

If multiple files are opened, this action is available to close all open editors in the current group/stack of tabs except for the one you are currently viewing.

**Close Files to the Right**

Closes all open editors to the right of the currently selected editor.

**Close All**

If multiple files are opened, this action is available to close all open editors.

**Move editor tab to the left (Ctrl + Alt + Comma)**

Moves the current editor tab one position to the left.

**Move editor tab to the right (Ctrl + Alt + Period)**

Moves the current editor tab one position to the right.

**Reopen last closed editor Ctrl + Alt + T (Command + Alt + T on macOS)**

Reopens the last closed editor.

**Maximize Editing Area**

Collapses all the side views and spans the editing area to cover the entire width of the main window.

**Add to project**

Adds the file you are editing to the current project.

**Add all to project**

If multiple files are opened, this action is available to add all the open files in the current group/stack of tabs to the current project.

**Copy Location**

Copies the disk location of the file.

**Show in Explorer (Show in Finder on macOS)**

Opens the Explorer to the file path of the file.

**Viewing File Properties**

The **Properties** view displays information about the currently edited document. The information includes:

- Character encoding.
- Full path on the file system.
- Schema used for content completion and document validation.
- Document type name and path.
- Associated transformation scenario.
- Read-only state of a file.
• Bidirectional text (left to right and right to left) state.
• Total number of characters in the document.
• Line width.
• Indent with tabs state.
• Indent size.

The view can be accessed from **Window > Show View > Properties**.

To copy a value from the **Properties** view in the clipboard (for example, the full file path), use the **Copy** action available on the contextual menu of the view.

**Simple Text Editor**

While Oxygen XML Editor specializes in XML-related technologies, you can also use it to create and edit various types of non-XML files. Non-XML files are opened in a simple text editor and many of the helpful features that are commonly used when editing XML files in the Oxygen XML Editor **Text editing mode** *(on page 517)* are available in this simple editor.

**Types of Non-XML Files That are Supported in the Simple Text Editor**

The types of non-XML files that can be created and edited in the simple text editor include:

- Java
- C++
- C
- PHP
- Perl
- Properties
- SQL
- Shell executables
- Batch
- Python
- Text

**Features Available in the Simple Text Editor**

When editing files in the simple text editor, the features that are available include the following:

- **Project Support** - The unique features that are designed to help you work with projects *(on page 398)* are available for all types of files.
- **Shortcut Actions** - Many of the shortcut actions that are available in **Text mode** *(on page 521)* are also available in the simple text editor.
- **Drag and Drop** - The normal drag and drop support is available in the simple text editor.
- **Content Selection Features** - The content selection shortcuts *(on page 529)* that are available in **Text mode** (including the *Rectangular Selection* feature) are also available in the simple text editor.
• **Bookmarks** - You can use *bookmarks* to mark positions *(on page 519)* in any type of file so that you can return to it later.

• **Convert Hexadecimal Characters** - You can convert a sequence of hexadecimal characters to the corresponding Unicode character *(on page 569)*.

• **Encoding/Decoding Actions** - Contextual menu actions are available to encode or decode Base 64, Base 32, and Hex schemes *(on page 569)*.

• **Code Templates** - You can define your own *code templates* *(on page 536)* for any type of file and use the *Content Completion Assistant* *(on page 3217)* to invoke them.

• **Syntax Highlighting** - Non-XML files also support syntax highlighting with dedicated coloring schemes. To customize them, open the *Preferences* dialog box *(Options > Preferences)* *(on page 124)* and go to *Editor > Syntax Highlight* *(on page 224)*. Select and expand the appropriate section in the top pane for the type of file you are editing and you can see the effects of your changes in the *Preview* pane.

• **Find/Replace** - You can use the *Find/Replace action* *(on page 432)* to find or replace all the occurrences of a word or string of characters in any type of file that you are editing.

• **File Comparison Tool** - The *Compare Files tool* *(on page 474)* can also be used to compare non-XML files.

### Using Projects to Group Documents

Oxygen XML Editor includes a *Project view* *(on page 402)* that helps you organize your projects. Oxygen XML Editor offers a variety of helpful features for working with projects and makes it easy to share your projects with other members of your team. This section presents various unique features that will help you to create and work with projects.

**Tip:**

There are several sample project templates available for DITA users that can be used as a starting point or for inspiration. These sample project templates are found in the *Framework templates > DITA* folder in the *New Project wizard: (on page 399)*

• **Sample DITA Project** - This is a basic DITA project meant to help new users see how a DITA project is structured.

• **Startup DITA Project** - This is a startup DITA project that imposes a custom set of options (e.g. spell check settings and custom dictionaries), a custom DITA framework extension (e.g. custom new file templates, custom actions, custom CSS used for visual editing) and a folder structure for a DITA project according to best practices. Once created, the project contains a *Readme.html* file that explains all customizations and their benefits. If you plan to start your own DITA project using a version control system (such as Git), you can use this startup DITA project template to customize various aspects of DITA editing and share them with your team.
Creating a New Project

Oxygen XML Editor allows you to organize your XML-related files into projects. This helps you manage and organize your files and also allows you to perform batch operations (such as validation and transformation) over multiple files. You can also share your project settings and transformation/validation scenarios (on page 415) with other users. Use the Project view (on page 402) to manage projects, and the files and folders contained within.

Creating a New Project

To create a new project, select New Project from the Project menu, the New menu in the contextual menu, or the drop-down menu at the top-left of the Project view.

This opens a new project wizard:

Figure 66. New Project Wizard

Tip:
There are several sample project templates available for DITA users that can be used as a starting point or for inspiration. These sample project templates are found in the Framework templates > DITA folder in the New Project wizard: (on page 399)

- **Sample DITA Project** - This is a basic DITA project meant to help new users see how a DITA project is structured.
- **Startup DITA Project** - This is a startup DITA project that imposes a custom set of options (e.g. spell check settings and custom dictionaries), a custom DITA framework extension (e.g. custom new file templates, custom actions, custom CSS used for visual editing) and a folder structure for a DITA project according to best practices. Once created, the project contains a
Readme.html file that explains all customizations and their benefits. If you plan to start your own DITA project using a version control system (such as Git), you can use this startup DITA project template to customize various aspects of DITA editing and share them with your team.

With the exception of the Default project template, which is a pseudo-template and does not exist on the local disk (it is used only to create a new .xpr file), project templates are actually ZIP archives (with a .zxpr extension) and are stored within the file template directory structure (for example, frameworks\dita\templates\sample-project\Sample DITA Project.zxpr).

Tip:
Archives with a .zxpr extension can be edited in the Archive Browser view (on page 2008).

After selecting a project template, you can specify the following:

**Project file name**

Specifies the name of the new project file. Oxygen XML Editor provides a default proposal for the file name based on the following rules:

- If there is an .xpr file inside the archive, its name is used.
- Otherwise, the name of the template is used.

**Project directory**

Specifies the directory where the archive content will be extracted.

Note:
The archive should not contain an extra single folder as the root. For the Project directory path to work properly, the archive must have the .xpr file on the first level, along with the other resources (files and folders).

Once you are done, click the Create button to begin the creation process. Oxygen XML Editor extracts the content from the archive inside the path specified in the Project directory field.

**Editor Variables in Project Templates**

By default, the editor variables inside project resources created from a project template are not resolved. To start having them resolved, the project template must be customized (on page 377) by using the expandEditorVariablesIncludeFilter property. This filter determines the resources where the editor variables will be resolved. If you need to exclude a subset of resources from the set specified by the expandEditorVariablesIncludeFilter property, the expandEditorVariablesExcludeFilter property can be used.
Note:
Usually, project files (*.xpr), framework files (*.framework), and framework extension scripts (*.exf) should be excluded from the editor variable resolving process.

The values of the inclusion and exclusion filters can be file paths relative to the project directory that can use wildcards or simply wildcards. Each filter can have multiple values, separated by spaces.

Possible filter values:

- ./* - Matches all resources from the first level in the project directory.
- * or ./** - Matches all resources on all levels inside the project directory.
- dir1/dir2/*.dita - Matches all .dita files from [PROJECT_DIR]/dir1/dir2, but not from subdirectories of dir2.
- dir1/dir2/**/*.dita - Matches all .dita files from [PROJECT_DIR]/dir1/dir2, including those from subdirectories of dir2.
- dir1/**/* - Matches all resources on all levels inside [PROJECT_DIR]/dir1.
- dir1/article1.xml, dir2/article2.xml - Matches only the two .xml files.
- ./**/_suffix.md, ./**/prefix_*.html - Matches all .md files with names that end with _suffix and all .html files with names that start with prefix_.

Adding Items to the Project

To add items to the project, select any of the following actions that are available when invoking the contextual menu in the Project view:

New > File
Opens a New file dialog box that helps you create a new file and adds it to the project structure.

New > Folder
Opens a New Folder dialog box that allows you to specify a name for a new folder and adds it to the structure of the project.

The project itself is considered a logical folder. You can add a logical folder, or content to a logical folder, by using one of the following actions that are available in the contextual menu, when invoked from the project root:

New > Logical Folder
Creates a logical folder in the tree structure (the icon is a magenta folder on macOS - �OLDER).

New > Logical Folders from Web
Replicates the structure of a remote folder accessible over FTP/SFTP/WebDAV, as a structure of logical folders. The newly created logical folders contain the file structure of the folder it points to.

Add Folder
Adds a link to a physical folder, whose name and content mirror a real folder that exists in the local file system (the icon of this action is different on macOS - 🗄).

Add Files

Adds links to files on the local file system.

Add Edited File

Adds a link to the currently edited file in the project.

Using Linked Folders (Shortcuts)

Another easy way to organize your XML working files is to place them in a directory and then to create a corresponding linked folder in your project. If you add new files to that folder, you can simply use the Refresh (F5) action from the project contextual menu and the Project view (on page 402) will display the existing files and subdirectories. If your files are scattered among several folders, but represent the same class of files, you might find it useful to combine them in a logical folder.

You can create linked folders (shortcuts) by dragging and dropping folders from the Windows Explorer (macOS Finder) to the project tree, or by selecting Add Folder in the contextual menu from the project root. Linked folders are displayed in the Project view (on page 402) with bold text. To create a file inside a linked folder, select the New > File action from the contextual menu. The linked files presented in the Project view (on page 402) are marked with a special icon.

Note:

Files may have multiple instances within the folder system, but cannot appear twice within the same folder.

For more information on managing projects and their content, see Project View (on page 402).

For more details about how you can share projects with other users, see Sharing a Project - Team Collaboration (on page 415).

Related information
Using Projects to Group Documents (on page 398)

Project View

The Project view is designed to assist you with organizing and managing related files grouped in the same XML project. The actions available in the contextual menu and on the toolbar associated to this panel allows you to create XML projects and provide shortcuts to various operations for the project documents.
By default, the view is positioned on the left side of the Oxygen XML Editor window, above the Outline view (on page 539). If the view has been closed, it can be reopened at any time from the Window > Show View menu (or using the Show Project View action from the Project menu).

**Project View Toolbar**

The tree structure occupies most of the view area. In the upper left side of the view, there is a drop-down menu that contains all recently used projects and some actions to open a project or create a new one. You can use this history drop-down menu to quickly switch to a recently opened project. If you enable the Remember layout changes for each project option in the Application Layout preferences page (on page 136), the application will remember the layout, open files, and editing location for your session when you switch projects.

The following actions are grouped in the upper right corner:

- **Collapse All**
  
  Collapses all project tree folders. You can also collapse/expand a project tree folder if you select it and press the Enter key or Left Arrow to collapse and Right Arrow to expand.

- **Link with Editor**
  
  When selected, the project tree highlights the currently edited file, if it is found in the project files.

  **Note:**
  
  This button is disabled automatically when you move to the Debugger perspective (on page 3221).

- **Settings**
  
  A submenu that contains the following actions:
Filters

Allows you to filter the information displayed in the Project view. Click the toolbar button to set filter patterns for the files you want to show or hide. Also, you can set filter patterns for the linked directories that are hidden.

Show Full Path

When selected, linked files and folders are presented with a full file path.

Enable Main Files Support

Select this option to enable the Main Files support (on page 418).

Change Search and Refactor operations scope

Allows you to change the collection of documents that define the context of the search and refactor operations (on page 831).

- Use only Main Files, if enabled - Restricts Oxygen XML Editor to perform the search and refactor operations starting from the main files that are defined for the current resource. This option is available when you select Project in the Select the scope for Search and Refactor operations dialog box and the Main Files support is enabled.
- Working sets - Allows you to specify the set of files that will be used for the scope of the search and refactor operations.

File Explorer Area

The rest of the view is basically a file explorer similar to most other commonly used file explorers. The XML project (.xpr file) is a logical container with a collection of resources (folders and files). The types of resources displayed include:

- Logical folders with Linked folders/files - Marked with a blue icon on Windows and Unix/Linux ( ), and a magenta icon on macOS ( ), they help you group files within the project. This folder type is used as containers for linked resources (shortcuts). The icons for file shortcuts include a shortcut symbol ( ) and names of folder shortcuts are displayed in bold text. The logical folders are created on the project root or inside other logical folders by using the contextual menu action New > Logical Folder, and the linked folders/files are added using Add Files, Add Folder, or by dragging and dropping files/folders from the view or the system file explorer. Remove from Project can be used to remove them from the project and the Remove from Disk (Shift+Delete) action can be used to remove them from both the project and the local file system.
- Physical folders and files - Marked with the operating system-specific icon for folders (usually a yellow icon on Windows and a blue icon on macOS). These folders and files are mirrors of real folders or files that exist in the local file system. They are created or added to the project by using contextual menu actions or contextual menu actions and then using the contextual menu actions.
actions (such as New > File, New > Folder, Copy, and Paste) or by dragging and dropping files/folders from the view or the system file explorer. Also, the contextual menu action Remove from Disk (Shift+Delete) can be used to remove them from the project and local file system.

**Figure 68. Project View with Both Types of Resources**

Creating New Projects

The following action is available from the New menu when right-clicking any item, the Project menu, or from the drop-down menu in the top-left of the Project view:

- **New Project**
  
  Opens a wizard that assists you with creating a new project. For more details, see Creating a New Project (on page 399).

Managing Project Contents

There are various contextual menu actions, shortcuts, and ways to organize the folders and files inside the project:

Creating New Folders and Files

- **Right-click any item > New > File**
  
  Opens a New file dialog box (on page 368) that helps you create a new file and adds it to the project structure.

- **Right-click any item in a physical folder > New > Folder**
Opens a **New Folder** dialog box that allows you to specify a name for a new folder and adds it to the structure of the project.

**Right-click any item in a logical folder > New > Logical Folder**

Creates a logical folder in the tree structure (the icon is a magenta folder on macOS).

**Right-click on a logical folder > New > Logical Folders from Web**

Replicates the structure of a remote folder accessible over FTP/SFTP/WebDAV, as a structure of logical folders. The newly created logical folders contain the file structure of the folder it points to.

**Adding Resources**

You can add resources by using drag and drop (or **Copy** and **Paste**) actions from within the **Project** view or dragging them from the system file explorer. Files may have multiple instances within the folder system, but cannot appear twice within the same folder.

**Adding Resources to Logical Folders**

You can add resources to logical folders by using the following actions available in the contextual menu when invoked on a logical folder (or the project's root container):

- **Add Folder**
  Adds a link to a physical folder, whose name and content mirror a real folder that exists in the local file system (the icon for this action is different on macOS).

- **Add Files**
  Adds links to files on the local file system.

- **Add Edited File**
  Adds a link to the currently edited file in the project.

**Removing Folders and Files**

To remove logical folders or the linked resources inside them from the project, use **Remove from Project** from the contextual menu (or press **Delete** on your keyboard).

To remove folders or files from both the project and the local file system, use **Remove from Disk** from the contextual menu (or press **Shift+Delete** on your keyboard).

**Moving Folders and Files**

You can move the resources by using drag and drop actions from within the **Project** view (the **Enable drag-and-drop in Project view** option must be selected in the **View preferences page (on page 306)**).
You can also use the usual Cut, Copy, and Paste actions to move resources in the project.

You can also move certain types of files (such as XML, XML Schema, Relax NG, WSDL, and XSLT) by using the Refactoring > Move resource action from the contextual menu. This action opens the Move resource dialog box that includes the following options:

- **Destination** - Presents the path to the current location of the resource you want to move and gives you the option to introduce a new location.
- **New name** - Presents the current name of the moved resource and gives you the option to change it.
- **Update references of the moved resource(s)** - Select this option to update the references to the resource you are moving, based upon the selected scope. You can select or configure the scope (on page 831) by using the button.

Renaming Folders and Files

There are several ways to rename a folder or file in the project (this works for both physical and linked resources):

- Select Rename from the contextual menu.
- Press F2 on your keyboard.
- Select the item, then click the name, and type the new name.

You also can rename certain types of files (such as XML, XML Schema, Relax NG, WSDL, and XSLT) by using the Refactoring > Rename resource action from the contextual menu. This action opens the Rename resource dialog box that includes the following options:

- **New name** - Presents the current name of the edited resource and allows you to modify it.
- **Update references of the renamed resource** - Select this option to update the references to the resource you are renaming. You can select or configure the scope (on page 831) by using the button.

Opening Files

There are several ways to open a file:

- Double-click the file.
- Select it and press Enter on your keyboard.
- Right-click the file and select Open.
- If there are no other files open in the editor area, you can drag the file from the project tree and drop it in the editor area.
- If you want to choose the application or location where to open it, you can right-click the file and select Open with.
Saving the Project

The project file is automatically saved every time the content of the Project view is saved or modified by actions such as adding or removing files and drag and drop.

Other Contextual Menu Actions

Numerous other actions are available in the contextual menu, depending on the type of file or folder where it is invoked from (some actions are available for multiple selected files):

Show in submenu

Explorer (Finder on macOS)

On Windows and macOS, the parent directory of the selected file or folder is presented in a specific Explorer/Finder window, and the selected resource is highlighted. On Linux, the selected file or folder is not highlighted after opening its parent in the file explorer.

Terminal

Opens a console (terminal) at the location of the selected physical resource. If the resource is a file, it will start at the parent directory.

Copy Location

Copies an application-specific URL for the selected resource to the clipboard.

Refactoring submenu

Oxygen XML Editor includes some refactoring operations that help you manage the structure of your documents. The following actions are available from the contextual menu in the Refactoring submenu:

Rename resource (Available for certain types of XML documents)

Opens the Rename resource dialog box (on page 412) where you can change the name of a resource. It also includes an option to update the references to the renamed resource and you can choose between various scopes for the operation.

Move resource (Available for certain types of XML documents)

Opens the Move resource dialog box (on page 412) where you can choose a destination and change the name of a resource. It also includes an option to update the references to the moved resource and you can choose between various scopes for the operation.

XML Refactoring

Opens the XML Refactoring tool wizard (on page 840) that presents refactoring operations to assist you with managing the structure of your XML documents.

Other XML Refactoring Actions
For your convenience, the last 5 XML Refactoring tool operations (on page 840) that were finished or previewed will also appear in this submenu.

**Resource Hierarchy**

Opens the Resource Hierarchy/Dependencies view (on page 832) that allows you to see the resource hierarchy for an XML document.

**Resource Dependencies**

Opens the Resource Hierarchy/Dependencies view (on page 832) that allows you to see the resource dependencies for an XML document.

**Refresh**

Refreshes the content and the dependencies between the resources in the Main Files directory (on page 418).

**Find/Replace in Files**

Opens the Find/Replace in Files dialog box (on page 436) that allows you to find and replace text in multiple files.

**XPath in Files**

Opens the XPath/XQuery Builder view (on page 2002) that allows you to compose XPath and XQuery expressions and execute them over the currently edited XML document.

**Open/Find Resource**

Opens the Open/Find Resource dialog box (on page 426).

**Check Spelling in Files**

Allows you to check the spelling of multiple files. (on page 458)

**Format and Indent Files**

Opens the Format and Indent Files dialog box (on page 561) that allows you to configure the format and indent (pretty-print (on page 322)) action that will be applied on the selected documents.

**Open in SVN Client**

Syncro SVN Client (on page 2718) tool is opened and it highlights the selected resource in its corresponding working copy.

**Compare**

Allows you to compare multiple files or directories and the order of your selection determines where they are opened in the Compare Files (on page 474) or Compare Directories (on page 494) tool. If you select two files or folders, your first selection will be opened in the left panel and the other one in the right panel.
You can also select 3 files and the tool will automatically be opened in the three-way comparison mode (on page 477). If you select three files, your first selection will be opened in the left panel, the second in the right panel, and the third selection will be the base (ancestor) file.

**HTML to XML Well-formed (Available when selecting multiple resources)**

Batch converts the selected HTML documents to be XML well-formed. This means that missing end tags will be added to applicable elements, unclosed tags will be properly closed, and quotes will be added to attribute values that were missing the quotes.

---

**Notes:**

- All selected HTML files are backed up before being processed (same path/name but with the ".bak" extension added at the end).
- Any detected conversion errors are grouped and listed in a dedicated tab in the Results pane at the bottom of the application.
- A brief report is displayed at the end of the operation.

---

**Transform submenu**

The currently selected files in the Project view can be transformed in one step with one of the following actions available from contextual menu in the Transform submenu:

- **Apply Transformation Scenario(s)**
  
  Obtains the output with one of the built-in scenarios (on page 1433).

- **Configure Transformation Scenario(s)**
  
  Opens a dialog box (on page 1548) that allows you to configure pre-defined transformation scenarios.

- **Transform with**
  
  Allows you to select a transformation scenario to be applied to the currently selected files.

---

**Validate submenu**

The currently selected files in the Project view can be checked to be XML well-formed or validated against a schema (DTD, XML Schema, Relax NG, Schematron or NVDL) with one of the following contextual menu actions found in the Validate submenu:

- **Check Well-Formedness**
  
  Checks if the selected file or files are well-formed.

- **Validate**
  
  Performs the validation process.
Validates the selected file or files against their associated schema. For EPUB files, this action triggers an [EPUB Validate and Check for Completeness](on page 2011) operation.

**Validate with Schema**
Validates the selected file of files against a specified schema.

🔗 **Configure Validation Scenario(s)**
Allows you to configure and run a validation scenario (on page 787).

**Generate Documentation submenu**

- **Generate Documentation > XML Schema Documentation**
  Opens the XML Schema Documentation Dialog Box (on page 1011).

- **Generate Documentation > XSLT Stylesheet Documentation**
  Opens the XSLT Stylesheet Documentation Dialog Box (on page 925).

- **Generate Documentation > XQuery Documentation**
  Opens the XQuery Documentation Dialog Box (on page 1045).

- **Generate Documentation > WSDL Documentation**
  Opens the WSDL Documentation Dialog Box (on page 1067).

🔗 **Properties**
Displays the properties of the current file in a Properties dialog box.

**Enable Main Files Support (Available from the project container)**
Allows you to enable the [Main Files Support](on page 419) for each project you are working on.

🔗 **Detect Main Files (Available from the project container when Main Files Support is enabled)**
Opens the Detect Main Files wizard (on page 420) that enables the automatic detection of main files.

🔗 **Add to Main Files (Available when Main Files Support is enabled)**
Adds the selected files to the Main Files folder (on page 421).

**Project Menu Actions**
The following actions are available in the Project menu:

🔗 **New Project**
Opens a wizard that assists you with creating a new project. For more details, see [Creating a New Project](on page 399).

🔗 **Open Project** *(Ctrl + F2 (Command + F2 on macOS))**
Opens an existing project. Alternatively, you can open a project by dropping an Oxygen XML Editor XPR project file from the file explorer into the **Project** panel.

**Notice:**
When a project is opened for the first time, a confirmation dialog box will be displayed that asks you to confirm that the project came from a trusted source. This is meant to help prevent potential security issues.

**Save Project As**
Allows you to save the current project under a different name.

**Validate all project files**
Checks if the project files are well-formed and their mark-up conforms with the specified DTD, XML Schema, or Relax NG schema rules. It returns an error list in the message panel.

**Filters**
Opens the **Project filters** dialog box that allows you to decide which files and directories will be shown or hidden.

**Enable Main Files Support**
Allows you to enable the **Main Files Support** *(on page 419)* for each project you are working on.

**Change Search and Refactor operations scope**
Opens a dialog box that allows you to define the context of search and refactor operations.

**Show Project View**
Displays the **Project** view.

**Reopen Project**
Contains a list of links of previously used projects. This list can be emptied by invoking the **Clear history** action.

**Moving/Renaming Resources in the Project View**

The **Refactoring** submenu in the contextual menu of the **Project view** *(on page 402)* provides actions for moving or renaming certain types of XML resources in the current project while offering the option to update the references to the resources.

**Moving Resources**

You can move certain types of files (such as XML, XML Schema, Relax NG, WSDL, and XSLT) by using the **Refactoring > Move resource** action from the contextual menu. This action opens the **Move resource** dialog box that includes the following options:
• **Destination** - Presents the path to the current location of the resource you want to move and gives you the option to introduce a new location.

• **New name** - Presents the current name of the moved resource and gives you the option to change it.

• **Update references of the moved resource(s)** - Select this option to update the references to the resource you are moving, based upon the selected scope. You can select or configure the scope (on page 831) by using the button.

### Renaming Resources

You can rename certain types of files (such as XML, XML Schema, Relax NG, WSDL, and XSLT) by using the **Refactoring > Rename resource** action from the contextual menu. This action opens the **Rename resource** dialog box that includes the following options:

• **New name** - Presents the current name of the edited resource and allows you to modify it.

• **Update references of the renamed resource** - Select this option to update the references to the resource you are renaming. You can select or configure the scope (on page 831) by using the button.

### Problems Updating References of Moved/Renamed Resources

In some cases, the references of a moved or a renamed resource cannot be updated. For example, when a resource is resolved through an **XML Catalog (on page 3225)** or when the path to the moved or renamed resource contains entities. For these cases, Oxygen XML Editor displays a warning dialog box.
Batch Validation and Transformation

Oxygen XML Editor provides support for batch validation and batch transformation. Actions are available in the Project view that provide the ability to validate or transform one or more files attached to a project.

Batch Validation

To batch validate files, select the files (or directories), right-click, and choose one of the following actions from the Validate submenu:

- **Check Well-Formedness**
  Checks if the selected file or files are well-formed.

- **Validate**
  Validates the selected file or files against their associated schema. For EPUB files, this action triggers an EPUB Validate and Check for Completeness (on page 2011) operation.

Validate with Schema

 Validates the selected file of files against a specified schema.

Configure Validation Scenario(s)
Batch Transformation

To batch transform files, select the files (or directories), right-click, and choose one of the following actions from the Transform submenu:

- **Apply Transformation Scenario(s)**
  Obtains the output with one of the built-in scenarios (on page 1433).

- **Configure Transformation Scenario(s)**
  Opens a dialog box (on page 1548) that allows you to configure pre-defined transformation scenarios.

- **Transform with**
  Allows you to select a transformation scenario to be applied to the currently selected files.

Related information

- Contextual Project Operations Using 'Main Files' Support (on page 418)
- Quick Validation and Transformation for Main Files (on page 422)

Sharing a Project - Team Collaboration

You can use XML projects to make team collaboration and synergy efficient and effective. Not only can you share the project files and folders, but Oxygen XML Editor also allows you to store preferences, transformation scenarios, and validation scenarios at project level (on page 3222) in a project file (.xpr file extension). It can be saved on a version control system (such as SVN, CVS, or Source Safe) or in a shared folder, so that your team has access to the same resources stored in the project file.

Sharing Preferences (Creating a Project-Level Options File)

To share options that are configured in certain preferences pages, you can store them in a project file (.xpr file extension) that can easily be shared with others. To do so, follow these steps:

1. [Recommended] You may want to use a fresh install for this procedure to ensure that you do not copy personal or local preferences.
2. In the Project view (on page 402), create a project or open an existing one.
3. Open the Preferences dialog box (Options > Preferences) (on page 124).
4. Configure the options in each preferences page that you want to be included in the project file and switch the storage preference to Project Options (on page 3222) in each page.

**Note:**

Some pages do not have the Project Options button, since the options they host might contain sensitive data (such as passwords, for example) that is unsuitable for sharing with other users.
5. Click **OK** and close the **Preferences** dialog box.

All explicitly set values are now saved in the project file. You can then share the project file so that your team will have the same option configuration that you stored in the project file.

**Note:**
The project file extension (**.xpr**) must be preserved when the file is distributed to others.

**Notice:**
When a project is opened for the first time, a confirmation dialog box will be displayed that asks you to confirm that the project came from a trusted source. This is meant to help prevent potential security issues.

### Sharing Transformation Scenarios

To share created and edited transformation scenarios, you can store them in a project file (**.xpr** file extension) by following these steps:

1. In the **Project view** **(on page 402)**, create a project or open an existing one.
2. When you create a new transformation scenario **(on page 1464)** or edit an existing one **(on page 1545)**, there is a **Storage** option. Switch the storage preference to **Project Options** **(on page 3222)** in each transformation scenario you want to be included in the project file.
3. Click **OK** to store the scenario in the project file.

You can then share the project file so that your team will have access to the same transformation scenarios that you stored in the project file. When you create a scenario at the project level, the URLs from the scenario become relative to the project URL.

**Note:**
The project file extension (**.xpr**) must be preserved when the file is distributed to others.

**Notice:**
When a project is opened for the first time, a confirmation dialog box will be displayed that asks you to confirm that the project came from a trusted source. This is meant to help prevent potential security issues.

### Sharing Validation Scenarios

To share created and edited validation scenarios, you can store them in a project file (**.xpr** file extension) by following these steps:
1. In the **Project view (on page 402)**, create a project or open an existing one.

2. When you create a new validation scenario (on page 788) or edit an existing one (on page 798), there is a **Storage** option. Switch the storage preference to **Project Options (on page 3222)** in each validation scenario you want to be included in the project file.

3. Click **OK** to store the scenario in the project file.

You can then share the project file so that your team will have access to the same validation scenarios that you stored in the project file. When you create a scenario at the project level, the URLs from the scenario become relative to the project URL.

**Note:**
The project file extension (`.xpr`) must be preserved when the file is distributed to others.

**Notice:**
When a project is opened for the first time, a confirmation dialog box will be displayed that asks you to confirm that the project came from a trusted source. This is meant to help prevent potential security issues.

---

**Using Git for Collaboration**

To assist you with team collaboration, sharing projects, and version control, an add-on is available that contributes a built-in Git client directly in Oxygen XML Editor. The **Git Client** is developed independent of the normal Oxygen release cycle, so it is updated and improved more frequently than the main application. It is an optional tool so it needs to be installed as an add-on. Once installed, a **Git Staging** view is available that includes various actions that perform common Git commands, such as push, pull, change branch, commit, and more. It also includes a built-in tool for comparing and merging changes.

For more details, see [Git Client Add-on (on page 2515)](on_page_2515).

**Using Subversion (SVN) for Collaboration**

Oxygen XML Editor also includes an embedded **SVN (Subversion) Client (on page 2718)**. Even if you start developing a new project, or you want to migrate an existing one to Subversion, the Syncro SVN Client allows you to easily share it with the rest of your team. It can be accessed from the **Tools** menu and can be used for synchronizing your working copy with a central repository. It can also be started by selecting the **Open in SVN Client** action from the contextual menu of the **Project view (on page 402)**. This action opens the Syncro SVN Client and shows the selected project file in the **Working Copy** view.

---

**Related information**

- [Sharing Application Settings (on page 312)](on_page_312)
- [Sharing Transformation Scenarios (on page 1555)](on_page_1555)
- [Sharing Validation Scenarios (on page 808)](on_page_808)
Minimize Differences Between Versions Saved on Multiple Computers

The number of differences between versions of the same file saved by multiple content authors on multiple computers can be minimized by imposing the same set of formatting options when saving the file, for all the content authors. An example, the following procedure can be used to minimize the differences:

1. Create an Oxygen XML Editor project file (.xpr) that will be shared by all content authors.
2. Configure your own formatting preferences. To do this, open the Preferences dialog box (Options > Preferences) (on page 124), go to Editor > Format, configure the appropriate options in this page, then go to Editor > Format > XML and configure the options there.
3. Save the configured options into your project file by selecting Project Options (on page 3222) in both of the preferences pages.
4. Save the project and commit the project file to your versioning system so all the content authors can use it.
5. Make sure the project is opened in the Project view (on page 402).
6. Open and save your XML files in the Author mode.
7. Commit the saved XML files to your versioning system.

When other content authors change the files, only the changed lines will be displayed in your diff tool instead of one big change that does not allow you to see the changes between two versions of the file.

Contextual Project Operations Using 'Main Files' Support

Oxygen XML Editor allows you to define Main Files (on page 3220) at project level. These main files are automatically used by Oxygen XML Editor to determine the context for operations such as validation, transformation, content completion, refactoring, or searches for XML, XSD, XSL, WSDL, and RNG modules. Oxygen XML Editor maintains the hierarchy of the main files, helping you to determine the editing context.

Oxygen XML Editor also provides unique support for using the Main Files support in DITA projects. In DITA, when you rename or move non-DITA resources, it allows you to update all the references to these resources in the scope of the Main Files (in this case the main DITA map (root map) (on page 3223)). For more information, see Main Files Support in DITA (on page 3181).

For more information about the Main Files support for XML documents, watch our video demonstrations:

https://www.youtube.com/embed/e2oo4RWNxW8

https://www.youtube.com/embed/UZwg385RKNw

https://www.youtube.com/embed/FQNSsg57S4E

https://www.youtube.com/embed/gn_YPD5xDCo
Main Files Benefits

Using the *Main Files* support in Oxygen XML Editor includes the following benefits:

- When the *main file* is validated, Oxygen XML Editor automatically identifies the modules included in the *main file* and validates all of them.
- When the *main file* is transformed, Oxygen XML Editor automatically identifies the modules included in the *main file* and transforms them accordingly.
- The *Content Completion Assistant (on page 3217)* presents all the components that are collected from the *main files* for the modules they include.
- The *Outline view (on page 539)* displays all the components that are defined in the *main files* hierarchy.
- The *main files* that are defined for the current module determine the *scope of the search and refactoring actions (on page 831)*. Oxygen XML Editor performs the search and refactoring actions in the context that the *main files* determine, thus improving the speed of execution.

Enabling the Main Files Support

Oxygen XML Editor stores the *main files* in a folder located in the *Project view (on page 402)*, as the first child of the project root. The *Main Files Support* is disabled by default and Oxygen XML Editor allows you to enable or disable the *Main Files Support* for each project you are working on.

To enable *Main Files* support, do one of the following:

- Select *Enable Main Files Support* from the *Settings* menu in the top-right corner of the *Project view (on page 402)*.
- Select *Enable Main Files Support* from the contextual menu of the project root folder in the *Project view (on page 402)*. If a disabled *Main Files* folder exists, you can also select that option from its contextual menu.
- Click the *Enable* button in the tooltip located at the bottom of the *Project view (on page 402)*. This tooltip window is displayed when the *Main Files* support is disabled. Clicking the *Read more* link takes you to the user guide. Clicking the *Enable* button opens the *Enable Main Files* dialog box. This dialog box contains general information about the *Main Files Support* and allows you to enable it.
can also use the Detect and Enable button in this dialog box to detect the main files from the current project.

**Warning:**
Once you close this window tip, Oxygen XML Editor hides it for all projects. You can make the window tip reappear by resetting Oxygen XML Editor to its default settings (on page 314). However, doing so will result in you losing your customized options.

**Related information**
Detecting Main Files (on page 420)
Adding or Removing Files in the Main Files Directory (on page 421)

## Detecting Main Files

Oxygen XML Editor allows you to detect the main files using the Detect Main Files option. This action applies to the folders you select in the project.

To detect main files over the entire project, do one of the following:

- Right-click the root of the project and select Detect Main Files.
- Use the Detect Main Files from Project option, available in the contextual menu of the Main Files folder.

Both of these options display the Detect Main Files wizard. In the first panel you can select the type of main files you want Oxygen XML Editor to detect. In the subsequent panel the detected main files are presented in a tree-like fashion. The resources are grouped into three categories:

- **Possible main files** - The files presented on the first level in this category are not imported or included from other files. These files are most likely to be set as main files.

**Note:**
For DITA projects, only DITA Maps (on page 3218) are reported as possible main files.

- **Cycles** - The files that are presented on the first level have circular dependencies between them. Any file presented on the first level of a cycle is a possible main file.
- **Standalone** - Files that do not include or import other files and are also not included or imported themselves. It is not necessary to set them as main files.

To set them as main files, simply select their checkboxes. Oxygen XML Editor marks all the children of a main file as modules. Modules are rendered in gray and their tool-tip presents a list of their main files. A module can be accessed from multiple main files.
The next panel displays a list with the selected main files. Click the Finish button to add the main files in the Main Files folder.

You can use the Select Main Files option to automatically mark all main files. This action sets all the resources from the Possible Main Files category and the first resource of each Cycle as main files. The Deselect All button simply removes all of your selections.

Tip:
It is recommended that you only add top-level files (files that are at the root of the include/import graph) in the Main Files directory.

Attention:
If the Main Files Support is disabled, the Main Files directory is rendered only if it contains main files.

Related information
Enabling the Main Files Support (on page 419)
Adding or Removing Files in the Main Files Directory (on page 421)

Adding or Removing Files in the Main Files Directory

Adding Files to the Main File Directory

The Main Files directory only contains logical folders and linked files. To add files in the Main Files directory, use one of the following methods:

- Right-click a file from your project and select Add to Main Files from the contextual menu.
- Select Add Files or Add Edited File from the contextual menu of the Main Files directory.
- Drag and drop files into the Main Files directory.
- From the contextual menu of the Resource Hierarchy Dependencies view (on page 832), use the Add to Main Files action.

You can view the main files for the current resource by selecting Properties from the contextual menu (on page 411) of the Project view (on page 402) and the main files for the current editor in the Properties (on page 396) and Information (on page 512) views.

Removing Files from the Main Files Directory

The main files that are already defined in the project are automatically marked in the tree. To disable a main file, remove it from the Main Files folder (for example, use the Ctrl+X or Delete keyboard shortcuts). Deleting files from the Main Files folder does NOT delete the files from disk. It just removes the logical files from that logical folder.
Quick Validation and Transformation for Main Files

If **Main Files Support** is enabled (on page 419), you can hover the cursor over the Main Files directory, or a node within the directory, and Oxygen XML Editor will display inline ✔️ **Validate** and 🎥 **Transform** buttons that can be used to quickly run a validation or transformation over the directory or node. For nodes within the Main Files directory, hovering over the ✔️ **Validate** and 🎥 **Transform** buttons also displays the most recently used validation or transformation scenario. To change the assigned validation or transformation scenario, right-click the node and select **Validate > 🎥 Configure Validation Scenario(s)** or **Transform > 🎥 Configure Transformation Scenario(s)**, respectively.

Search and Find/Replace Features

Oxygen XML Editor includes advanced search capabilities to help you locate documents and resources. The search features are powered by Apache Lucene. Apache Lucene is a free open source information retrieval software library. You can perform simple text searches or more complex searches using the Apache Lucene - Query Parser Syntax.

**Note:**

When Oxygen XML Editor performs the indexing of resources, referenced content is not taken into account. For example, when DITA documents are indexed, the content referenced in a `@conref` or `@conkeyref` attribute is not parsed. The files that make up the index are stored on disk in the `[user_home_directory]\AppData\Roaming\com.oxygenxml\lucene` folder.

Open/Find Resource View

The **Open/Find Resource** view is designed to offer advanced search capabilities either by using a simple text search or by using the Apache Lucene - Query Parser Syntax. By default, the view is presented in the left side of the Oxygen XML Editor layout, next to the **Project view** (on page 402) or **DITA Maps Manager** (on page 2897). If the view is not displayed, it can be opened by selecting it from the **Window > Show View** menu.
You can use this view to find a file in the current Oxygen XML Editor project or in one of the DITA maps (on page 3218) opened in the DITA Maps Manager view (on page 2897) by typing only a few letters of the file name of a document or a fragment of the content you are searching for. The Open/Find Resource view also supports searching in document edits (comments, tracked change insertions/deletions, and highlighted content) by selecting the In reviews option (on page 424).

**Note:**
Full support for searching in document edits (the In reviews option) is available only in the Enterprise edition of Oxygen XML Editor. The Professional edition offers limited support to search through a maximum of 10 edits.

**Search Results**

Search results are presented instantly, after you finish typing the content. The matching fragments of text are highlighted in the results list displayed in the dialog box. When you open one of the documents from the results list, the matching fragments of text are highlighted in the editing area. To remove the highlighting from your document, close the corresponding tab in the Results view (on page 548) at the bottom of the editor.

To display the search history, position the cursor in the search field and press Ctrl + DownArrow (Command + DownArrow on macOS) or Ctrl + UpArrow (Command + UpArrow on macOS) on your keyboard. Pressing only the DownArrow key moves the selection to the list of results.
Note:
Searches are not case-sensitive. For example, if you search for car you get the same results as when you search for Car.

Tip:
Suffix searches are also supported, both for searching in the content of your resources and in their name. For this, you can use wildcards. If you search for *ing with the in content option selected, you will find documents that contain the word presenting. If you search for */samples/*.*.gif with the in file paths option selected, you will find all the gif images from the samples directory.

Options Available in the View

The Open/Find Resource view offers the following options:

- **Settings** - Drop-down menu that includes the following settings for the view:
  - **Clear Index** - Clears the index.
  - **Show description** - Presents the search results in a more compact form, displaying only the title and the location of the resources.
  - **Options** - Opens the Open/Find Resource preferences page (on page 298) where you can configure various search options. For example, you can specify a Content language that differs from the default UI language in case your document contains multiple languages.
- **In file paths (on page 431)** - Select this option to search for resources by their name or by its path (or a fragment of its path).
- **In content (on page 429)** - Select this option to search through the content of your resources.
- **In reviews (on page 431)** - Select this option to search through the comments, tracked change insertions/deletions, or highlights in your resources.
- **Reindex** - Use this option to reindex your resources.

Contextual Menu Actions

A contextual menu is available on each search result and provides actions applicable to that particular document. These actions include:

- **Open** - Opens the document in one of Oxygen XML Editor internal editors.
- **Open with** - Allows you to choose to open the document in the Internal editor or an external System application.
- **Show in Explorer** - Identifies the document in the system file explorer.
- **Copy Location** - Copies the file path and places it in the clipboard.
Indexing Process

The content of the resources used to search in is parsed from an index. The indexing is performed both automatically and on request. Automatic indexing is performed when you modify, add, or remove resources in the currently indexed project. If the index was never initialized, the index is not updated on project changes.

To improve performance, the indexing process skips the following set of common English words (the so-called stop words): a, an, and, are, as, at, be, but, by, for, if, in, into, is, it, no, not, of, on, or, such, that, the, their, then, there, these, they, this, to, was, will, with. This means that if you are searching for any of these words, the indexing process will not be able to match any of them. However, you can configure the list of stop words in the Open/Find Resource preferences page (on page 298).

Caching Mechanism

When you perform a search, a caching mechanism is used to gather the paths of all files linked in the current project. When the first search is performed, all project files are indexed and added to the cache. The next search operation uses the information extracted from the cache, thus improving the processing time. The cache is kept for the currently loaded project only, so when you perform a search in a new project, the cache is rewritten. Also, the cache is reset when you click the Reindex button.

Important:

Files larger than 2GB are not indexed.

If there is no file found that matches your file pattern or text search, a possible cause is that the file you are searching for was added to the Oxygen XML Editor project after the last caching operation. In this case, re-indexing the project files with the Reindex button enables the file to be found. The date and time of the last index operation are displayed below the file list.

Opening the Results

Once you find the files that you want to open, select them in the list and click the Open button (or double-click them). Each of the selected files is opened in the editor associated with the type of the file (on page 297).

Note:

You can drag a resource from the Open/Find Resource view and drop it in a DocBook, DITA, TEI or XHTML document to create a link to that resource.

For more information about the Open/Find Resource feature and its search capabilities, watch our video demonstration:

https://www.youtube.com/embed/PENoDNdaGao

Related information

Open/Find Resource Dialog Box (on page 426)
Open/Find Resource Dialog Box

The Open/Find Resource dialog box offers advanced search capabilities. To open the dialog box, go to Find > Open/Find Resource (Ctrl + Shift + R (Command + Shift + R on macOS)). You can also click the Open/Find Resource toolbar button or use the Search for file action that is available in some URL input fields.

You can use this dialog box to find a file in the current Oxygen XML Editor project or in one of the DITA maps (on page 3218) opened in the DITA Maps Manager view (on page 2897) by typing a few letters of the file name or a fragment of the content you are searching for. The Open/Find Resource dialog box also supports searching in document edits (comments, tracked change insertions/deletions, and highlighted content).

Note:

Full support for searching in document edits (the In reviews option) is available only in the Enterprise edition of Oxygen XML Editor. The Professional edition offers limited support to search through a maximum of 10 edits.

Search Results

Search results are presented instantly, after you finish typing the content. The matching fragments of text are highlighted in the results list displayed in the dialog box. When you open one of the documents from the results list, the matching fragments of text are highlighted in the editing area. To remove the highlighting from your document, close the corresponding tab in the Results view (on page 548) at the bottom of the editor.
To display the search history, position the cursor in the search field and press Ctrl + DownArrow (Command + DownArrow on macOS) or Ctrl + UpArrow (Command + UpArrow on macOS) on your keyboard. Pressing only the DownArrow key moves the selection to the list of results.

**Note:**

Searches are not case-sensitive. For example, if you search for `car` you get the same results as when you search for `Car`.

**Tip:**

Suffix searches are also supported, both for searching in the content of your resources and in their name. For this, you can use wildcards. If you search for `*ing` with the in content option selected, you will find documents that contain the word `presenting`. If you search for `*/samples/*.*.gif` with the in file paths option selected, you will find all the `gif` images from the `samples` directory.

### Options Available in the Dialog Box

The **Open/Find Resource** dialog box includes the following options:

- **In file paths (on page 431)** - Select this option to search for resources by their name or by its path (or a fragment of its path).
- **In content (on page 429)** - Select this option to search through the content of your resources.
- **In reviews (on page 431)** - Select this option to search through the comments, tracked change insertions/deletions, or highlights in your resources.
- **Options** - Opens the **Open/Find Resource** preferences page (on page 298) where you can configure various search options. For example, you can specify a Content language that differs from the default UI language in case your document contains multiple languages.
- **Clear Index** - Clears the index.
- **Reindex** - Use this option to reindex your resources.

### Contextual Menu Actions

A contextual menu is available on each search result and provides actions applicable to that particular document. These actions include:

- **Open** - Opens the document in one of Oxygen XML Editor internal editors.
- **Open with** - Allows you to choose to open the document in the Internal editor or an external System application.
- **Show in Explorer** - Identifies the document in the system file explorer.
- **Copy Location** - Copies the file path and places it in the clipboard.
Indexing Process

The content of the resources used to search in is parsed from an index. The indexing is performed both automatically and on request. Automatic indexing is performed when you modify, add, or remove resources in the currently indexed project. If the index was never initialized, the index in not updated on project changes.

To improve performance, the indexing process skips the following set of common English words (the so-called stop words): a, an, and, are, as, at, be, but, by, for, if, in, into, is, it, no, not, of, on, or, such, that, the, their, then, there, these, they, this, to, was, will, with. This means that if you are searching for any of these words, the indexing process will not be able to match any of them. However, you can configure the list of stop words in the Open/Find Resource preferences page (on page 298).

Caching Mechanism

When you perform a search, a caching mechanism is used to gather the paths of all files linked in the current project. When the first search is performed, all project files are indexed and added to the cache. The next search operation uses the information extracted from the cache, thus improving the processing time. The cache is kept for the currently loaded project only, so when you perform a search in a new project, the cache is rewritten. Also, the cache is reset when you click the Reindex button.

Important:
Files larger than 2GB are not indexed.

If there is no file found that matches your file pattern or text search, a possible cause is that the file you are searching for was added to the Oxygen XML Editor project after the last caching operation. In this case, re-indexing the project files with the Reindex button enables the file to be found. The date and time of the last index operation are displayed below the file list.

Opening the Results

Once you find the files that you want to open, select them in the list and click the Open button (or double-click them). Each of the selected files is opened in the editor associated with the type of the file (on page 297).

For more information about the Open/Find Resource feature and its search capabilities, watch our video demonstration:

https://www.youtube.com/embed/PENoDNdaGao

Related information
Open/Find Resource View (on page 422)
Open/Find Resource Preferences Page (on page 298)
Searching in Content

To perform a search through the content of your resources, open the Open/Find Resource dialog box (on page 426) from the Find menu or with Ctrl + Shift + R (Command + Shift + R on macOS) or the Open/Find Resource view (on page 422) (by default, located on the left side of the editor), select the in content option, and in the search field enter the terms that you want to search for.

The Open/Find Resource feature is powered by Apache Lucene. Apache Lucene is a free open source information retrieval software library.

You can use the Open/Find Resource feature to either perform a simple text search or a more complex search using the Apache Lucene - Query Parser Syntax.

Complex Query Patterns Using Lucene Syntax

Using the Apache Lucene - Query Parser Syntax means you can perform any of the following searches:

- **Term Searches**

  Searching for plain text:

  Garden Preparation

- **Element-Specific Searches**

  Searching for content that belongs to a specific element:

  title: "Garden Preparation"

- **Wildcard Searches**

  Using wildcards to make your search more permissive:

  Garden Prepar?tion

- **Fuzzy Searches**

  If you are not sure of the exact form of a term that you are interested in, use the fuzzy search to find the terms that are similar to the search term. To perform a fuzzy search, use the ~ symbol after the word that you are not sure of:

  Garden Preparing~

- **Proximity Searches**

  Use proximity searches to find words that are within a specific distance away. To perform a proximity search, use the ~ symbol at the end of your search. For example, to search for the word Garden and the word Preparation within 6 words of each other use:

  "Garden Preparation"~6

- **Range Searches**
Use range searches to match documents whose element values are between the lower and upper bound specified in the range query. For example, to find all documents whose titles are between Iris and Lilac, use:

```
title:{Iris TO Lilac}
```

The curly brackets denote an exclusive query. The results you get when using this query are all the documents whose titles are between Iris and Lilac, but not including Iris and Lilac. To create an inclusive query use square brackets:

```
title:[Iris to Lilac]
```

**Term Boosting Searches**

Use term prioritising searches if the fragment of text that you are searching for contains certain words that are more important to your search than the rest of them. For example, if you are searching for Autumn Flower, a good idea is to prioritize the word Autumn since the word Flower occurs more often. To prioritize a word use the ^ symbol:

```
Autumn^6 Flower
```

**Searches Using Boolean Operators**

You can use the AND, +, OR, -, and NOT operators.

To search for documents that contain both the words Garden and Preparation, use:

```
Garden AND Preparation
```

To search for documents that must contain the word Garden and may contain the word Preparation, use:

```
+Garden Preparation
```

To search for documents that contain either the word Garden or the word Preparation, use:

```
Garden OR Preparation
```

To search for documents that contain Garden Preparation but not Preparation of the Flowers, use:

```
"Garden Preparation" - "Preparation of the Flowers"
```

**Searches Using Grouping**

To search either for the word Garden or Preparation, and the word Flowers, use:

```
(Garden OR Preparation) AND Flowers
```

**Searches Using Element Grouping**

To search for a title that contains both the word Flowers and the phrase Garden Preparation, use:
• **Searching for Special Characters**

Sometimes you might need to search your content for special character, such as:

```
+ - & | ^ \ ~ * : \\
```

In this case, you should surround your search query with quotes. For example, to search for (Hydrogen + Oxygen)=Water, use:

```
"(Hydrogen + Oxygen) = Water"
```

### Searching in File Paths

To perform a search in the file paths of your resources, open the Open/Find Resource dialog box *(on page 426)* (from the Find menu or with Ctrl + Shift + R *(Command + Shift + R on macOS)*) or the Open/Find Resource view *(on page 422)* (by default, located on the left side of the editor), select the In file paths option, and in the search field enter the terms that you want to search for.

The Open/Find Resource feature allows you to search for a resource either by its name or by its path (or by a fragment of its path).

You can use wildcards when you perform such searches:

- Use "+*" to match any sequence of characters.
- Use "?" to match any single character.

For example, if you search for "*-preferences-page" you will find all the resources that contain the -preferences-page fragment in their name. If you search for "*/samples/*.gif", you will find all the .gif images from the samples directory.

### Searching in Reviews

To perform a search in the edits of your resources, open the Open/Find Resource dialog box *(on page 426)* (from the Find menu or with Ctrl + Shift + R *(Command + Shift + R on macOS)*) or the Open/Find Resource view *(on page 422)* (by default, located on the left side of the editor), select the In reviews option, and in the search field enter the terms that you want to search for.

The following options are available:

- **Type** - Specifies whether you want to search for content in comments, tracked change insertions/deletions, or highlighted content.
- **Author** - Displays all the authors of the edits in your resources. The authors are collected when indexing. You can set a specific author for your search or search all of them.
- **Time** - Specifies the time when the edits that you are searching through were created.
Both the view and the dialog box display the edits that contain the search results and their parent topics along with a short description. To hide this description, go to Settings and deselect the Show Description option.

### Find/Replace Dialog Box

To open the Find/Replace dialog box, use the Find/Replace action that is available in the Find menu, on the toolbar, or by pressing Ctrl + F (Command + F on macOS). It is also invoked by the Find/Replace contextual menu action found in certain views.

You can use the Find/Replace dialog box to perform the following operations:

- Replace occurrences of target defined in the Find area with a new fragment of text defined in Replace with area.
- Find all the occurrences of a word or string of characters (that can span over multiple lines) in the document you are editing. This operation also takes into account all the whitespaces contained in the fragment you are searching for. The Find/Replace dialog box counts the number of occurrences of the text you are searching for and displays it at the bottom of the dialog box, above the Close button. This number is also displayed in the Results view (on page 548) after you click the Find All button.

The find operation works on multiple lines, meaning that a find match can cover characters on multiple lines of text. To input multiple-line text in the Find and Replace with areas, do one of the following:

- Press Ctrl + Enter (Command + Enter on macOS) on your keyboard.
- Use the Insert newline contextual menu action.

You can use Perl-like regular expressions syntax (on page 446) to define patterns. A content completion assistance window is available in the Find and Replace with areas to help you edit regular expressions. It is activated every time you type \ (backslash key) or on-demand if you press Ctrl + Space (Command + Space on macOS) on your keyboard.

The replace operation can bind regular expression capturing groups ($1, $2, etc.) from the find pattern.

**Tip:**

To replace the `<tag-name>` start tag and its attributes with the `<new-tag-name>` tag use as Find the expression `<tag-name>(\s+)(.*)>` and as Replace with the expression `<new-tag-name$1$2>`.
Find/Replace Dialog Box

The **Find/Replace** dialog box contains the following options:

**Find text area box**

This is where you enter the character string to search for. You can search for Unicode characters specified in the \uNNNN format. Also, hexadecimal notation (\xNNNN) and octal notation (\0NNNN) can be used. In this case you have to select the **Regular expression option** (on page 435). For example, to search for a space character you can use the \u0020 code.

You can use the **History** button to select from a list of the most recently used expressions. Use the **Clear history** action from the bottom of the lists to remove these expressions.

**Replace with text area box**

The character string with which to replace the target. The string for replace can be on a line or on multiple lines. It can contain Perl notation capturing groups, only if the search expression is a regular expression and the **Regular expression option** (on page 435) is selected.
Note:
Some regular expressions can indefinitely block the application. If the execution of the regular expression does not end in about 5 seconds, the application displays a dialog box that allows you to interrupt the operation.

Tip:
Special characters such as newline and tab can be inserted in the Find and Replace with text boxes using dedicated actions in the contextual menu (Insert newline and Insert tab).

Unicode characters in the \uNNNN format can also be used in the Replace with area.

You can use the History button to select from a list of the most recently used expressions. Use the Clear history action from the bottom of the lists to remove these expressions.

XPath

The XPath 2.0 expression you input in this combo is used for restricting the search scope. The cursor position does not affect the result of the XPath evaluation. The context of the XPath expression evaluation from the Find/Replace dialog box is the XML document root. The XPath is used for determining the intervals to be searched from the document, so the XPath result must be a node-set.

Tip:
You can use the Content Completion Assistant (on page 3217) to help you input XPath expressions that are valid in the current context. See Working with XPath Expressions (on page 1999) for more information and some common examples of how to write XPath expressions.

Clicking the XPath Options button opens a preferences page where you can configure some XPath-related options.

Direction
Specifies if the search direction is from current position to end of file (Forward) or to start of file (Backward).

Scope
Specifies whether the Find/Replace operation is executed over the entire content of the edited document (All option), or over the selected content/lines.

Options section

Case sensitive
When selected, the search operation follows the exact letter case of the text entered in the **Find** field.

**Incremental**

The search operation is started every time you type or delete a letter in the **Find** text box.

**Wrap around**

When the end of the document is reached, the search operation is continued from the start of the document, until its entire content is covered.

**Ignore extra whitespaces**

If selected, the application normalizes the content (collapses any sequence of whitespace characters into a single space) and trims its leading and trailing whitespaces when performing the search operation. This is helpful when searching for spaced-separated words since line breaks and indentation between words will not affect the results. This option is automatically disabled if the **Regular expression** option is selected.

**Whole words only**

Only entire occurrences of a word are included in the search operation. This option is automatically disabled if the **Regular expression** option is selected.

**Regular expression**

When this option is selected, you can use regular expressions in Perl-like regular expressions syntax *(on page 446)* to look for specific pieces of text.

- **Dot matches all** - A dot used in a regular expression also matches end of line characters.
- **Canonical equivalence** - If selected, two characters will be considered a match if, and only if, their full canonical *(on page 3217)* decompositions match. For example, the á symbol can be inserted as a single character or as two characters (the a character followed by the tilde ~ character). This option is not selected by default.

**Find All Elements link**

Available when editing in **Author** mode, you can use this link to extend the search scope to XML-specific markup (names and values of both attributes and elements).

**Find button**

Executes a find operation for the next occurrence of the target. It stops after highlighting the find match in the editor panel.

**Replace/Find button**
Executes a replace operation for the target followed by a find operation for the next occurrence.

**Replace button**

Executes a replace operation for the target without going to the next occurrence.

**Find All button**

Executes a find operation and displays all results in the Results view (on page 548).

**Replace All button**

Executes a replace operation in the entire scope of the document.

**Replace to End button**

Executes a replace operation starting from current target until the end of the document, in the direction specified by the current selection of the Direction switch (Forward or Backward).

**Find/Replace in Multiple Files**

The Find/Replace in Files feature enables you to define Search for or Search for and Replace operations across multiple files (for example, in DITA projects you can search in the scope of an entire DITA map (on page 3218)). To open the Find/Replace in Files dialog box, use the Find/Replace in Files action that is available in the following locations:

- The Find menu.
- The Find/Replace in Files button on the main toolbar.
- The contextual menu of the DITA Maps Manager view (on page 2897).
- The contextual menu of the Project view (on page 402).
- The contextual menu of the Data Source Explorer view (on page 2015) for most types of database connections.

The operation works on both local and remote files from an (S)FTP, WebDAV, or CMS server.
Find/Replace in Files Dialog Box

Figure 73. Find / Replace in Files Dialog Box (When Opened from the Toolbar Button)

The dialog box contains the following options:

**Text to Find section**

The first text field is where you enter the character string to search for. You can search for Unicode characters specified in the \uNNNN format. Also, hexadecimal notation (\xNNNN) and octal notation (\0NNNN) can be used. In this case you have to select the Regular expression option. For example, to search for a space character you can use the \u0020 code.

The rest of the options in this section can be used to refine your search:

**Case sensitive**

When selected, the search operation follows the exact letter case of the value entered in the Text to find field.

**Whole words only**
Only entire occurrences of a word are included in the search operation. This option is automatically disabled if either the **Ignore extra whitespaces** or **Regular expression** options are selected.

**Ignore extra whitespaces**

If selected, the application normalizes the content (collapses any sequence of whitespace characters into a single space) and trims its leading and trailing whitespaces when performing the search operation. This is helpful when searching for spaced-separated words since line breaks and indentation between words will not affect the results. This option is automatically disabled if the **Regular expression** option is selected.

**Regular expression**

When this option is selected, you can use regular expressions in Perl-like regular expressions syntax ([on page 446](#)) to look for specific pieces of text.

- **Dot matches all** - A dot used in a regular expression also matches end of line characters.
- **Canonical equivalence** - If selected, two characters will be considered a match if, and only if, their full canonical ([on page 3217](#)) decompositions match. For example, the å symbol can be inserted as a single character or as two characters (the a character followed by the tilde ~ character). This option is not selected by default.

**Restrict to XPath**

The XPath 2.0 expression you input in this combo is used for restricting the search scope. The XPath is used for determining the intervals to be searched from the document, so the XPath result must be a node-set.

**Tip:**

You can use the Content Completion Assistant ([on page 3217](#)) to help you input XPath expressions that are valid in the current context. See Working with XPath Expressions ([on page 1999](#)) for more information and some common examples of how to write XPath expressions.

Clicking the **XPath Options** button opens a preferences page where you can configure some XPath-related options.

**Enable XML search options**

This option is only available when editing in **Text** mode. It provides access to a set of options that allow you to search specific XML component types:
• **Element names** - Only the element names are included in the search operation that ignores XML-tag notations (`<`, `/`, `>`), attributes or white-spaces.

• **Element contents** - Search in the text content of XML elements.

• **Attribute names** - Only the attribute names are included in the search operation, without the leading or trailing white-spaces.

• **Attribute values** - Only the attribute values are included in the search operation, without single quotes(′) or double quotes(″).

• **Comments** - Only the content of comments is included in the search operation, excluding the XML comment delimiters (`<!--`, `-->`).

• **PIs** (Processing Instructions) - Only the content is searched, skipping `<?...?>` (for example, `<?processing instruction?>`).

• **CDATA** - Searches inside content of CDATA sections.

• **DOCTYPE** - Searches inside content of DOCTYPE sections.

• **Entities** - Only the entity names are searched.

The two buttons **Select All** and **Deselect All** allow a simple activation and deactivation of all types of XML components.

**Note:**

Even if you select all options of the **Enable XML search options** section, the search is still XML-aware. If you want to perform the search over the entire file content, deselect **Enable XML search options**.

**Replace with section**

Use the text field in this section to specify a character string to replace the target with. It may contain regular expression group markers if the search expression is a regular expression and the **Regular expression** checkbox is selected.

**Tip:**

If you want to change the XML structure, you could use the **built-in XML refactoring operations** (on page 844). You can even **customize your own refactoring operations** (on page 856).

**Make backup files with extension**

In the replace process Oxygen XML Editor makes backup files of the modified files. The default extension is `.bak`, but you can change the extension as you prefer.

**Scope section**

The options available in this section depend on the context (how the dialog box was opened). Select one of the listed options to specify the scope for the operation. The possible options include:
Selected project resources

Searches only in the selected files.

Project files

Searches in all files from the current project.

All opened files

Searches in all files opened in Oxygen XML Editor (regular files or DITA maps). You are prompted to save all modified files before any operation is performed.

Current file directory

The search is done in the directory of the file opened in the current editor panel. If there is no open file, this option is not available.

Current DITA Map hierarchy (only available if opened from the DITA Maps Manager)

The search is done in all maps and topics referenced by the currently selected DITA map, opened in the DITA Maps Manager view (on page 2897).

Selected references (only available if opened from the DITA Maps Manager)

Searches only in the selected DITA references.

Opened archive (only available if opened from the Archive Browser view)

The search is done in an archive opened in the Archive Browser (on page 2008) view.

Specified path

Use this option to specify the search path.

Filters section

The options available in this section depend on the context (how the dialog box was opened) and they can be used to filter the search operation. The possible options include:

Include files

Narrows the scope of the operation only to the files that match the given filters. For example, you can choose to filter the search to only include files with a certain file extension (such as *.xml).

Recurse subdirectories

When selected, the search is performed recursively for the specified scope.The one exception is that this option is ignored if the scope is set to All opened files.\n
Recurse references (only available if opened from the DITA Maps Manager)

When selected, the search is performed recursively for the selected scope.

Include hidden files

When selected, the search is also performed in the hidden files.
Include archives

When selected, the search is also done in all individual file entries from all supported ZIP-type archives.

Show separate results for each search expression

When selected, the application opens a new tab to display the result of each new search expression. When the option is unchecked, the search results are displayed in the Find in Files tab, replacing any previous search results.

Always open selected results in Text mode

If selected, double-clicking results will always open the documents in Text mode (even if the particular document type is set to open in Author mode, by default). If not selected (default state), double-clicking results will open the documents in whatever editing mode is specified as the default for that document type. For example, by default, DITA documents will open in Author mode (as specified in the default framework configuration for DITA document types). Specialized XML documents such as XSLT or XML Schema will continue being opened in the Text editing mode.

Find All

Use the Find All button to execute the search operation. The results are displayed in a view (on page 548) that allows grouping the results as a tree with two levels.

Replace All

Use the Replace All button to execute the search operation and replace all occurrences with the specified string. When you replace a fragment of text, Oxygen XML Editor offers an option to preview of the changes you make. The Preview dialog box is divided in two sections. The first section presents a list of all the documents containing the fragment of text you want to modify. The second section offers a view of the original file and a view of the final result. It also allows you to highlight all changes using the vertical bar from the right side of the view. The Next change and Previous change buttons allow you to navigate through the changes displayed in the Preview dialog box.

⚠️ CAUTION:

Use the Replace All option with caution. Global searches may result in matching strings being replaced in instances that were not originally intended.

Note:

- You can use Perl-like regular expression syntax (on page 446) to match patterns in text content. The replace operation can bind regular expression capturing groups ($1, $2, etc.) from the find pattern.
- Exclusion patterns are accepted. For example, *.java, !*Test.java would search for all files with a .java extension, with the exception of any file whose name ends in Test.
• To replace the `<tag-name>` start tag and its attributes with the `<new-tag-name>` tag use as **Text to find** the expression `<tag-name(\s+)(.*)>` and as **Replace with** the expression `<new-tag-name $1$2>`.

• The encoding used to read and write the files is detected from the XML header or from the BOM. If a file does not have an XML header or BOM Oxygen XML Editor uses by default the UTF-8 encoding for files of type XML, that is for files with one of the extensions: `.xml`, `.xsl`, `.fo`, `.xsd`, `.rng`, `.nvdl`, `.sch`, `.wsdl` or an extension associated with the XML editor type *(on page 297)*. For the other files it uses the encoding configured for non-XML files *(on page 168)*.

• You can cancel a long operation at any time by pressing the **Cancel** button of the progress dialog box, but doing so will not revert any replacements that have been processed up to that point.

• Since the content of read-only files cannot be modified, the **Replace** operation does not process those files. For every such file, a warning message is displayed in the message panel.

---

**Find All Elements Dialog Box**

To open the **Find All Elements** dialog box, go to **Find > Find All Elements** *(Ctrl + Shift + E (Command + Shift + E on macOS))* or from the shortcut **Find All Elements** that is available in the **Find / Replace** dialog box *(on page 432)*. It assists you in defining XML element / attribute search operations in the current document.

![Figure 74. Find All Elements Dialog Box](image)

The dialog box can perform the following actions:
• Find all the elements with a specified name.
• Find all the elements that contain, or does not contain, a specified string in their text content.
• Find all the elements that have a specified attribute.
• Find all the elements that have an attribute with, or without, a specified value.

You can combine all of these search criteria to filter your results.

The following fields are available in the dialog box:

• **Element name** - The qualified name of the target element to search for. You can use the drop-down menu to find an element or enter it manually. It is populated with valid element names collected from the associated schema. To specify any element name, leave the field empty.

  **Note:**
  Use the qualified name of the element (\(<\text{namespace prefix}>::<\text{element name}>\)) when the document uses this element notation.

• **Element text** - The target element text to search for. The drop-down menu beside this field allows you to specify whether you are looking for an exact or partial match of the element text. For any element text, select **contains** from the drop-down menu and leave the field empty. If you leave the field empty but select **equals** from the drop-down menu, only elements with no text will be found. Select **not contains** to find all elements that do not include the specified text.

• **Attribute name** - The name of the attribute that must be present in the element. You can use the drop-down menu to select an attribute or enter it manually. It is populated with valid attribute names collected from the associated schema. For any or no attribute name, leave the field empty.

  **Note:**
  Use the qualified name of the attribute (\(<\text{namespace prefix}>::<\text{attribute name}>\)) when the document uses this attribute notation.

• **Attribute value** - The drop-down menu beside this field allows you to specify that you are looking for an exact or partial match of the attribute value. For any or no attribute value, select **contains** from the drop-down menu and leave the field empty. If you leave the field empty but select **equals** from the drop-down menu, only elements that have at least an attribute with an empty value will be found. Select **not contains** to find all elements that have attributes without a specified value.

• **Case sensitive** - When this option is selected, operations are case-sensitive.

When you select **Find All**, Oxygen XML Editor tries to find the items that match all the search parameters. The results of the operation are presented as a list in the message panel.
Find and Invoke Actions

Oxygen XML Editor includes a **Find action** feature that provides a quick way to find actions that are available throughout the application. You can also assign shortcuts for particular actions and invoke actions using this feature.

The **Find action** operation is available in the **Find** or **Help** menus and it opens a pop-up window where all the actions are presented in a sortable, filterable table.

![Find Action Pop-Up Window](image)

This pop-up window includes the following features, options, and controls:

**Search Field**

You can use the search field at the top to search for a specific action and it includes a history drop-down menu for quickly preforming recently-used search criteria. You can use the **Delete** button to the right of the search field to clear the current text from the search field.

You can also search for actions using certain keyboard shortcuts (excluding the common editing commands such as **Delete**, **Home**, **End**, **Delete**, **Ctrl+A**, **Ctrl+C**, **Ctrl+V**, etc.)

**Filtering Options**

**All actions**

Filters the table to display all available actions.

**Actions in editing context**

Filters the table to display available actions based on the current editing context where the application is focused (for example, if the current focus is a particular side-view, the table displays actions that are available in that side-view).

**Disabled actions**

Filters the table to also display actions that are currently disabled.
Double-Click

You can double-click an action in the table (or select an action and press Enter) to execute the particular action. Some actions will not have an effect if they are not allowed in the current editing context.

Accessibility Shortcuts

The following keyboard shortcuts can be used to enjoy this feature using only a keyboard:

- **Ctrl + Alt + K** - Opens the Find action pop-up window feature.
- **Up arrow / Down arrow** - Navigates the table vertically and switches from the search field to the table, and vice versa.
- **Tab / Shift + Tab** - Navigates between the radio filtering options and the checkbox option.
- **Left arrow / Right arrow** - Toggles the selection for the radio filtering options.
- **Space** - Toggles the checkbox option.
- **Enter** - Executes the selected action.
- **Ctrl + Enter** - Opens a dialog box where you can assign a keyboard shortcut for the selected action.
- **Ctrl + Up arrow / Ctrl + Down arrow** - Accesses the history drop-down when the search field is in focus.
- **Esc** - Closes the Find action pop-up window feature.

Actions Table

Displays the available actions based on the selected filtering options or search criteria. Some actions might be disabled/deactivated depending on the current editing context. When the Disabled action filtering option is selected, the disabled actions are displayed at the end of the results in the table.

**Note:**

It is possible for certain actions not to be displayed in the actions table if they are created and implemented in other ways (for example, if they are implemented only to be available in a contextual menu).

Quick Find Toolbar

A reduced version of the Find / Replace dialog box (on page 432) is available as a dockable toolbar (on page 360). To display it, press the Alt + Shift + F (Command + Alt + F on macOS) key combination or select the Find > Quick Find action. By default, the toolbar is displayed at the bottom of the Oxygen XML Editor window, above the status bar, but can be changed at any time by dragging (and docking) it to a different location. To hide the toolbar, use the Close button.

All matches are highlighted in the current editor.
Figure 76. Quick Find Toolbar

The toolbar offers the following controls:

- **Search input box** - This is where you can insert the text you want to search for. The input box keeps a history of the last used search text. The background color of the input box turns red when no match is found.
- **Next** - Advances to the next match. You can also use the **Enter** key to jump forward to the next match.
- **Previous** - Jumps to the previous match. You can also use **Shift+Enter** to jump backward to the previous match.
- **All** - Highlights all matches of the search string in the current document.
- **Incremental** - If selected, the search operation is started every time you type or delete a character in the search input box.
- **Case sensitive** - If selected, the search operation follows the exact letter case of the search text.
- **Find/Replace** - Opens the **Find/Replace** dialog box (on page 432).
- **Find/Replace in Files** - Opens the **Find/Replace in Files** dialog box (on page 436).
- **Close** - Closes the Quick Find toolbar.

Keyboard Shortcuts for Finding the Next and Previous Match

Navigating from one match to the next or previous one is very easy to perform using the **F3** and **Shift + F3** (Command + Shift + G on macOS) keyboard shortcuts. They are useful for quickly repeating the last find action performed in the **Find / Replace** dialog box (on page 432), taking into account the same find options.

**Restriction:**

These shortcuts only take XPath expressions into account if the **Find / Replace** dialog box remains opened. Once you close it, the XPath expressions are no longer considered.

Regular Expressions Syntax

Oxygen XML Editor uses the **Java regular expression syntax**. It is similar to that used in Perl 5, with several exceptions. Thus, Oxygen XML Editor does not support the following constructs:

- The conditional constructs `{?(X)}` and `{?(condition)X|Y}`.
- The embedded code constructs `{?{code}}` and `{??{code}}`.
- The embedded comment syntax `{?#{comment}}`.
- The preprocessing operations \l, \u, \L, and \U.

When using regular expressions, note that some sets of characters from **XPath/XML Schema/Schematron** are slightly different than the ones used by Oxygen XML Editor/Java in the text searches from the **Find/Replace** dialog box (on page 432) and **Find/Replace in Files** dialog box (on page 436). The most common example is with the \w and \W set of characters. To ensure consistent results between the two, it is recommended that
You use the following constructs in the Find/Replace dialog box (on page 432) and Find/Replace in Files dialog box (on page 436):

- `/w - [{\#x0000-\#x10FFFF}-]\p{P}\p{Z}\p{C}]` instead of `\w`
- `/W - [\p{P}\p{Z}\p{C}]` instead of `\W`

There are some other notable differences that may cause unexpected results, including the following:

- In Perl, `\1` through `\9` are always interpreted as back references. A backslash-escaped number greater than 9 is treated as a back reference if at least that many sub-expressions exist. Otherwise, it is interpreted, if possible, as an octal escape. In this class octal escapes must always begin with a zero. In Java, `\1` through `\9` are always interpreted as back references, and a larger number is accepted as a back reference if at least that many sub-expressions exist at that point in the regular expression. Otherwise, the parser will drop digits until the number is smaller or equal to the existing number of groups or it is one digit.
- Perl uses the `g` flag to request a match that resumes where the last match left off.
- In Perl, embedded flags at the top level of an expression affect the whole expression. In Java, embedded flags always take effect at the point where they appear, whether they are at the top level or within a group. In the latter case, flags are restored at the end of the group just as in Perl.
- Perl is forgiving about malformed matching constructs, as in the expression `*a`, as well as dangling brackets, as in the expression `abc]`, and treats them as literals. This class also accepts dangling brackets but is strict about dangling meta-characters such as `+`, `?` and `*`.

**Spell Checking**

Oxygen XML Editor includes an automatic (as-you-type) spell checking feature (on page 456), as well as a manual spell checking action to open a Spelling dialog box that offers a variety of options.

To manually check spelling in the current document, use the ABC Check Spelling action on the toolbar or from the Edit menu.
The **Spelling** dialog box contains the following:

**Unrecognized word**

Displays the word that cannot be found in the selected dictionary. The word is also highlighted in the XML document.

**Replace with**

The character string that will replace the misspelled word.

**Guess**

Displays a list of suggested words to replace the unknown word. Double-click a word to automatically insert it in the document and resume the spell checking process.

**Default language**

Allows you to select the default language dictionary used by the spelling engine.

**Paragraph language**

In an XML document, you can mix content written in multiple languages. You can set the language code in the `@lang` or `@xml:lang` attribute for any particular section and Oxygen XML Editor will automatically instruct the spell checker engine to apply the appropriate language dictionary for that section.

**Begin at cursor position**
Instructs the spell checker to begin checking the document starting from the current cursor position.

**Action Buttons**

**Replace**

Use this button to replace the unrecognized word with the selected word from the Replace with field.

**Replace All**

Use this button to replace all occurrences of the unrecognized word with the selected word from the Replace with field, starting from the cursor's position to the end of the document.

*Note:*

This action is case-sensitive.

**Ignore**

Ignores the first occurrence of the unrecognized word and allows you to continue checking the document. Oxygen XML Editor skips the content of the XML elements marked to be ignored *(on page 456).*

**Ignore All**

Ignores all instances of the unrecognized word in the current document.

**Learn**

Adds the unrecognized word to the list of valid words.

**Options**

Opens the Spell Check preferences page *(on page 229)* where you can configure various options regarding the feature.

**Related information**

AutoCorrect Misspelled Words *(on page 460)*

**Spell Check Dictionaries and Term Lists**

Oxygen XML Editor uses the Hunspell engine for the spell checking feature. The Hunspell spell checking engine is open source and has an LGPL license. It is designed for languages with rich morphology and complex compounding or character encoding. Each language-country variant combination have their own specific dictionaries. Oxygen XML Editor includes the following built-in dictionaries for the spell checker:

- English (US) `[en_US]`
- English (UK) `[en_GB]`
- French `[fr]`
Other Hunspell Dictionaries

You can also download Hunspell dictionaries for other languages and add them to the Oxygen XML Editor spell checker. An example of a website that includes numerous dictionary files is: http://extensions.services.openoffice.org/dictionary.

If you cannot find a Hunspell dictionary that is already built for your language, you can build the dictionary you need. To build a full Hunspell dictionary, follow these instructions and then add the dictionary to the Oxygen XML Editor spell checker by following this procedure (on page 451).

Personalized Term Lists

Authoring in certain areas of expertise (for example, the pharmaceutical or automobile industries) might require the use of specific terms that are not part of the standard spell checker dictionary. To avoid marking these terms as errors, Oxygen XML Editor provides a way of adding personalized term lists (on page 453) to the spell check engine. This involves creating a term list file that the spell checker will recognize and it is similar to the file Oxygen XML Editor uses for storing learned words (on page 456).

The term list files are specific for each language and can be specific to each domain or area of expertise (for example, legal, medical, automotive). They can also be used to control forbidden words.

Related information
Adding Custom Spell Check Dictionaries (on page 451)
Adding Custom Spell Check Term Lists (on page 453)
Building and Testing Hunspell Dictionaries

Adding Custom Dictionaries and Term Lists

The Oxygen XML Editor spell checker allows you to add customized Hunspell dictionaries and personalized term lists. The Hunspell dictionary mechanism requires a dictionary file (with a .dic file extension) and an affix file (with the .aff file extension). The personalized term lists are custom files (with the .tdi file extension) that you can create to include specialized terms or specify forbidden words in the Oxygen XML Editor spell checker.

You can add dictionaries (on page 451) and personalized term lists (on page 453) to the default folder where they are stored or specify your own custom locations. You can view the default storage location in the Spell Check Dictionaries preferences page (on page 232) and the Include dictionaries and term list from option (on page 232) allows you to choose a custom storage location. All the dictionaries and term lists for a particular language that are found in either location are merged and used by the spell checker in Oxygen XML Editor.
Adding Custom Spell Check Dictionaries

There are three possible scenarios for adding Hunspell dictionaries to the Oxygen XML Editor spell checker:

- You can download a pre-built Hunspell dictionary and add it to the spell checking mechanism.
- You can create a custom Hunspell dictionary file that defines your own list of words and add it to the spell checking mechanism.
- You can build your own full Hunspell dictionary and add it to the spell checking mechanism.

**Download and Add a Pre-Built Hunspell Dictionary**

To add a downloaded pre-built dictionary, follow these steps:

1. Download the files needed for your dictionary. You will need a *dictionary* file (with a `.dic` file extension) and an *affix* file (with the `.aff` file extension). If the dictionary does not include an affix file (.aff), you can create one and leave it empty, but it is needed for the mechanism to work properly. An example of a website that includes numerous dictionary files is: http://extensions.services.openoffice.org/dictionary.

   - **Important:**
     The name of the files should begin with a two letter prefix for the language code, followed by an underscore or hyphen, then two letters that indicate the country code, followed by another underscore or hyphen, and then a descriptive name (for example, `en_US_medical.dic` for a medical dictionary in the US version of the English language, or for a less specific English medical dictionary, you could omit the country code like this: `en_medical.dic`). For a list of language codes, see https://en.wikipedia.org/wiki/List_of_ISO_639-1_codes.

2. Open the Preferences dialog box (Options > Preferences) (on page 124) and go to Editor > Spell Check > Dictionaries (on page 232).
3. Choose one of the following two options for adding the downloaded files.
   a. Copy both files (.dic and .aff) to the default directory displayed in the Dictionaries and term lists default folder option (on page 232).
   b. Copy both files (.dic and .aff) to any other directory, select the Include dictionaries and term list from option (on page 232), and select that directory. If you choose this option, make sure you read this important note (on page 232).
4. Restart the application for the spell checker to start using the new dictionary.

**Create a Custom Hunspell Dictionary that Defines a List of Words**

To create a custom Hunspell dictionary that defines your own list of words, follow these steps:
1. Create a dictionary file (with a .dic file extension) and an affix file (with the .aff file extension). The affix file (.aff) can be left empty, but it is needed for the mechanism to work properly.

**Important:**

The name of the files should begin with a two letter prefix for the language code, followed by an underscore or hyphen, then two letters that indicate the country code, followed by another underscore or hyphen, and then a descriptive name (for example, en_US_medical.dic for a medical dictionary in the US version of the English language, or for a less specific English medical dictionary, you could omit the country code like this: en_medical.dic). For a list of language codes, see [https://en.wikipedia.org/wiki/List_of_ISO_639-1_codes](https://en.wikipedia.org/wiki/List_of_ISO_639-1_codes).

2. In the dictionary file (.dic extension), add the words you want to be included in your custom dictionary. Add one word per row and the first line needs to contain the number of words, as in the following example:

```
2
parabola
asymptotic
```

**Tip:**

Words stored in dictionaries are not handled as case-sensitive. Therefore, you do not need to include both uppercase and lowercase versions of the words.

**Note:**

If you save the .dic file using UTF-8 encoding, then the corresponding .aff file should specify the encoding as a property inside it (if you do not specify the encoding, the default platform encoding will be used):

```
SET UTF-8
```

3. Open the Preferences dialog box (Options > Preferences) (on page 124) and go to Editor > Spell Check > Dictionaries (on page 232).

4. Choose one of the following two options for saving the files.
   a. Save both files (.dic and .aff) to the default directory displayed in the Dictionaries and term lists default folder option (on page 232).
   b. Save both files (.dic and .aff) to any other directory, select the Include dictionaries and term list from option (on page 232), and select that directory. If you choose this option, make sure you read this important note (on page 232).

5. Restart the application for the spell checker to start using the new dictionary.

**Build and Add a Full Hunspell Dictionary**

To build and add a full Hunspell dictionary, follow these steps:
1. Create your Hunspell dictionary. For more information on how to do this, see: Editing the Spell Checking Dictionaries.

**Step Result:** You should end up with a dictionary file (with a .dic file extension) and an affix file (with an .aff file extension). The affix file (.aff) can be empty, but it is needed for the mechanism to work properly.

**Important:**
The name of the files should begin with a two letter prefix for the language code, followed by an underscore or hyphen, then two letters that indicate the country code, followed by another underscore or hyphen, and then a descriptive name (for example, en_US_medical.dic for a medical dictionary in the US version of the English language, or for a less specific English medical dictionary, you could omit the country code like this: en_medical.dic). For a list of language codes, see https://en.wikipedia.org/wiki/List_of_ISO_639-1_codes.

2. Open the Preferences dialog box (Options > Preferences) (on page 124) and go to Editor > Spell Check > Dictionaries (on page 232).
3. Choose one of the following two options for saving the files.
   a. Save both files (.dic and .aff) to the default directory displayed in the Dictionaries and term lists default folder option (on page 232).
   b. Save both files (.dic and .aff) to any other directory, select the Include dictionaries and term list from option (on page 232), and select that directory. If you choose this option, make sure you read this important note (on page 232).
4. Restart the application for the spell checker to start using the new dictionary.

**Related information**

Adding Custom Spell Check Term Lists (on page 453)
Editing the Spell Checking Dictionaries

**Adding Custom Spell Check Term Lists**

You can create personalized term lists that are used to store specialized terms or control forbidden words. They can then be added to one of the directories that store the spell check dictionaries, and the spell checker will merge them with all the dictionaries and other term lists for a particular language.

**Create and Add Personalized Term Lists**

To create and add a personalized term list, follow these steps:

1. Create a term list file (with the .tdi file extension). The name of the file must begin with a two letter prefix that indicates the language it should be attached to, followed by an underscore or hyphen, and then a descriptive name (for example, en_US_myterms.tdi for term list in the US version of the
English language or en_myterms.tdi for a less specific English term list). For a list of language codes, see https://en.wikipedia.org/wiki/List_of_ISO_639-1_codes.

2. In the term list file (.tdi extension), add the terms you want to be included in your custom dictionary. If you need to specify forbidden terms, those words simply need to be preceded by an asterisk. Add one word per row, as in the following example:

<table>
<thead>
<tr>
<th>word</th>
</tr>
</thead>
<tbody>
<tr>
<td>parabola</td>
</tr>
<tr>
<td>asimptotic</td>
</tr>
<tr>
<td>*hyperbola</td>
</tr>
</tbody>
</table>

Note: Words stored in term lists are not handled as case-sensitive. Therefore, you do not need to include both uppercase and lowercase versions of the words.

3. Open the Preferences dialog box (Options > Preferences) (on page 124) and go to Editor > Spell Check > Dictionaries (on page 232).

4. Choose one of the following two options for saving the file.
   a. Save the file (.tdi) to the default directory displayed in the Dictionaries and term lists default folder option (on page 232).
   b. Save the file (.tdi) to any other directory, select the Include dictionaries and term list from option (on page 232), and select that directory. If you choose this option, make sure you read this important note (on page 232).

5. Restart the application for the spell checker to start using the new term list.

Related information
Adding Custom Spell Check Dictionaries (on page 451)

Replacing a Spell Check Dictionary

There are several possible scenarios for replacing an existing Hunspell dictionary for the Oxygen XML Editor spell checker:

- You can download a pre-built Hunspell dictionary and replace an existing dictionary with it.
- You can build your own full Hunspell dictionary and replace an existing dictionary with it.

Download a Pre-Built Hunspell Dictionary and Replace an Existing One

To replace an existing dictionary with a downloaded pre-built dictionary, follow these steps:

1. Download the files needed for your dictionary. You will need a dictionary file (with a .dic file extension) and an affix file (with the .aff file extension). If the dictionary does not include an affix file (.aff), you can create one and leave it empty, but it is needed for the mechanism to work properly. An example of a website that includes numerous dictionary files is: http://extensions.services.openoffice.org/dictionary.
2. Open the Preferences dialog box (Options > Preferences) (on page 124) and go to Editor > Spell Check > Dictionaries (on page 232).

3. Choose one of the following two options to replace existing files.
   a. Replace the existing files (.dic and .aff) for the particular language in the default directory displayed in the Dictionaries and term lists default folder option (on page 232). Leave the Include dictionaries and term list from option deselected.
   b. Replace existing files (.dic and .aff) for the particular language in a directory specified in the Include dictionaries and term list from option (on page 232). If you choose this option, make sure you read this important note (on page 232).

   **Important:**
   Do not alter the naming convention. The name of the files must begin with a two letter prefix that indicates the language it should be attached to (for example, `en_US.dic` for a US English dictionary or `en.dic` for a less specific English dictionary). For a list of language codes, see https://en.wikipedia.org/wiki/List_of_ISO_639-1_codes.

4. Restart the application for the spell checker to start using the new dictionary.

**Build a Full Hunspell Dictionary and Replace an Existing One**

To replace an existing dictionary with a full Hunspell dictionary that you build, follow these steps:

1. Follow these instructions: Building and Testing Hunspell Dictionaries.

   **Step Result:** You should end up with a dictionary file (with a .dic file extension) and an affix file (with the .aff file extension). The affix file (.aff) can be empty, but it is needed for the mechanism to work properly.

2. Open the Preferences dialog box (Options > Preferences) (on page 124) and go to Editor > Spell Check > Dictionaries (on page 232).

3. Choose one of the following two options to replace existing files.
   a. Replace the existing files (.dic and .aff) for the particular language in the default directory displayed in the Dictionaries and term lists default folder option (on page 232). Leave the Include dictionaries and term list from option deselected.
   b. Replace existing files (.dic and .aff) for the particular language in a directory specified in the Include dictionaries and term list from option (on page 232). If you choose this option, make sure you read this important note (on page 232).

4. Restart the application for the spell checker to start using the new dictionary.

Related information

Adding Custom Dictionaries and Term Lists (on page 450)
Learned Words

Spell checker engines rely on dictionaries to decide if a word is spelled correctly. To instruct the spell checker engine that an unknown word is actually correctly spelled, you need to add that word to a list of learned words. There are two ways to do this:

- Invoke the contextual menu on an unknown word, then select Learn word.
- Click the Learn button from the Spelling dialog box (on page 447) that is invoked by using the Check Spelling action on the toolbar.

Note:
To delete items from the list of learned words, use the Delete learned words option in the Editor > Spell Check > Dictionaries preferences page (on page 232).

Related information
Adding Custom Spell Check Term Lists (on page 453)

Ignored Words (Elements)

You may want the content of certain XML elements to always be skipped during the spell check process (for example, <programlisting>, <codeblock>, <codeph>, <filepath>, or <screen>). This can be done in one of several ways:

- You can skip through them manually, word by word, using the Ignore button in the Spelling dialog box (on page 447) that is invoked by using the Check Spelling action on the toolbar.
- You can automatically skip the content of certain elements by maintaining a set of known element names that should never be checked. You can manage this set of element names by using the Ignore elements section (on page 231) in the Spell Check preferences page.

Automatic Spell Check

Oxygen XML Editor includes an option to automatically check the spelling as you type. Not only does it check spelling when you are typing in the main editor, but also when you are typing in a comment (on page 653). This feature is disabled by default, but it can be enabled and configured in the Spell Check preferences page (on page 229). When the Automatic Spell Check option (on page 229) is selected, unknown words are underlined and some actions are available in the contextual menu to help you correct the word or prevent the word from being reported in the future.
The contextual menu includes the following actions:

**Delete Repeated Word**

Allows you to delete words that were repeated in consecutive order.

**List of Suggestions**

A list of words suggested by the spell checking engine as possible replacements for the unknown word.

**Learn Word**

Allows you to add the current unknown word to the persistent dictionary of learned words (on page 456).

**Spell check options (Available in Author mode only)**

Opens the Spell Check preferences page (on page 229).

**Other actions**

This submenu give you access to all the usual contextual menu actions.
Spell Check Multiple Files

The Check Spelling in Files action allows you to check the spelling on multiple local or remote documents. This action is available in the following locations:

- The Edit menu.
- The contextual menu of the Project view (on page 402).
- The contextual menu of the DITA Maps Manager view (on page 2897), when editing DITA documents.

This action opens the Check Spelling in Files dialog box that allows you to define the scope and several other options. After you configure the settings for the operation, click the Check All button to check the spelling in all specified files. The spelling corrections are displayed in the Results view (on page 548) at the bottom of the editor and you can group the reported errors as a tree with two levels.

Tip:
If you want to instruct the spell checking engine to not report a particular word as being a spelling error in the future, use the Learn Word(s) action from the contextual menu in the Results view.

Figure 80. Check Spelling in Files Dialog Box (Invoked from Project View)

The following scopes are possible, depending on where the action was invoked:

- **All opened files** - The spell check is performed in all open files.
- **Current file directory** - All the files in the folder of the currently edited file.
• **Current DITA map hierarchy** - Option available when the dialog is invoked from the DITA Maps Manager view. Checks the spelling in all references contained in the DITA map.
• **Project** - All files from the current project.
• **Selected project resources** - The selected files from the current project.
• **Specified path** - Checks the spelling in the files located at a path that you specify.

The **Options** section includes the following options:

• **File filter** - Allows you to filter the files from the selected scope.
• **Recurse subdirectories** - When selected, the spell check is performed recursively for the specified scope. The one exception is that this option is ignored if the scope is set to **All opened files**.
• **Include hidden files** - When selected, the spell check is also performed in the hidden files.
• **Spell Check Options** - The spell check processor uses the options available in the **Spell Check preferences page** (on page 229).

When working with DITA documents, if you invoke the **Check Spelling in Files** action in the DITA Maps Manager view (on page 2897), a slightly different version of the dialog box is displayed:

**Figure 81. Check Spelling in Files Dialog Box (Invoked from the DITA Maps Manager View)**

The following scopes are available when you check the spelling in files from the **DITA Maps Manager** (on page 2897):

• **Current DITA Map hierarchy** - All the files referenced in the currently selected DITA map (on page 3218) from in the DITA Maps Manager view.
• **Selected references** - Checks the spelling in the selected references.
• **Specified path** - Checks the spelling in the files located at a path that you specify.
AutoCorrect Misspelled Words

Oxygen XML Editor includes an AutoCorrect feature to automatically correct misspelled words, as well as to insert certain symbols or other text, as you type in Author mode. Oxygen XML Editor includes a default list of commonly misspelled words and symbols, but you can modify the list to suit your needs. You can also choose to have the AutoCorrect feature use suggestions from the main spell checker. The suggestions will only be used if the misspelled words are not found in the Replacements Table (on page 194).

When enabled, the AutoCorrect feature can be used to do the following:

- Automatically correct misspelled words while you edit in Author mode. The actual operation of replacing a word is triggered by a space, dash, or certain punctuation characters (, . ; : ? ! " ) ] ).
- Easily insert symbols. For example, if you want to insert a ® character, you would type (R).
- Quickly insert text fragments.
- Quickly insert XML fragments. For example, if you enter a hyphen (-) in an empty paragraph followed by a space, it will automatically be converted to a list with a list item.

AutoCorrect is enabled by default. To configure this feature, open the Preferences dialog box (Options > Preferences) (on page 124) and go to Editor > Edit Modes > Author > AutoCorrect.

AutoCorrect Drop-down Actions

After the automatic operation of replacing a misspelled word (triggered by a space, dash, or certain punctuation characters), the affected string is highlighted. The highlight is removed upon the next editing action (text insertion or deletion). If you hover over the highlight, a small widget appears below the word. If you hover over the widget, it expands and you can click it to present a drop-down list that includes the following actions:

- Change back to "[original word]" - Reverts the correction back to its original form.
- Stop Automatically Correcting "[original word]" - This option is presented if the correction is performed based on the AutoCorrect Replacements Table (on page 194) and selecting it will delete the corresponding entry from the Replacements Table.
- Learn Word "[original word]" - This option is presented if the Use additional suggestions from the spell checker option (on page 193) is selected in the AutoCorrect preferences page (on page 193) and the correction is performed based on the Spell Checker. Selecting this option will add the item to the list of learned words (on page 456).
- AutoCorrect options - Opens the AutoCorrect preferences page (on page 193) that allows you to configure the feature.
AutoCorrect Case-Sensitivity

The AutoCorrect feature results in the following types of substitutions regarding case-sensitivity:

- Words with all lower-case characters will be replaced with lower-case substitutions (for example, "abotu" is replaced with "about").
- Words with irregular-case characters will be replaced with lower-case substitutions ("ABotU" is replaced with "about").
- Words with all upper-case characters will be replaced with upper-case substitutions ("ABOTU" is replaced with "ABOUT").
- Words starting with an upper-case character will be replaced with substitutions having the same pattern ("Abotu" is replaced with "About").

Note:
The AutoCorrect feature also uses the list of ignored elements from the Spell Check preferences page (on page 231). All elements (along with their descendant elements) included in this list will be ignored by the AutoCorrect engine.

Add Dictionaries for the AutoCorrect Feature

To add new dictionaries for the AutoCorrect mechanism (on page 460), or to replace an existing one, follow these steps:

1. Download an AutoCorrect dictionary file for the desired language. The file needs to have a .dat file extension. An example of a website that includes some AutoCorrect dictionary files is: OpenOffice Extensions Search Page.
2. Open the Preferences dialog box (Options > Preferences) (on page 124) and go to Editor > Edit Modes > Author > AutoCorrect > Dictionaries (on page 195).
3. Choose one of the following two options for adding the downloaded files:
a. Copy the downloaded .dat file to the default directory displayed in the Dictionaries default folder option. (on page 195). Note that if you are replacing an existing dictionary file, this is the best option.

b. Copy the downloaded .dat file to any other directory, select the Include dictionaries from option (on page 195), and select that directory. If you choose this option, make sure you read this important note (on page 195).

4. Restart the application for the AutoCorrect mechanism to start using the new dictionary.

Working with Special Characters and Encoding

While regular characters make up the English and European alphabets and the corresponding basic set of figures and symbols, there are many other special characters that belong to various other language representations, such as Arabic, Indian, Japanese, Chinese, or Korean. Oxygen XML Editor provides support for special characters in various ways:

Opening and Saving Documents

The Unicode standard provides support for all the character symbols in all known languages and Oxygen XML Editor provides support for all Unicode characters (on page 463). There are various encoding options and features to help determine how to handle documents with unsupported characters (on page 464).

Fonts

Oxygen XML Editor provides the ability to choose the fonts to be used in the various editing modes (on page 133). In some cases, changing the font may be a solution when special characters are not rendered as expected.

For special characters that are not included in any of the default fonts, Oxygen XML Editor tries to find that symbol in a fallback font (on page 464). For the Author editing mode, you can specify a set of fallback fonts in the font-family CSS property (in the particular CSS file used for rendering your documents). For more information, see the CSS Support in Author Mode (on page 2307) section.

Tip:

For documents written in languages that use special characters (such as Japanese or Chinese), change the font to one that supports the specific characters (a Unicode font). For the Windows platform, Arial Unicode MS or MS Gothic is recommended. To change the font in Oxygen XML Editor, open the Preferences dialog box (Options > Preferences) (on page 124), go to Appearance > Fonts. You can select a font for each editing mode in this preferences page.

Navigation and Layout

Oxygen XML Editor supports bidirectional text, such as Arabic, Hebrew, and certain Asian languages, or other special characters that are combined into a single glyph. In Text mode, you
can enable or disable the support for special characters. See Special Character Support in Text Mode (on page 564) for details about which option to choose.

**Editing**

Oxygen XML Editor includes a contextual menu action that converts a sequence of hexadecimal characters to the corresponding Unicode character (on page 569).

If you do not have a special way of inserting special characters using your keyboard, you can insert special characters using the Character Map feature (on page 465).

For more information about working with special characters in specific editing modes, see the following sections:

- Special Character Support in Author Mode (on page 752)
- Special Character Support in Text Mode (on page 564)
- Special Character Support in Grid Mode (on page 586)

**Unicode Support**

Unicode is a standard for providing consistent encoding, representation, and handling of text. There is a unique Unicode number for every character, independent of the platform and language. Unicode is internationally recognized and is required by modern standards (such as XML, Java, JavaScript, LDAP, CORBA 3.0, WML, etc.).

Oxygen XML Editor provides support for the Unicode standard, enabling your XML application to be targeted across multiple platforms, languages, and countries without re-engineering. Internally, the Oxygen XML Editor uses 16-bit characters covering the Unicode Character set.

As a Java application, Oxygen XML Editor includes a default Java input method for typing characters with Unicode codes. However, the default input method does not cover all the Unicode codes (for example, the codes for some accented characters or characters found in East Asian languages). Such characters can be inserted in the editor panel of Oxygen XML Editor either with the Character Map dialog box (on page 467) available from Edit > Insert from Character Map or by installing a Java input method that supports the insertion of the needed characters. The installation of a Java input method depends on the platform (Windows, macOS, Linux, etc.) and is the same for any Java application.

**Note:**

Oxygen XML Editor may not be able to display characters that are not supported by the operating system (either not installed or unavailable).

**Tip:**

On windows, you can enable the support for CJK (Chinese, Japanese, Korean) languages from Control Panel / Regional and Language Options / Languages / Install files for East Asian languages.
Opening and Saving Documents with Unsupported Characters

When loading documents, Oxygen XML Editor reads the document prolog to determine the specified encoding type. This encoding is then used to instruct the Java Encoder to load support for and to save the document using the specified code chart. When the encoding type cannot be determined, Oxygen XML Editor displays the Available Java Encodings dialog box that provides a list of all encodings supported by the Java platform.

Opening Documents with Unsupported Characters

When opening a document in Oxygen XML Editor, if it contains characters that are not supported by the specified encoding standard (these unrecognized characters are rendered as an empty box □), the application determines how to handle them based upon the setting specified in the Encoding Errors Handling option in the Encoding preferences page (on page 169). The default setting is REPORT, which means an error message is displayed for characters that cannot be represented in the specified encoding. If the option is set to REPLACE, the character is replaced with a standard replacement character for the particular encoding. If the option is set to IGNORE, the error is ignored and the character is not rendered.

Saving Documents with Unsupported Characters

When saving a document edited in the Text, Grid, or Design modes, if it contains characters that are not supported by the encoding declared in the document prolog, Oxygen XML Editor displays a notification that you need to resolve the conflict before saving the document.

When saving a document edited in the Author mode, all characters that fall outside the detected encoding will be automatically converted to hexadecimal character entities.

When saving a document with UTF-16 encoding, the saved document has a Byte Order Mark (BOM) that specifies the byte order of the document content. The default byte order is platform-dependent. That means that a UTF-16 document created on a Windows platform (where the default byte order mark is UnicodeLittle) has a different BOM than one created on a macOS platform (where the byte order mark is UnicodeBig). The byte order and the BOM of an existing document are preserved when the document is edited and saved. This behavior can be changed in Oxygen XML Editor from the Encoding preferences page (on page 168).

Unicode Fallback Font Support

Oxygen XML Editor provides fonts for most common Unicode ranges. However, if you use special symbols or characters (on page 465) that are not included in the default fonts, they will be rendered as small rectangles. A fallback font is a reserve typeface that contains symbols for as many Unicode characters (on page 463) as possible. When a display system encounters a character that is not part of the range of any of the available fonts, Oxygen XML Editor will try to find that symbol in a fallback font.
Example of a Scenario Where a Fallback Font is Needed

Suppose that you need to insert the wheelchair symbol (🎎 - U+267F) into your content in a Windows operating system. By default, Oxygen XML Editor does not render this symbol correctly since it is not included in any of the default fonts. It is included in Segoe UI Symbol, but this font is not part of the default fonts that come with Oxygen XML Editor. To allow Oxygen XML Editor to recognize and render the symbol correctly, you can add Segoe UI Symbol as a fallback font.

Adding a Fallback Font in Windows (7 or Later)

To add a fallback font to the Oxygen XML Editor installation, use the following procedure:

1. Start Windows Explorer and browse to the \{OXYGEN_INSTALL_DIR\}/jre/lib/fonts directory.
2. Create a directory called fallback (if it is not already there).
3. Copy a font file (True Type Font - TTF) that includes the special characters into this directory.

   Tip:
   You could, for example, copy the Segoe UI Symbol Regular font from C:\Windows\Fonts.

4. Restart Oxygen XML Editor for the changes to take full effect.

Result: Whenever Oxygen XML Editor finds a character that cannot be rendered using its standard fonts, it will look for the glyph in the fonts stored in the fallback folder.

Adding a Fallback Font in Other Platforms

For macOS or other platforms, you could use the following approach:

1. Use a font editor (such as FontForge) to combine multiple true type fonts into a single custom font.
2. Install the font file into the dedicated font folder of your operating system.
3. In Oxygen XML Editor, open the Preferences dialog box (Options > Preferences) (on page 124), go to Appearance > Fonts.
4. Click the Choose button for the particular editing mode (Editor for Text mode) and select your custom font from the drop-down list in the subsequent dialog box.
5. Restart Oxygen XML Editor for the font changes to take full effect.

Related information

Unicode Support (on page 463)
Inserting Special Characters with the Character Map (on page 465)

Inserting Special Characters with the Character Map

Oxygen XML Editor includes a Character Map for inserting special characters. It can also be used to find the decimal, hexadecimal, or character entity equivalent for a particular character or symbol.
Inserting Special Characters

To insert a special character at the current location within a document, follow these steps:

1. Open the Character Map dialog box (on page 467) by selecting More symbols from the Symbols drop-down menu on the toolbar (if this button is not displayed, right-click in the toolbar area, select Configure Toolbars and chosen to display the Symbols toolbar (on page 365)).
2. Find the symbol you want to insert and double-click it (or select it and click Insert).

Tip:
The most recently used characters and some of the most common characters are listed when you click the Symbols drop-down button so you can easily insert any of those characters by simply selecting it from the drop-down.

Finding the Decimal, Hexadecimal, or Character Entity Equivalent

You can see the hexadecimal value for any character that is already inserted in your document by placing the cursor right after the character and you can see its value in the status bar at the bottom of the application.

For other characters, or to find the decimal equivalent, or even the character entity equivalent, following these steps:

1. Open the Character Map dialog box (on page 467) by selecting More symbols from the Symbols drop-down menu on the toolbar (if this button is not displayed, right-click in the toolbar area, select Configure Toolbars and chosen to display the Symbols toolbar (on page 365)).
2. Find the symbol and select it. You can use the filters and the Search field at the top of the dialog box to narrow the search.
3. Click the Details tab on top of the preview window to see the decimal, hexadecimal, and description of the character. The character entity equivalent (both its decimal and hexadecimal values) are displayed at the bottom of the dialog box.
Character Map Dialog Box

The Character Map dialog box allows you to visualize all characters that are available in a particular font, pick the character you need, and insert it in the document you are editing. It includes the following fields and sections:

Font

Use this drop-down list to choose the font that will have characters displayed.

Unicode Block

Use this drop-down list to only see a certain range of characters. This will filter the number of characters displayed, showing only a contiguous range of characters corresponding to the selected block. Unassigned characters are displayed as empty squares.

Search

Use this filter to search for a character by one of the following attributes:

- hexadecimal
- decimal
• description

Note:
Selecting description opens the Details tab (on page 468). If you enter a character description in the Search field, the description is selected automatically.

Character Table Section
The characters that are available to be inserted are listed in two tabs:

• **Compact** - Matrix-like table that displays a visual representation of the characters.
• **Details** - Displays the available characters in a tabular format, presenting their decimal and hexadecimal value along with their description.

Recently Used Characters Section
Displays the symbols that you have used recently and you can also select one from there to insert it in the current document.

Character Mode Section
The next section of the dialog box allows you to select how you want the character to appear in the Text editing mode. You can choose between the following:

• **Character**
• **Character entity - decimal**
• **Character entity - hexadecimal**

You can see the character or code that will be inserted in Text mode next to the selections in this section and a box on the right side of the dialog box allows you to see the character that will be inserted in Author mode. You can also see the name and range name of a character either at the bottom of the dialog box, or in a tooltip when hovering the cursor over the character.

Click the **Insert** button to insert the selected character in the current editor at the cursor position. You will see the character in the editor if the editor font (on page 133) is able to render it. The **Copy** button copies it to the clipboard without inserting it in the editor.

Note:
The Character Map dialog box cannot be used to insert Unicode characters in the Grid editor (on page 354). Accordingly, the Insert button of the dialog box will be disabled if the current document is edited in Grid mode.

Related information
Working with Special Characters and Encoding (on page 462)
Image Preview

Images and SVG files can be previewed in a separate pane. The supported image types are GIF, JPEG/JPG, PNG, BMP.

There are several ways to open an image in the Image Preview pane:

- In the Project view (on page 402), double-click the image name.
- In the Project view (on page 402), right-click an image and select Preview.
- In the DITA Maps Manager view (on page 2897), double-click the key definition of the image.
- In the DITA Maps Manager view (on page 2897), right-click the key definition of the image and select Open.
- In Text mode, Ctrl + Mouse Click or Ctrl + Enter with the cursor located within the image file path.

Once the image is displayed in the Image Preview pane, you have access to some contextual menu actions by right-clicking anywhere in the Image Preview pane. You can scale the image to its original size (by selecting the 1:1 action) or scale it down to fit in the pane (by selecting the Scale to fit action). Other actions include Open in System Application, Print preview, and Print.

If the image is an SVG file (on page 1251), the Image Preview pane also includes the following other contextual menu actions: Zoom in, Zoom out, Rotate, and Refresh.

While the Image Preview view is visible, selecting an image in the Project view (on page 402) or DITA Reusable Components view (on page 3061) will automatically display the resource in the view.

Tip:
You can drag an image from the Image Preview view and drop it in a DITA, DocBook, or TEI document.

Loading Large Documents

When you open a document with a file size larger than the limit configured in Open preferences (on page 200), Oxygen XML Editor prompts you to choose whether you want to optimize the loading of the document for large files or for huge files.
If your file has a size smaller than 300 MB, the recommended approach is **Optimize loading for large files** *(on page 470)*. For documents that exceed 300 MB, the recommended approach is **Optimize loading for huge files** *(on page 471)*.

### Optimize Loading for Large Files

If you open a document that exceeds the limit configured in **Open preferences** *(on page 200)* (the default limit is 30 MB), a dialog box will be displayed *(on page 469)* prompting you to choose whether you want to optimize the loading of the document for large files or for huge files. If you choose the **Optimize loading for large files** option (typically recommended for files smaller than 300 MB), a special memory optimization is implemented so that the total memory allocated for the application is not exceeded. A temporary buffer file is created on disk and the available free disk space needs to be at least double the size of the file you want to open.

When opening a large file in this optimized editing environment, some editing features are disabled, including:

- The file can only be opened in **Text** mode.
- The **automatic validation** *(on page 775)* is not available.
- The XPath filter is disabled in the **Find/Replace dialog box** *(on page 432)*.
- The bidirectional Unicode support (right-to-left writing) is disabled.
- The **Format and indent the document on open** option *(on page 202)* is automatically deselected for non-XML documents. For XML documents, the formatting is done while optimizing the memory usage by ignoring the options set in the **Format preferences page** *(on page 202)*.
- Localizations for the results of an **XPath expression** *(on page 2000)* will be less precise.
Optimize Loading for Huge Files

If you open a document that exceeds the limit configured in Open preferences (on page 200) (the default limit is 30 MB), a dialog box will be displayed (on page 469) prompting you to choose whether you want to optimize the loading of the document for large files or for huge files. If you choose the Optimize loading for huge files option (typically recommended for files larger than 300 MB), the file is split in multiple pages (each approximately 1MB in size). Each page is individually loaded (and edited) in Text mode by using a special horizontal slider located at the top of the editing area.

![Figure 85. Huge File Editor Horizontal Slider](image)

When opening a file in this special huge file editor, some editing features are disabled, including:

- For XML files, the UTF-8, UTF-16, ASCII, Windows-1252, and ISO 8859-1 encodings are supported. No other encoding is supported.
- The file can only be opened in Text editing mode.
- The automatic validation (on page 775) is disabled.
- The XPath filter is disabled in the Find/Replace dialog box (on page 432).
- The bidirectional Unicode support (right-to-left writing) is disabled.
- The Format and indent the document on open option (on page 202) is automatically deselected for non-XML documents. For XML documents, the formatting is done while optimizing the memory usage by ignoring the options set in the Format preferences page (on page 202).
- The Outline view (on page 539) is not supported.
- The file content is soft wrapped by default.
- The Find/Replace dialog box (on page 432) only supports the Find action.
- Saving changes is only possible if the Safe save option (on page 201) (in the Save preferences page) is enabled.
- The undo operation is not available if you go to other pages and come back to the modified page.

Related information
Optimize Loading for Large Files (on page 470)

Documents with Long Lines

When working with documents that contain lines of text that exceed the boundaries of your monitor, you might want to see the text wrapped. To do so, use one of the following methods:

- Press Ctrl + Shift + Y (Command + Shift + Y on macOS) to toggle the line wrap feature for the current document only.
- Select the Line wrap (on page 173) option in the Text preferences page to apply the line wrap to all documents.
Features that Might be Affected by Wrapping Lines of Text

Documents that contain thousands of characters per line can affect the performance of Oxygen XML Editor Text mode. When a certain line length limit is reached (controlled from the Optimize loading for documents with lines longer than (Characters) (on page 201) option), Oxygen XML Editor prompts you to wrap the lines of text. By doing so, the following features may be affected to maintain a reasonable level of productivity:

- The editor uses the **Monospaced** font.
- You cannot set font styles.
- **Automatic validation** (on page 775) is disabled.
- **Automatic spell checking** (on page 456) is disabled.
- When editing XML documents, the **XPath** field is disabled in the **Find/Replace** dialog box (on page 432).
- Less precise localization for executed XPath expressions in XML documents. The XPath executions use SAX sources for a smaller memory footprint. It is recommended to use XPath 2.0 instead of XPath 1.0 because it features an increased execution speed and uses a smaller memory footprint. Running an XPath expression requires additional memory of about 2 or 3 times the size of the document on disk.

Handling Read-Only Files

If a file marked as read-only is opened in Oxygen XML Editor you can by default perform modifications to it. This behavior is controlled by the **Can edit read only files** option (on page 171). When attempting to save such files you will be prompted to save them to another location.

You can check out the read-only state of the file by looking in the Properties view (on page 396). If you modify the file properties from the operating system and the file becomes writable, you can modify it on the spot without having to reopen it.

The read-only state is marked with a lock decoration that appears in the editor tab and specified in the tooltip for a certain tab.

Scratch Buffer

The **Scratch Buffer** view can be used for storing fragments of arbitrary text during the editing process. It can be used to drop bits of paragraphs (including arbitrary XML markup fragments) while rearranging and editing the document and also to drag and drop fragments of text from the Scratch Buffer to the editor panel. The **Scratch Buffer** is basically a text area offering XML syntax highlight. The view's contextual menu contains basic edit actions such as **Cut**, **Copy**, and **Paste**.

If the view is not displayed, it can be opened by selecting it from the Window > Show View menu.
**Compare Files or Directories**

Oxygen XML Editor provides a simple means of performing file and folder comparisons. You can see the differences in your files and folders and merge the changes. You can also use the file comparison to compare fragments or files inside zip-based archives.

There are two types of comparison tools: **Compare Directories** or **Compare Files**. These utilities are available from the **Tools** menu or can be opened as stand-alone applications from the Oxygen XML Editor installation folder (**diffDirs.exe** and **diffFiles.exe**).

**Starting the Tools from a Command Line**

The comparison tools can also be started by using command-line arguments. In the installation folder there are two executable shells (**diffFiles.bat** and **diffDirs.bat** on Windows, **diffFiles.sh** and **diffDirs.sh** on macOS and Linux). To specify files or directories to compare, you can pass command-line arguments to each of these shells. The arguments can point to file or folder paths in directories or archives (supported formats: **zip**, **docx**, and **xlsx**).

**Directory Comparison Example**

To start a **comparison between the two directories** *(on page 494)*, use the following construct:

```
diffDirs.bat/diffDirs.sh [directory path 1] [directory path 2]
```

If you pass only one argument, you are prompted to manually choose the second directory or archive.

For example, to start a comparison between two Windows directories, the command line would look like this:

```
diffDirs.bat "c:\documents new" "c:\documents old"
```

**Tip:** If there are spaces in the path names, surround the paths with quotes.

**File Comparison Example**

To start a **comparison between 2 or 3 files** *(on page 474)*, use the following construct: **diffFiles.bat**/**diffFiles.sh** *[path to left file]* *[path to right file]* *[path to base file]*.

If three files are specified, the tool will start in the **3-way comparison mode** *(on page 477)*. If only two files are specified, the tool will start in the **2-way comparison mode** *(on page 474)*. The first specified file will be added to the left panel in the comparison tool, the second file to the right panel, and the optional third file will be the base (ancestor) file used for a 3-way comparison. If you pass only one argument, you are prompted to manually choose another file.

For example, to do a 3-way comparison on Windows, the command line would look like this:

```
diffFiles.bat "c:\docs\file 1" "c:\docs\file 2" c:\docs\basefile
```
Tip:
If there are spaces in the path names, surround the paths with quotes.

Compare Files Tool

The built-in Compare Files tool can be used to compare files or XML file fragments. The tool provides a mechanism for comparing two files or fragments, as well as the mechanism for a three-way comparison. The utility is available from the Tools menu or can be opened as a stand-alone application from the Oxygen XML Editor installation folder (diffFiles.exe).

Figure 86. Compare Files Tool

Two-Way Comparisons

The Compare Files tool can be used to compare the differences between two files or XML fragments.

Compare Files

To perform a two-way comparison, follow these steps:
1. Open a file in the left panel and the file you want to compare it to in the right panel. You can specify the path by using the text field, the history drop-down, or the browsing actions in the ✗ Browse drop-down menu.

   **Step Result:** The selected files are opened in the two side-by-side editors. A text editing mode is used to offer a better view of the differences.

2. To highlight the differences between the two files, click the ✗ Perform File Differencing button from the toolbar.

3. You can use the drop-down menu on the left side of the toolbar to change the algorithm (on page 476) for the operation.

4. You can also use the Diff Options button to access the Files Comparison preferences page where you can choose to ignore certain types of markup and configure various options.

5. If you are comparing XML documents using the XML Fast or XML Accurate algorithms, you can enter an XPath 2.0 expression in the Ignore nodes by XPath text field to ignore certain nodes from the comparison.

The resulting comparison will show you differences between the two files. The line numbers on each side and colored marks on the right-side vertical stripe help you to quickly identify the locations of the differences. Adjacent changes are grouped into blocks of changes. This layout allows you to easily identify and focus on a group of related changes.

**Figure 87. Two-Way Differences**

| 1 | <?xml version="1.0" encoding="UTF-8"?>  | 1 | <?xml version="1.0" encoding="UTF-8"?> |
| 2 | <!DOCTYPE personnel PUBLIC "PERSONNEL" | 2 | <?xml-stylesheet type="text/css" href=" | 1 |
| 3 | <?xml-stylesheet type="text/css" href=" | 3 | personnel xmlns="http://www.oxygenxml | 2 |
| 4 | <personnel>                          | 4 | <person xmlns="http://www.oxygenxml | 3 |
|   |                                       |   | person id="robert.taylor" photo= |   |

**Highlighting Colors**

The differences are also highlighted in several colors, depending on the type of change, and dynamic lines connect the compared fragments in the middle section between the two panes. The highlighting colors can be customized in the Files Comparison / Appearance preferences page (on page 288), but the default colors and their shades mean the following:

- **Pink** - Identifies modifications on either side.
- **Gray** - Identifies an addition of a node in the left side (your outgoing changes).
- **Blue** - Identifies an addition of a node in the right side (incoming changes).
- **Lighter Shade** - Identifies blocks of changes that can be merged in their entirety.
- **Darker Shade** - Identifies specific changes within the blocks that can be merged more precisely.

**Comparing Fragments (Copy/Paste)**

To compare XML file fragments, you need to copy and paste the fragments you want to compare into each side, without selecting a file. If a file is already selected, you need to close it using the ✗ Close (Ctrl + W (Command + W on macOS)) button, before pasting the fragments. Other notes for pasting fragments:
• As long as the fragment is more than 10 characters, the application will attempt to automatically detect the content type. It can detect the following types: XML, DTD, CSS, JSON, and Markdown (if it starts with #). If one of those content types is detected, the fragments will be displayed with syntax highlights.
• If you save modified fragments, a dialog box opens that allows you to save the changes as a new document.

Navigate Differences

To navigate through differences, do one of the following:

• Use the navigation buttons on the toolbar (or in the Compare menu).
• Select a block of differences by clicking its small colored marker in the overview ruler located in the right-most part of the window. At the top of the overview ruler there is a success indicator that turns green where there are no differences, or red if differences are found.
• Click a colored area in between the two text editors.

Editing Actions

You can edit the files directly in either editing pane. The two editors are constantly synchronized and the differences are refreshed when you save the modified document or when you click the Perform File Differencing button.

A variety of actions are available on the toolbar (on page 486) and in the various menus (on page 490) (these same actions are also available in the contextual menu in both editing panes). The tool also includes some inline actions to help you merge, copy, or remove changes. When you select a change, the following inline action widgets are available, depending on the type of change:

- Append left change to right and Append right change to left
  Copies the content of the selected change from one side and appends it on the other, according to the content of the corresponding change. As a result, the side where the arrow points to will contain the changes from both sides.

- Copy change from left to right and Copy change from right to left
  Replaces the content of a change from one side with the content of the corresponding change from the other side.

- Remove change
  Rejects the change on the particular side and preserves the particular content on the other side.

Two-Way Diff Algorithms

Oxygen XML Editor offers the following two-way diff algorithms to compare files or fragments:
• **Auto** - Selects the most appropriate algorithm, based on the compared content and its size (selected by default).

• **Characters** - Computes the differences at character level, meaning that it compares two files or fragments looking for identical characters. This algorithm is not available when the file comparison is in **Author** comparison mode.

• **Words** - Computes the differences at word level, meaning that it compares two files or fragments looking for identical words. This algorithm is not available when the file comparison is in **Author** comparison mode.

• **Lines** - Computes the differences at line level, meaning that it compares two files or fragments looking for identical lines of text. This algorithm is not available when the file comparison is in **Author** comparison mode.

• **Syntax Aware** - Computes differences for known file types or fragments. This algorithm splits the files or fragments into sequences of **tokens** and computes the differences between them. The meaning of a **token** depends on the type of compared files or fragments.

Known file types include those listed in the **New** dialog box, such as XML file types (XSLT files, XSL-FO files, XSD files, RNG files, NVDL files, etc.), XQuery file types (**.xquery**, **.xq**, **.xqy**, **.xqm** extensions), DTD file types (**.dtd**, **.ent**, **.mod** extensions), TEXT file type (**.txt** extension), or PHP file type (**.php** extension).

For example:

- When comparing XML files or fragments, a token can be one of the following:
  - The name of an XML tag
  - The `<` character
  - The `/>` sequence of characters
  - The name of an attribute inside an XML tag
  - The `=` sign
  - The `"` character
  - An attribute value
  - The text string between the start tag and the end tag (a text node that is a child of the XML element corresponding to the XML tag that encloses the text string)

- When comparing plain text, a token can be any continuous sequence of characters or any continuous sequence of whitespaces, including a new line character.

• **XML Fast** - Comparison that works well on large files or fragments, but it is less precise than **XML Accurate**.

• **XML Accurate** - Comparison that is more precise than **XML Fast**, at the expense of speed. It compares two XML files or fragments looking for identical XML nodes.

### Three-Way Comparisons

Oxygen XML Editor also includes a three-way comparison feature to help you solve conflicts and merge changes between multiple modifications. It is especially helpful for teams who have multiple authors editing
and committing the same documents. It provides a comparison between a local change, another change, and the original base revision. Some additional advantages include:

- Visualize and merge content that was modified by you and another member of your team.
- Marks differences correctly even when the document structure is rearranged.
- Allows you to merge XML-relevant modifications.

**Figure 88. Three-Way Comparison**

![Three-Way Comparison Diagram](image)

**Compare Files**

To perform a three-way comparison, follow these steps:

1. Open a file in the left panel and the file you want to compare it to in the right panel. You can specify the path by using the text field, the history drop-down, or the browsing actions in the Browse drop-down menu.

   **Step Result:** The selected files are opened in the two side-by-side editors. A text editing mode is used to offer a better view of the differences.

2. Click the Three-Way Comparison button on the toolbar and select the base (original) file in the Base field. You can specify the path by using the text field, the history drop-down, or the browsing actions in the Browse drop-down menu.

3. To highlight the differences, click the Perform File Differencing button on the toolbar.

4. You can use the drop-down menu on the left side of the toolbar to change the algorithm (on page 476) for the operation.

5. You can also use the Diff Options button to access the Files Comparison preferences page where you can choose to ignore certain types of markup and configure various options.

The resulting comparison will show you differences between the two files, as well as differences between either of them and the base (original) file. The line numbers on each side and colored marks on the right-side vertical stripe help you to quickly identify the locations of the differences. Adjacent changes are grouped into blocks of changes.
Highlighting Colors

The differences are also highlighted in several colors, depending on the type of change, and dynamic lines connect the compared fragments in the middle section between the two panes. The highlighting colors can be customized in the Files Comparison / Appearance preferences page (on page 288), but the default colors and their shades mean the following:

- **Pink** - Identifies blocks of changes that include conflicts.
- **Gray** - Identifies your outgoing changes that do not include conflicts.
- **Blue** - Identifies incoming changes that do not include conflicts.
- **Lighter Shade** - Identifies blocks of changes that can be merged in their entirety.
- **Darker Shade** - Identifies specific changes within the blocks that can be merged more precisely.

Navigate Differences

To navigate through differences, do one of the following:

- Use the navigation buttons on the toolbar (or in the Compare menu).
- Select a block of differences by clicking its small colored marker in the overview ruler located in the right-most part of the window. At the top of the overview ruler there is a success indicator that turns green where there are no differences, or red if differences are found.
- Click a colored area in between the two text editors.

Editing Actions

You can edit the files directly in either editing pane. The two editors are constantly synchronized and the differences are refreshed when you save the modified document or when you click the Perform File Differencing button.

A variety of actions are available on the toolbar (on page 486) and in the various menus (on page 490) (these same actions are also available in the contextual menu in both editing panes). The tool also includes some inline actions to help you merge, copy, or remove changes. When you select a change, the following inline action widgets are available, depending on the type of change:

- **Append left change to right** and **Append right change to left**
  
  Copies the content of the selected change from one side and appends it on the other, according to the content of the corresponding change. As a result, the side where the arrow points to will contain the changes from both sides.

- **Copy change from left to right** and **Copy change from right to left**
Replaces the content of a change from one side with the content of the corresponding change from the other side.

Remove change

Rejects the change on the particular side and preserves the particular content on the other side.

Three-Way Diff Algorithms

Oxygen XML Editor offers the following three-way diff algorithms to compare files:

- **Auto** - Selects the most appropriate algorithm, based on the compared content and its size (selected by default).
- **Lines** - Computes the differences at line level, meaning that it compares two files or fragments looking for identical lines of text. This algorithm is not available when the file comparison is in Author comparison mode.
- **XML Fast** - Comparison that works well on large files or fragments, but it is less precise than **XML Accurate**.
- **XML Accurate** - Comparison that is more precise than **XML Fast**, at the expense of speed. It compares two XML files or fragments looking for identical XML nodes.

Second-Level Comparisons

For both two-way and three-way comparisons, Oxygen XML Editor automatically performs a second-level comparison for the **Lines**, **XML Fast**, and **XML Accurate** algorithms. After the first comparison is finished, the second-level comparison for the **Lines** algorithm is processed on text nodes using a word level comparison, meaning that it looks for identical words. For the **XML Fast** and **XML Accurate** algorithms, the second-level comparison is processed using a syntax-aware comparison, meaning that it looks for identical tokens. This second-level comparison makes it easier to spot precise differences and you can merge or reject the precise modifications.

![Second-Level Diff Comparison](image)

Note:

If a modified text fragment contains XML markup (such as processing instructions, XML comments, CDATA, or elements), the second-level comparison will not automatically be performed. In this case you can manually select a second-level comparison by doing a word level or character level comparison.

To do a word level comparison, select **Show word level details** from the contextual menu or **Compare** menu.
To do a character level comparison, select **Show Character Level details** from the contextual menu or ** Compare** menu.

**Author Visual Mode**

The **Compare Files** tool includes an **Author** comparison mode that displays the files in a visual mode similar to the **Author** editing mode in **Oxygen XML Editor/Author**. This makes it easier to see how the compared changes will look in the final output. This visual mode is available when the compared files are detected as being XML. To determine whether the files are initially opened in the merge tool's **Text** or **Author** comparison mode, it detects the **Initial Edit Mode** in the **Document Type Association** configuration (on page 141) and the mode the files were last opened in **Oxygen XML Editor/Author**.

**Note:**

This mode is not available if the **Enable file comparison in Author mode** option (on page 286) is not selected in the Diff > Files Comparison preferences page.

This visual mode includes unique features such as a **Tags Display Mode** drop-down button (on page 488) on the toolbar that allows you to select the amount of tags to display in this visual mode. This mode also presents differences that were made using the **Track Changes** feature (although the **Track Changes** feature is not available in the comparison tool).
Author Mode Algorithms

The visual Author comparison mode offers the following diff algorithms to compare files:

- **Auto** - Selects the most appropriate algorithm, based on the compared content and its size (selected by default).
- **XML Fast** - Comparison that works well on large files or fragments, but it is less precise than **XML Accurate**.
- **XML Accurate** - Comparison that is more precise than **XML Fast**, at the expense of speed. It compares two XML files or fragments looking for identical XML nodes.

Author Mode Second-Level Comparisons

The visual Author comparison mode automatically performs a second-level comparison for the **XML Fast** and **XML Accurate** algorithms. After the first comparison is finished, the second-level comparisons is processed on text nodes using a word-level comparison, meaning that it looks for identical words. This second-level comparison makes it easier to spot precise differences and you can merge or reject the precise modifications.
Starting File Comparison Tool from a Command Line

The file comparison tool can be started by using command-line arguments. In the installation folder there is an executable shell (diffFiles.bat on Windows, diffFiles.sh on macOS and Linux). To specify the files to compare, you can pass command-line arguments using the following construct: `diffFiles.bat/diffFiles.sh [path to left file] [path to right file] [path to 3-way base file].`

If three files are specified, the tool will start in the 3-way comparison mode (on page 477). If only two files are specified, the tool will start in the 2-way comparison mode (on page 474). The first specified file will be added to the left panel in the comparison tool, the second file to the right panel, and the optional third file will be the base (ancestor) file used for a 3-way comparison. If you pass only one argument, you are prompted to manually choose another file.

If you want to launch the file comparison tool from an external application with specified files and you want the file browsing buttons at the top of both panels to be hidden, you should use the `-ext` argument as the first command. There are some additional arguments that are allowed and to see all the details for the command-line construct, type `diffFiles.bat --help` in the command line.

**Example:**

To do a 3-way comparison, the command line might look like this:

**Windows**

```bash
diffFiles.bat "c:\docs\file 1" "c:\docs\file 2" c:\docs\basefile
```

**Tip:**

If there are spaces in the path names, surround the paths with quotes.

**Linux**

```bash
diffFiles.sh home/file1 home/file2 home/basefile
```

**macOS**

```bash
diffFiles.sh documents/file1 documents/file2 documents/basefile
```

How to Integrate the File Comparison Tool with Git

The file comparison tool can be integrated with Git clients. It requires that you configure your `.gitconfig` file and then you can simply start the tool from the command line.

To integrate the **Compare Files** tool with your Git client, follow this procedure:
1. Use one of the following methods to instruct your Git client to use the *Oxygen Compare Files* tool:

- **Manual Configuration** - Locate your Git user-specific configuration file (.gitconfig) and edit it with a text editor (for example, in Windows, the .gitconfig file is most likely located in your user home directory). Add (or replace) the following lines:

```plaintext
[diff]
  tool = oxygendiff

[merge]
  tool = oxygendiff

[difftool "oxygendiff"]
  cmd = '[pathToOxygenInstallDir]/diffFiles.exe' -ext $REMOTE $LOCAL $LOCAL

[mergetool "oxygendiff"]
  cmd = '[pathToOxygenInstallDir]/diffFiles.exe' -ext $LOCAL $REMOTE $BASE $MERGED
  trustExitCode = true

[difftool]
  prompt = false
```

**Note:**
For macOS, the cmd lines would start with something like: `sh "/Applications/Oxygen XML Editor/diffFiles.sh"`. For Linux, the cmd lines would start with something like: `sh "/Oxygen XML Editor/diffFiles.sh"`.

**Tip:**
On Redhat 7, the following command would work, where the whole command is quoted and then inside that, the path to `diffFiles.sh` is quoted:

```plaintext
[difftool "oxygendiff"]
  cmd = "'/home/user/Oxygen XML Editor 21/diffFiles.sh'" -ext $REMOTE $LOCAL $LOCAL

[mergetool "oxygendiff"]
  cmd = "'/home/user/Oxygen XML Editor 21/diffFiles.sh'" -ext $LOCAL $REMOTE $BASE $MERGED
  trustExitCode = true
```

- **Command Line Configuration** - To automatically configure the .gitconfig file, you can run the following commands from a command line:

```bash
git config --global diff.tool oxygendiff
git config --global difftool.oxygendiff.cmd '[Oxygen install dir]/diffFiles.exe -ext $REMOTE $LOCAL $LOCAL'

git config --global merge.tool oxygendiff
git config --global mergetool.oxygendiff.cmd '[Oxygen install dir]/diffFiles.exe'
```
Note:
For macOS, the Oxygen file comparison tool would be specified in the second and fourth commands with something like: `sh "/Applications/Oxygen XML Editor/diffFiles.sh"`. For Linux, it would be something like: `sh "/Oxygen XML Editor/diffFiles.sh"`.

2. To start the Compare Files tool and see a comparison of changes for a particular file, run the following command from a command line:

```
git difftool [PathToFile]
```

Tip:
If the file you want to compare has conflicts, you can start the Compare Files tool as a merge conflict resolution tool by running the following command:

```
git mergetool [PathToFile]
```

For more information about the Git `difftool` syntax, see [https://git-scm.com/docs/git-difftool](https://git-scm.com/docs/git-difftool).

For more information about the Git `mergetool` syntax, see [https://git-scm.com/docs/git-mergetool](https://git-scm.com/docs/git-mergetool).

### How to Integrate the File Comparison Tool with Sourcetree

The file comparison tool can be integrated with Sourcetree so that you can use it to compare changes. The advantages of doing this include:

- The **Oxygen Compare Files** tool presents the files side-by-side and makes it much easier to determine real changes.
- The **Oxygen Compare Files** tool includes XML comparison algorithms.
- The **Oxygen Compare Files** tool includes various options for configuring the comparison.
- The **Oxygen Compare Files** tool allows you to navigate through changes.

To integrate the Compare Files tool with Sourcetree, follow this procedure, depending on your operating system:

#### Windows

1. In Sourcetree, go to **Tools > Options**.
2. Go to the **Diff** tab.
3. In the **External Diff/Merge** section, configure the settings as follows:
   - **External Diff Tool** - Select **Custom**.
   - **Diff Command** - Enter the path of the **Oxygen** `diffFile.exe` file (for example: `c:\Programs\Oxygen XML Editor 19\diffFiles.exe`).
- Arguments - Enter -ext $REMOTE $LOCAL $LOCAL.
- Merge Tool - Select Custom.
- Diff Command - Enter the path of the Oxygen diffFile.exe file (for example: \Programs\Oxygen XML Editor 19\diffFiles.exe).
- Arguments - Enter -ext $LOCAL $REMOTE $BASE $MERGED.

4. Click OK.

**Result:** In Sourcetree, you can now compare file changes with the Oxygen Compare Files tool by simply selecting External Diff from the contextual menu, Actions menu, or Ctrl+D.

### macOS

1. In Sourcetree, go to Sourcetree > Preferences.
2. Go to the Diff tab.
3. In the External Diff/Merge section, configure the settings as follows:
   - External Diff Tool - Select Custom.
   - Diff Command - Enter a command-line argument to launch the Oxygen diffFiles.sh file (for example: sh "/Applications/Oxygen XML Editor/diffFiles.sh").
   - Arguments - Enter -ext $REMOTE $LOCAL $LOCAL.
   - Merge Tool - Select Custom.
   - Diff Command - Enter a command-line argument to launch the Oxygen diffFiles.sh file (for example: sh "/Applications/Oxygen XML Editor/diffFiles.sh").
   - Arguments - Enter -ext $LOCAL $REMOTE $BASE $MERGED.
4. Close the preferences dialog box.

**Result:** In Sourcetree, you can now compare file changes with the Oxygen Compare Files tool by simply selecting External Diff from the contextual menu or Actions menu.

### Toolbar and Contextual Menu Actions of the Compare Files Tool

The toolbar of the Compare Files tool contains operations that can be performed on the source and target files or XML fragments. Many of the actions are also available in the contextual menu.

**Figure 94. Compare Toolbar**

The following actions are available:

**Algorithm**

This drop-down menu allows you to select one of the following diff algorithms (depending on whether it is a two-way or three-way comparison):
• **Auto** - Selects the most appropriate algorithm, based on the compared content and its size (selected by default).

• **Characters** - Computes the differences at character level, meaning that it compares two files or fragments looking for identical characters. This algorithm is not available when the file comparison is in **Author** comparison mode.

• **Words** - Computes the differences at word level, meaning that it compares two files or fragments looking for identical words. This algorithm is not available when the file comparison is in **Author** comparison mode.

• **Lines** - Computes the differences at line level, meaning that it compares two files or fragments looking for identical lines of text. This algorithm is not available when the file comparison is in **Author** comparison mode.

• **Syntax Aware** - Computes differences for the file types or fragments known by Oxygen XML Editor, taking the syntax (the specific types of tokens) into consideration. This algorithm is not available when the file comparison is in **Author** comparison mode.

• **XML Fast** - Comparison that works well on large files or fragments, but it is less precise than **XML Accurate**.

• **XML Accurate** - Comparison that is more precise than **XML Fast**, at the expense of speed. It compares two XML files or fragments looking for identical XML nodes.

**Diff Options**

Opens the **Files Comparison preferences page (on page 286)** where you can configure various options.

**Three-Way Comparison**

Toggle action that allows you to perform a three-way comparison between the two files displayed in the two editing panes and a base (ancestor) file.

**Perform Files Differencing**

Looks for differences between the two files displayed in the left and right side panels.

**Synchronized scrolling**

Toggles synchronized scrolling on or off so that a selected difference can be seen on both sides of the application window. This option is on by default.

**Ignore Whitespaces**

Enables or disables the whitespace ignoring feature. Ignoring whitespace means that before performing the comparison, the application normalizes the content and trims its leading and trailing whitespaces. This option is not available when in the **Author** comparison mode.

**Format and Indent Both Files (Ctrl + Shift + P (Command + Shift + P on macOS))**
Formats and indents both files before comparing them. Use this option for comparisons that contain long lines that make it difficult to spot differences. This option is not available when in the Author comparison mode.

**Note:**
When comparing two JSON files, the Format and Indent Both Files action will automatically sort the keys in both files the same to make it easier to compare.

**Tags Display Mode**
Allows you to select the amount of markup to be displayed in the Author visual comparison mode (on page 481). You can choose between: Full Tags with Attributes, Full Tags, Block Tags, Block Tags without Element Names, Inline Tags, Partial Tags, or No Tags.

**Copy Change from Right to Left**
Copies the selected difference from the file in the right panel to the file in the left panel.

**Copy All Changes from Right to Left**
Copies all changes from the file in the right panel to the file in the left panel.

**Next Block of Changes (Ctrl + Period (Command + Period on macOS))**
Jumps to the next block of changes. This action is not available when the cursor is positioned on the last change block or when there are no changes.

**Previous Block of Changes (Ctrl + Comma (Command + Comma on macOS))**
Jumps to the previous block of changes. This action is not available when the cursor is positioned on the first change block or when there are no changes.

**Next Change (Ctrl + Shift + Period (Command + Shift + Period on macOS))**
Jumps to the next change from the current block of changes. When the last change from the current block of changes is reached, it highlights the next block of changes. This action is not available when the cursor is positioned on the last change or when there are no changes.

**Previous Change (Ctrl + Shift + Comma (Command + Shift + M on macOS))**
Jumps to the previous change from the current block of changes. When the first change from the current block of changes is reached, it highlights the previous block of changes. This action is not available when the cursor is positioned on the first change or when there are no changes.

**Copy All Changes from Left to Right**
Copies all changes from the file in the left panel to the file in the right panel.

**Copy Change from Left to Right**

Copies the selected difference from the file in the left panel to the file in the right panel.

**Ignore Nodes by XPath**

You can use this text field to enter an XPath expression (on page 1999) to ignore certain nodes from the comparison. It will be processed as XPath version 2.0. You can also enter the name of the node to ignore all nodes with the specified name (for example, if you want to ignore all ID attributes from the document, you could simply enter @id). This field is only available when comparing XML documents using the XML Fast or XML Accurate algorithms.

**Note:**

If an XPath expression is specified in the Ignore nodes by XPath option (on page 288) in the Diff / File Comparison preferences page, that one is used as a default when the application is started. If you then enter an expression in this field on the toolbar, this one will be used instead of the default. If you delete the expression from this field, neither will be used.

**First Change (Ctrl + B (Command + B on macOS))**

Jumps to the first change.

**Base**

Available for three-way comparisons (on page 477). It is the base file that will be compared with the files opened in the left and right editors. You can specify the path to the file by using the text field, its history drop-down, or the browsing actions in the Browse drop-down menu.

**Left-Side (Source) File**

You can specify the path to the file to be compared on the left side (source) by using the text field, its history drop-down, or the browsing actions in the Browse drop-down menu.

**Save**

Saves the changes made in the source (left-side) file.

**Reload**

Reloads the source (left-side) file.

**Close**

Closes the source (left-side) file.

**Right-Side (Target) File**

You can specify the path to the file to be compared on the right side (target) by using the text field, its history drop-down, or the browsing actions in the Browse drop-down menu.
Save
Saves the target (right-side) file.

Reload
Reloads the target (right-side) file.

Close
Closes the target (right-side) file.

Compare Files Tool Menus
The menus in the Compare Files tool contain some of the same actions that are on the toolbar, as well as some common actions that are identical to the same actions in the Oxygen XML Editor menus. The menu actions include:

File Menu

Source > Open
Browses for a file that will be displayed in the left panel.

Source > Open URL
Browses for a remote file that will be displayed in the left panel.

Source > Open File from Archive
Browses an archive for a file that will be displayed in the left panel.

Source > Reload
Reloads the file in the left panel.

Source > Save
Saves the changes made to the file in the left panel.

Source > Save As
Allows you to choose a destination to save the file in the left panel.

Source > Close
Closes the file in the left panel.

Target > Open
Browses for a file that will be displayed in the right panel.

Target > Open URL
Browses for a remote file that will be displayed in the right panel.

Target > Open File from Archive
Browses an archive for a file that will be displayed in the right panel.
Target > Reload

Reloads the file in the right panel.

Target > Save

Saves the changes made to the file in the right panel.

Target > Save As

Allows you to choose a destination to save the file in the right panel.

Target > Close

Closes the file in the right panel.

Base > Open

Browses for a file that will be compared with both files in a three-way comparison (on page 477).

Base > Open URL

Browses for a remote file that will be compared with both files in a three-way comparison (on page 477).

Base > Open File from Archive

Browses an archive for a file that will be compared with both files in a three-way comparison (on page 477).

Save Results as HTML (Available in Text mode only)

Generates an HTML file that contains detailed information about the comparison result. See an example of what the generated report look like in the Generate HTML Report for Directory Comparison topic (on page 511).

Close (Ctrl + W (Command + W on macOS))

Closes the application.

Edit Menu

Cut

Cut the selection from the currently focused editor panel to the clipboard.

Copy

Copy the selection from the currently focused editor panel to the clipboard.

Paste

Paste content from the clipboard into the currently focused editor panel.

Select all

Selects all content in the currently focused editor panel.
Undo

Undo changes in the currently focused editor panel.

Redo

Redo changes in the currently focused editor panel.

Find Menu

Find/Replace

Perform find/replace operations in the currently focused editor panel.

Find Next

Go to the next match using the same options as the last find operation. This action runs in both editor panels.

Find Previous

Go to the previous match using the same options as the last find operation. This action runs in both editor panels.

Compare Menu

Three-Way Comparison

Toggle action that allows you to perform a three-way comparison between the two files displayed in the two editing panes and a base (ancestor) file.

Perform Files Differencing

Looks for differences between the two files displayed in the left and right side panels.

Next Block of Changes (Ctrl + Period (Command + Period on macOS))

Jumps to the next block of changes. This action is not available when the cursor is positioned on the last change block or when there are no changes.

Previous Block of Changes (Ctrl + Comma (Command + Comma on macOS))

Jumps to the previous block of changes. This action is not available when the cursor is positioned on the first change block or when there are no changes.

Next Change (Ctrl + Shift + Period (Command + Shift + Period on macOS))

Jumps to the next change from the current block of changes. When the last change from the current block of changes is reached, it highlights the next block of changes. This action is not available when the cursor is positioned on the last change or when there are no changes.

Previous Change (Ctrl + Shift + Comma (Command + Shift + M on macOS))
Jumps to the previous change from the current block of changes. When the first change from the current block of changes is reached, it highlights the previous block of changes. This action is not available when the cursor is positioned on the first change or when there are no changes.

**Last Change** *(Ctrl + E (Command + E on macOS))*

Jumps to the last change.

**First Change** *(Ctrl + B (Command + B on macOS))*

Jumps to the first change.

**Copy All Changes from Right to Left**

Copies all changes from the file in the right panel to the file in the left panel.

**Copy All Changes from Left to Right**

Copies all changes from the file in the left panel to the file in the right panel.

**Copy Change from Right to Left**

Copies the selected difference from the file in the right panel to the file in the left panel.

**Copy Change from Left to Right**

Copies the selected difference from the file in the left panel to the file in the right panel.

**Show Word Level Details**

Provides a word-level comparison of the selected change.

**Show Character Level Details**

Provides a character-level comparison of the selected change.

**Format and Indent Both Files** *(Ctrl + Shift + P (Command + Shift + P on macOS))*

Formats and indents both files before comparing them. Use this option for comparisons that contain long lines that make it difficult to spot differences.

**Note:**

When comparing two JSON files, the **Format and Indent Both Files** action will automatically sort the keys in both files the same to make it easier to compare.

**Options Menu**

**Preferences**

Opens the preferences dialog box that includes numerous pages of options that can be configured.

**Menu Shortcut Keys**

Opens the **Menu Shortcut Keys** option page where you can configure keyboard shortcuts available for menu items.
Reset Global Options

Resets options to their default values. Note that this option appears only when the tool is executed as a stand-alone application.

Import Global Options

Allows you to import an options set that you have previously exported.

Export Global Options

Allows you to export the current options set to a file.

Help Menu

Help (F1)

Opens a Help dialog box that displays the User Manual at a section that is appropriate for the context of the current cursor position.

Use Online Help

If this option is selected, when you select Help or press F1 while hovering over any part of the interface, Oxygen XML Editor attempts to open the help documentation in online mode. If this option is not selected or an internet connection fails, the help documentation is opened in offline mode.

Report problem

Opens a dialog box that allows the user to write the description of a problem that was encountered while using the application. You can change the URL where the reported problem is sent by using the com.oxygenxml.report.problems.url system property. The report is sent in XML format through the report parameter of the POST HTTP method.

Support Center

Opens the Oxygen XML Editor Support Center web page in a browser.

Compare Directories Tool

The Compare Directories tool can be used to compare and manage changes to files and folders within the structure of your directories. The utility is available from the Tools menu or can be opened as a stand-alone application from the Oxygen XML Editor installation folder (diffDirs.exe).
Starting the Tool from a Command Line

The directory comparison tool can also be started by using command-line arguments. In the installation folder there is an executable shell (\texttt{diffDirs.bat} on Windows, \texttt{diffDirs.sh} on macOS and Linux). To specify the directories to compare, you can pass command-line arguments using the following construct:

\texttt{diffDirs.bat/diffDirs.sh \{directory path 1\} \{directory path 2\}}.

If you pass only one argument, you are prompted to manually choose the second directory or archive.

Example:

To do a comparison between two directories, the command line would look like this:

\textbf{Windows}

\begin{verbatim}
diffDirs.bat "c:\documents new" "c:\documents old"
\end{verbatim}

\textbf{Tip:}

If there are spaces in the path names, surround the paths with quotes.

\textbf{Linux}

\begin{verbatim}
diffDirs.sh home/documents1 home/documents2
\end{verbatim}

\textbf{macOS}

\begin{verbatim}
diffDirs.sh documents1 documents2
\end{verbatim}
**Directory Comparisons**

To perform a directory comparison, follow these steps:

1. Select a folder in the left panel and the folder you want to compare it to in the right panel. You can specify the path by using the text field, the history drop-down, or the **Browse for local directory** action in the **Browse** drop-down menu.

   **Step Result:** The selected directory structures are opened in the two side-by-side panels.

2. To highlight the differences between the two folders, click the **Perform Directories Differencing** button from the toolbar.

3. You can also use the **Diff Options** button to access the **Directories Comparison preferences page** (on page 289) where you can configure various options.

To compare the content of two archives, follow these steps:

1. Use the **Browse for archive file** action in the **Browse** drop-down menu to select the archives in the left and right panels.

2. By default, the supported archives are not treated as directories and the comparison is not performed on the files inside them. To make Oxygen XML Editor treat supported archives as directories, select the **Look in archives** option (on page 290) in the **Directories Comparison** preferences page.

3. To highlight the differences, click the **Perform Directories Differencing** button from the toolbar.

The directory comparison results are presented using two tree-like structures showing the files and folders, including their name, size, and modification date. A column that contains graphic symbols separates the two tree-like structures. The graphic symbols can be one of the following:

- An `X` symbol, when a file or a folder exists in only one of the compared directories.
- A `≠` symbol, when a file exists in both directories but the content differs. The same sign appears when a collapsed folder contains differing files.

The color used for the symbol and the directory or file name can be customized in the **Directories Comparison / Appearance** preferences page (on page 290). You can double-click lines marked with the `≠` symbol to open a **Compare Files** window, which shows the differences between the two files.

The directories that contain files that differ are expanded automatically so that you can focus directly on the differences. You can merge the contents of the directories by using the copy actions. If you double-click (or press **Enter**) on a line with a pair of files, Oxygen XML Editor starts a **file comparison** (on page 474) between the two files, using the **Compare Files** tool.

---

**Related information**

- **Compare Files Tool** (on page 474)
- **Compare Directories Script** (on page 3206)
Toolbar and Contextual Menu Actions of the Compare Directories Tool

The toolbar of the Compare Directories tool contains operations that can be performed on the compared directory structure. Some of the toolbar actions are also available in the contextual menu.

**Figure 96. Compare toolbar**

<table>
<thead>
<tr>
<th>Action</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1" alt="Perform Directories Differencing" /></td>
<td>Looks for differences between the two directories displayed in the left and right side of the application window.</td>
</tr>
<tr>
<td><img src="image2" alt="Perform Files Differencing" /></td>
<td>Opens the Compare Files tool (on page 474) that allows you to compare the currently selected files.</td>
</tr>
<tr>
<td><img src="image3" alt="Copy Change from Right to Left" /></td>
<td>Copies the selected change from the right side to the left side (if there is no file/folder in the right side, the left file/folder is deleted).</td>
</tr>
<tr>
<td><img src="image4" alt="Copy Change from Left to Right" /></td>
<td>Copies the selected change from the left side to the right side (if there is no file/folder in the left side, the right file/folder is deleted).</td>
</tr>
<tr>
<td><img src="image5" alt="Binary Compare" /></td>
<td>Performs a byte-level comparison on the selected files.</td>
</tr>
<tr>
<td><img src="image6" alt="Diff Options" /></td>
<td>Opens the Directory Comparison preferences page (on page 289) where you can configure various options.</td>
</tr>
<tr>
<td><img src="image7" alt="Show Only Modifications" /></td>
<td>Displays a more uncluttered file structure by hiding all identical files.</td>
</tr>
<tr>
<td><img src="image8" alt="Save Results as HTML" /></td>
<td>Generates an HTML file that contains detailed information about the comparison result.</td>
</tr>
</tbody>
</table>

**File and folder filters**

Differences can be filtered using three combo boxes: Include files, Exclude files, and Exclude folders. They come with predefined values and are editable to allow custom values. All of them accept multiple comma-separated values and the * and ? wildcards. For example, to filter out...
all JPEG and GIF image files, edit the Exclude files filter box to read *.jpeg, *.png. Each filter includes a drop-down menu with the latest 15 filters applied.

Contextual Menu Actions

⚠️ Perform Files Differencing

Opens the Compare Files tool (on page 474) that allows you to compare the currently selected files.

(enable)

Binary Compare

Performs a byte-level comparison on the selected files.

Copy Change from Right to Left

Copies the selected difference from the file in the right panel to the file in the left panel.

Copy Change from Left to Right

Copies the selected difference from the file in the left panel to the file in the right panel.

Open

If the action is invoked on a file, the selected file is opened in Oxygen XML Editor. If the action is invoked on a directory, the selected directory is opened in the default file browser for your particular operating system.

Open in System Application

Opens the selected file in the system application that is associated with that type of file. The action is available when launching the Compare Directories tool from the Tools menu in Oxygen XML Editor.

Show in Explorer

Opens the default file browser for your particular operating system with the selected file highlighted.

Compare Directories Tool Menus

The menus in the Compare Directories tool contain some of the same actions that are on the toolbar, as well as some common actions that are identical to the same actions in the Oxygen XML Editor menus. The menu actions include:

File Menu

Save Results as HTML

Generates an HTML file that contains detailed information about the comparison result. See an example of what the generated report look like in the Generate HTML Report for Directory Comparison topic (on page 511).

Close (Ctrl + W (Command + W on macOS))
Closes the application.

**Compare Menu**

**Perform Directories Differencing**
Looks for differences between the two directories displayed in the left and right side of the application window.

**Perform Files Differencing**
Opens the Compare Files tool *(on page 474)* that allows you to compare the currently selected files.

**Copy Change from Right to Left**
Copies the selected change from the right side to the left side (if there is no file/folder in the right side, the left file/folder is deleted).

**Copy Change from Left to Right**
Copies the selected change from the left side to the right side (if there is no file/folder in the left side, the right file/folder is deleted).

**Options Menu**

**Preferences**
Opens the preferences dialog box that includes numerous pages of options that can be configured.

**Menu Shortcut Keys**
Opens the Menu Shortcut Keys option page where you can configure keyboard shortcuts available for menu items.

**Reset Global Options**
Resets options to their default values. Note that this option appears only when the tool is executed as a stand-alone application.

**Import Global Options**
Allows you to import an options set that you have previously exported.

**Export Global Options**
Allows you to export the current options set to a file.

**Help Menu**

**Help (F1)**
Opens a Help dialog box that displays the User Manual at a section that is appropriate for the context of the current cursor position.

**Use Online Help**
If this option is selected, when you select Help or press F1 while hovering over any part of the interface, Oxygen XML Editor attempts to open the help documentation in online mode. If this option is not selected or an internet connection fails, the help documentation is opened in offline mode.

**Report problem**

Opens a dialog box that allows the user to write the description of a problem that was encountered while using the application. You can change the URL where the reported problem is sent by using the `com.oxygenxml.report.problems.url` system property. The report is sent in XML format through the `report` parameter of the POST HTTP method.

**Support Center**

Opens the Oxygen XML Editor Support Center web page in a browser.

### Compare Images

You can use the **Compare Directories** tool to compare images. If you double-click a line that contains two different images, the **Compare images** window is displayed. This dialog box presents the images in the left and right sides, scaled to fit the available view area. You can use the contextual menu actions to scale the images to their original size or scale them down to fit in the view area.

The supported image types are: GIF, JPG, JPEG, PNG, and BMP.

### Compare Directories Against a Base (3-Way) Tool

The **Compare Directories Against a Base (3-way)** tool allows you to perform three-way comparisons on directories to help you identify and merge changes between multiple modifications of the same directory structure. It is especially helpful for teams that have multiple authors contributing documents to the same directory system. It offers information about conflicts and changes, and includes actions to easily merge, accept, overwrite, or ignore changes to the directory system.

#### How to Perform 3-Way Directory Comparisons

To perform a 3-way directories comparison, follow these steps:

1. Select **Compare Directories Against a Base (3-way)** from the **Tools** menu.

   **Step Result:** This opens a dialog box that allows you to select the 3 file sets that will be used for the comparison.
2. Select the file sets to be compared:
   - **Base directory** - This is the original (base) file set before any modifications were made by you or others.
   - **Directory with your changes** - This is the file set with changes that you have made. This file set will be displayed in the left panel in the comparison tool.
   - **Directory with changes made by others** - This is the file set with changes made by others that you want to merge with your changes. This file set will be displayed in the right panel in the comparison tool.

3. Click the **Compare** button to compare the file sets and open the comparison and merge tool.

4. Use the features and actions described in the next section to identify and merge the changes.
3-Way Directory Comparison and Merge Tool

Figure 98. Comparison and Merge Tool

The 3-way directory comparison and merge tool includes the following information, features, and actions:

**Number of Changes and Conflicts**

The first thing you see in the top-left corner of the tool is the grand total of all the changes made by others, changes made by you, and the number of conflicts.

**Filter Buttons**

In the top-right corner you can use the toggle buttons to filter the list of modifications:

- **Show all files**
  Use this button to show all modified and unmodified files, as well as conflicts.

- **Show only files modified by you and others**
  Filters the list to show all files that have been modified, including conflicts.

- **Show only files modified by others**
  Filters the list to only show the files that were modified by others.

- **Show only files modified by you**
  Filters the list to only show the files that were modified by you.
**Show only conflicting files**

Filters the list to only show files that contain conflicts.

**List of Files Panel**

This panel shows the list of files in the compared file sets based upon the filter button that is selected. This panel includes the following sortable columns:

- **Name** - The file names.
- **Status** - An icon that represents the file status. Red icons indicate some sort of conflict. Gray icons indicate modifications made by you. Blue icons indicate modifications made by others.
- **Description** - A description of the file status.
- **Merge Action** - This column provides a drop-down menu for each file that allows you to choose some merge actions depending upon its status. A default action is always set to *Automatically merge* the changes made by others with your changes. If there is a conflict, the default is `<Select action>` and you are required to make a selection. Click this column to access the drop-down menu where you can make a selection. The same actions are available in the contextual menu.

**Tip:**

If the solution proposed in the **Merge Action** column for any particular file is not satisfactory, you can change it directly in that column (even if that file is not selected) without automatically re-triggering the comparison (except for in certain cases where re-triggering the comparison is necessary).

You can click a file to open it in the file comparison panel (the file from your file set is shown in the left panel while the file from the file set with changes made by others is shown in the right panel). For image files, the comparison panel shows a preview of the image. For other binary files, a preview is not available and you will just see its status.

**File Comparison Panels**

If you click a file in the top panel, the file is opened in this file comparison section. The file from your file set is shown in the left panel and the file from the other file set is shown in the right panel.

**Note:**

If Oxygen XML Editor does not recognize the file type, a dialog box will be displayed that allows you to select the type of editor you want it to be associated with for this comparison (if you want Oxygen XML Editor to remember this association, you can select the **Associate file type with editor** option at the bottom of the dialog box).

This panel includes the following information and toolbar actions:
File Path

The first thing you see in this panel is the file path where merge actions will be applied if you make changes.

× Close

Closes the file comparison panel.

Algorithm Drop-down Menu

This drop-down menu allows you to select one of the following diff algorithms to be used for file comparisons:

- **Auto** - Selects the most appropriate algorithm, based on the compared content and its size (selected by default).
- **Lines** - Computes the differences at line level, meaning that it compares two files or fragments looking for identical lines of text. This algorithm is not available when the file comparison is in Author comparison mode.
- **XML Fast** - Comparison that works well on large files or fragments, but it is less precise than XML Accurate.
- **XML Accurate** - Comparison that is more precise than XML Fast, at the expense of speed. It compares two XML files or fragments looking for identical XML nodes.

Diff Options

Opens the Files Comparison preferences page (on page 286) where you can configure various options.

Perform Files Differencing

Looks for differences between the two files displayed in the left and right side panels.

Synchronized scrolling

Toggles synchronized scrolling. When toggled on, a selected difference can be seen in both panels.

Ignore Whitespaces

Enables or disables the whitespace ignoring feature. Ignoring whitespace means that before performing the comparison, the application normalizes the content and trims its leading and trailing whitespaces. This option is not available when the file comparison is in Author mode.

Tags Display Mode

Allows you to select the amount of markup to be displayed in the Author visual mode. You can choose between: Full Tags with Attributes, Full Tags,
Block Tags, Block Tags without Element Names, Inline Tags, Partial Tags, or No Tags.

Copy Change from Right to Left
Copies the selected difference from the file in the right panel to the file in the left panel.

Copy All Changes from Right to Left
Copies all changes from the file in the right panel to the file in the left panel.

Next Block of Changes (Ctrl + Period (Command + Period on macOS))
Jumps to the next block of changes. This action is not available when the cursor is positioned on the last change block or when there are no changes.

Note:
A change block groups one or more consecutive lines that contain at least one change.

Previous Block of Changes (Ctrl + Comma (Command + Comma on macOS))
Jumps to the previous block of changes. This action is not available when the cursor is positioned on the first change block or when there are no changes.

Next Change (Ctrl + Shift + Period (Command + Shift + Period on macOS))
Jumps to the next change from the current block of changes. When the last change from the current block of changes is reached, it highlights the next block of changes. This action is not available when the cursor is positioned on the last change or when there are no changes.

Previous Change (Ctrl + Shift + Comma (Command + Shift + M on macOS))
Jumps to the previous change from the current block of changes. When the first change from the current block of changes is reached, it highlights the previous block of changes. This action is not available when the cursor is positioned on the first change or when there are no changes.

First Change (Ctrl + B (Command + B on macOS))
Jumps to the first change.

Left-Side File (Your changes)
Above the panel you can see the file path and the following two buttons:

Save
Saves changes made to the file.

Reload
Reloads the file.

Right-Side File (Changes made by others)

Above the panel you can see the file path and the following two buttons:

Reload

Reloads the file.

Displaying Changes in the File Comparison Panels

The line numbers on each side and colored marks on the right-side vertical stripe help you to quickly identify the locations of the differences. Adjacent changes are grouped into blocks of changes.

Figure 99. File Comparison Panels

The differences are also highlighted in several colors, depending on the type of change, and dynamic lines connect the compared fragments in the middle section between the two panes. The highlighting colors can be customized in the Files Comparison / Appearance preferences page (on page 288), but the default colors and their shades mean the following:

- **Pink** - Identifies modifications on either side.
- **Gray** - Identifies an addition of a node in the left side (your outgoing changes).
- **Blue** - Identifies an addition of a node in the right side (incoming changes).
- **Lighter Shade** - Identifies blocks of changes that can be merged in their entirety.
- **Darker Shade** - Identifies specific changes within the blocks that can be merged more precisely.

Direct Editing Actions in the File Comparison Panels

In addition to selecting merge actions from the drop-down menus in the Merge Action column in the top panel, you can also edit the files directly in the left pane (your local changes). The two editors are constantly synchronized and the differences are refreshed when you save the modified document (Save button or Ctrl+S) or when you click the Perform File Differencing button.

A variety of actions are available in the contextual menu in both editing panes. The tool also includes some inline actions to help you merge, copy, or remove changes. When you select a change, the following inline action widgets are available, depending on the type of change:

Append right change to left
Copies the content of the selected change from the right side and appends it on the left side.

- **Copy change from right to left**
  Replaces the content of a change in the left side with the content of the change in the right side.

- **Remove change**
  Removes the change from the left side.

Anytime you save manual changes (Save button or Ctrl+S), the selection in the **Merge Action** column in the top panel automatically changes to **Use merged** and a copy of the original file is kept so that you can revert to the original file if necessary. To discard your manual changes and revert to your original changes, select a different action in the **Merge Action** drop-down menu.

**Open Merged Files**
If you select this option, all the files that will be modified by the merge operation will be opened in the editor after the operation is finished.

**Applying Changes**
When you click the **Apply** button, all the merge actions you have selected and the changes you have made will be processed.

If there are unresolved conflicts (conflicts where no merge action is selected in the **Merge Action** drop-down menu), a dialog box will be displayed that allows you to choose how to solve the conflicts. You can choose between the following:

- **Keep your changes** - If you select this option and then click **Apply**, your local changes will be preserved for the unresolved conflicts.
- **Overwrite your changes** - If you select this option and then click **Apply**, your local changes will be overwritten with the changes made by others, for the unresolved conflicts.
- **Cancel** - You can click the **Cancel** button to go back to the merge tool to resolve the conflicts individually.

**Canceling Changes**
If you click the **Cancel** button at the bottom of the merge tool, all the changes you made in the tool will be lost.

**Author Visual Mode**
The **Comparison and Merge** tool includes an **Author** mode that displays the files in a visual mode similar to the **Author** editing mode in Oxygen XML Editor/Author. This makes it easier to see how the compared changes will look in the final output. This visual mode is available when the compared files are detected as being XML. To determine whether the files are initially opened in the merge tool's **Text** or **Author** mode, it
detects the Initial Edit Mode in the Document Type Association configuration (on page 141) and the mode the files were last opened in Oxygen XML Editor/Author.

**Note:**
This mode is not available if the Enable file comparison in Author mode option (on page 286) is not selected in the Diff > Files Comparison preferences page.

This visual mode includes unique features such as a Tags Display Mode drop-down button (on page 504) on the toolbar that allows you to select the amount of tags to display in this visual mode. This mode also presents differences that were made using the Track Changes feature (although the Track Changes feature is not available in the comparison tool).

**Figure 100. File Comparison Tool - Author Mode**

Author Mode Algorithms

The visual Author mode offers the following diff algorithms to compare files:

- **Auto** - Selects the most appropriate algorithm, based on the compared content and its size (selected by default).
- **XML Fast** - Comparison that works well on large files or fragments, but it is less precise than XML Accurate.
- **XML Accurate** - Comparison that is more precise than XML Fast, at the expense of speed. It compares two XML files or fragments looking for identical XML nodes.
Author Mode Second-Level Comparisons

The visual Author mode automatically performs a second-level comparison for the XML Fast and XML Accurate algorithms. After the first comparison is finished, the second-level comparison is processed on text nodes using a word level comparison, meaning that it looks for identical words. This second-level comparison makes it easier to spot precise differences and you can merge or reject the precise modifications.

Related information
- Compare Directories Tool (on page 494)
- Compare Files Tool (on page 474)

Generate HTML Report for Directory Comparison

The Generate HTML report for directory comparison tool can be used to generate a report in the form of an HTML file that contains the results of a directory comparison (for either 2-way or 3-way comparisons). The Generate HTML report for directory comparison action for invoking the tool can be found in the Tools menu. It opens a dialog box where you can specify the directories to compare as well as some other options.

![Generate HTML Report for Directory Comparison Dialog Box](image)

The Generate HTML report for directory comparison dialog box contains the following options:

**Base directory**

Specifies the path of the base directory that the other two directories will be compared against in a 3-way comparison. This field should be left empty for 2-way comparisons.
First directory

Specifies the path of the first directory to be included in the comparison.

Second directory

Specifies the path of the second directory to be included in the comparison.

Diff options

Specifies which option set to use for generating the comparison report. If you choose Use the current settings from Preferences, the options set in the Directories Comparison preferences page (on page 289) and the include/exclude filter options in the Compare Directories tool (on page 497) are taken into account when generating the comparison result. You can also click the Diff options button to open the Directories Comparison preferences page where you can see or modify the current settings. If you choose Use the default settings, the default values for all settings are used.

Generate additional file comparison reports

Generates further comparison reports for all non-binary modified file pairs and provides links to them in the main report (in the middle cells of the results table). See the example below (on page 511). These additional file comparison reports are saved to a directory that will have the same parent directory and the same name as the output file provided, suffixed by "-OXY-FC-REPORTS". The links created in the main report are relative to this directory. If the main HTML report is later copied or moved to another location, to retain full functionality in the browser, the directory with the additional file comparison reports must also be copied/moved to the same location.

Note:

Generating additional file comparison reports could significantly increase the execution time. A progress tracker for the whole operation is available.

Output file

Specifies the path for an output file to save the comparison results file.

Open in Browser/System Application

Opens the comparison results file in the browser or system application that is associated with HTML files.

After clicking the Generate report button, a report in the form of an HTML file is generated with details about the comparison results.
Figure 102. HTML Report for Directory Comparison

Figure 103. Example of an Additional File Comparison Report

Resources

For more information about how to generate HTML comparison reports, watch our video demonstration:

https://www.youtube.com/embed/6jPccHKUNNk
Related information

- Compare Directories Tool (on page 494)
- Compare Directories Against a Base (3-Way) Tool (on page 500)
- Compare Files Tool (on page 474)

**Viewing Status Information**

Status information generated by operations such as *schema detection, manual validation, automatic validation*, and *transformations* are fed into the **Information** view, allowing you to monitor how the operation is being executed.

![Figure 104. Information view messages](image)

Messages contain a timestamp, the name of the thread that generated it, and the actual status information. The number of displayed messages can be controlled with the **Maximum number of lines** option (on page 306) in the **Views** preference page.

To make the view visible, select **Window > Show View > Information**.

**Editor Highlights**

*An editor highlight* is a text fragment emphasized by a colored background.

Highlights are generated in both **Text** and **Author** mode when the following actions generate results:
• Find/Replace in Files (on page 436)
• Find/Replace (on page 432)
• Open/Find Resource (on page 426)
• Find All
• Find All Elements (on page 442)
• XPath in Files (on page 409)
• Search References (on page 577)
• Search Declarations (on page 577)

By default, Oxygen XML Editor uses a different color for each type of highlight (XPath in Files, Find/Replace, Search References, Search Declarations, etc.) You can customize these colors and the maximum number of highlights displayed in a document on the Editor preferences page (on page 170). The default maximum number of highlights is 10000.

You can navigate the highlights in the current document by using the following methods:

• Clicking the markers from the range ruler, located at the right side of the editor pane.
• Clicking the Next and Previous buttons (☞) from the bottom of the range ruler, located at the right side of the editor pane.

![Note:]
When there are multiple types of highlights in the document, the Next and Previous buttons (☞) navigate through highlights of the same type.

• Clicking the messages displayed in the Results view (on page 548) at the bottom of the editor.

To remove the highlights, you can do the following:

• Click the Remove all button from bottom of the range ruler, located at the right side of the editor pane.
• Close the results tab at the bottom of the editor that contains the output of the action that generated the highlights.
• Click the Remove all button on the right side of the Results panel (on page 548) at the bottom of the editor.

![Note:]
Use the Highlight all results in editor button (on the right side of the Results panel) to either display all the highlights or hide them.

Printing a Document

Printing is supported in Text, Author, and Grid modes.
The **Print (Ctrl + P (Command + P on macOS))** action that is available from **File** menu displays a series of dialog boxes that allow you to configure print settings. After defining the settings in each dialog box, click **OK** to continue to the next one.

A **Print Preview** action is also available in the **File** menu. It first opens a **Page Setup** dialog box where you can define some paper, orientation, and margin settings. After you click **OK**, it displays the **Print Preview** dialog box where you can see a preview of how the document will look when it is printed.

**Figure 105. Print Preview Dialog Box**

The main window is split in three sections:

- **Preview area** - Displays the formatted document page as it will appear on printed paper.
- **Left stripe** - The left-side stripe that displays a list of thumbnail pages. Clicking any of them displays the page content in the main preview area.
- **Toolbar** - The toolbar area at the top that contains controls for printing, page settings, page navigation, print scaling, and zoom.
Other Printing Features

- If you are printing a document that is opened in the **Author** visual editing mode, you can use the CSS print media type (**on page 2311**) to change the styling in the printed output.

- If you are printing a document that is opened in **Author** mode and it contains **Tracked Changes** (**on page 642**), you can print (or print preview) a copy of the document as if all changes have been accepted by switching the **Track Changes Visualization Mode** (**on page 647**) to **View Final**.

- If you are printing a document that is opened in **Text** mode and line numbers are displayed (the **Show line numbers** option (**on page 172**) is selected), the printed output will include the line numbers.

- If you are printing an XML document that is opened in **Text** mode and the **folding support** (**on page 528**) is activated (the **Enable folding** option (**on page 173**) is selected), the printed output will include the current folded state. Note that this applies to printing an entire document and not selections within the document.

- If you are printing an XML document that is opened in **Text** mode and a block of content is selected, you have the ability to print only the selection of text rather than the entire document. When you invoke the print action with a block of content selected in **Text** mode, a dialog box will be presented that offers you the choice to print the selection or the entire document.
8.

Editing Documents

Oxygen XML Editor includes built-in frameworks for the most popular XML document types (DITA, DocBook, TEI, XHTML, JATS) with a full set of features (full editing support, document templates, enhanced CSS rendering, specific actions, validation, content completion, transformation scenarios, and more). In addition, Oxygen XML Editor provides support for editing numerous other types of documents (all XML document types and even some non-XML formats).

Each type of document has unique features and options and this chapter includes a large amount of information about editing numerous types of documents and various editing features that are provided in Oxygen XML Editor, including general information about editing XML documents in Text mode, the visual Author mode, and Grid mode.

For extensive details about the DITA editing features included in Oxygen XML Editor, see the DITA Authoring chapter.

Related information

Built-in XML Frameworks (Document Types)

Editing XML Documents

The structure of an XML document and the required restrictions on its elements and their attributes are defined with an XML schema. This makes it easier to edit XML documents in the visual Author editing mode. For more information about schema association, see Associating a Schema to XML Documents.

Oxygen XML Editor includes fully supported built-in frameworks for the most popular XML document types (DITA, DocBook, TEI, XHTML, JATS) with a full set of features. These built-in frameworks are defined according to a set of rules and a variety of settings that improve editing capabilities for its particular file type. For extensive details about the DITA editing features included in Oxygen XML Editor, see the DITA Authoring chapter.

This section includes information about the user interface components and actions that are available in the various editing modes and numerous features to help you edit XML documents in any mode.

Related information

Text Editing Mode

Author Editing Mode
Editing XML Documents in Text Mode

This section includes topics that describe how to work with XML documents in Text mode, including various features, actions that are available, and much more.

The Oxygen XML Editor Text editing mode is designed to be a simple, yet powerful, XML source editor. You can use this mode to edit XML code, markup, and text and it provides support to help you transform, and debug XML-based documents. It is similar to other common text editors, but Oxygen XML Editor also includes numerous specialized editing actions, a powerful Content Completion Assistant (on page 532), a helpful Outline view (on page 539), and many other unique features.

To switch to this mode, select Text at the bottom of the editing area.

Navigating the Document Content in Text Mode

Oxygen XML Editor includes some useful features to help you navigate XML documents in Text mode.

Navigation Keyboard Shortcuts

- **Ctrl + CloseBracket (Command + CloseBracket on macOS)**
  - Navigate to the next XML node.
- **Ctrl + OpenBracket (Command + OpenBracket on macOS)**
  - Navigate to the previous XML node.
- **Ctrl + RightArrow (Command + RightArrow on macOS)**
  - Navigate one word forward.
- **Ctrl + LeftArrow (Command + LeftArrow on macOS)**
  - Navigate one word backward.
- **Ctrl + Home (Command + Home on macOS)**
  - Position the cursor at the beginning of the document.
- **Ctrl + End (Command + End on macOS)**
  - Position the cursor at the end of the document.

Navigating to a Modification

Oxygen XML Editor includes some actions that help you to quickly navigate to a particular modification. These navigation buttons are available in the main toolbar (they can also be accessed from the Find menu):

- Last Modification
Navigates to the last modification in any open tab.

← Back

Navigates to the last selected editor tab or to the last selected element/content in the current tab. You can also go back after clicking on links in Text or Author mode.

→ Forward

Available after you use the Back button at least once, and it navigates in the opposite direction as the Back button.

Navigating with the Outline View

Oxygen XML Editor includes an Outline view (on page 539) that displays a hierarchical tag overview of the currently edited XML Document.

You can use this view to quickly navigate through the current document by selecting nodes in the outline tree. It is synchronized with the editor area, so when you make a selection in the Outline view, the corresponding nodes are highlighted in the editor area.

**Figure 106. Outline View Navigation in Text Mode**

Using the Breadcrumb to Navigate

A breadcrumb on the top stripe indicates the path from the document root element to the current element. It can also be used as a helpful tool to navigate to specific elements throughout the structure of the document.

**Figure 107. Breadcrumb in Text Mode**

The last element listed in the breadcrumb is the element at the current cursor position. The current element is also highlighted by a thin light blue bar for easy identification. Clicking an element from the breadcrumb selects the entire element and navigates to it in the editor area.

Navigating with the Go To Dialog Box

In Text mode, you can navigate precisely to a location in the document you are editing by using the Go to dialog box. To open this dialog box, press (Ctrl+L (Command+L on macOS)) or select Find > Go to.
Navigating with Bookmarks

By using bookmarks, you can mark positions in an edited document so that you can return to it later. This is especially helpful for navigating through large documents or while editing multiple documents. You can place up to nine distinct bookmarks in any document. Shortcut keys are available to navigate to any of the marked positions (Ctrl+1 through Ctrl+9). There are also shortcuts for creating bookmarks (Ctrl+Shift+1 through Ctrl+Shift+9). You can also configure these shortcut keys in the Options > Menu Shortcut Keys (on page 294) menu.

To insert a bookmark in Text mode, do any of the following:

- Click in the vertical stripe on the left side of the editor (to the left of the line number).
- Press F9 on your keyboard or use any of the specific bookmark creation shortcuts (Ctrl+Shift+1 through Ctrl+Shift+9).
- Select the Create Bookmark action from the Edit > Bookmarks menu.

To remove bookmark in Text mode, do either of the following:

- Left-click its icon in the vertical stripe.
- Right-click its icon on the vertical stripe and select Remove or Remove all (Ctrl+F7 (Command+F7 on macOS)).
To navigate to a specific bookmark, do either of the following:

- Use any of the specific bookmark navigation shortcuts (Ctrl+1 through Ctrl+9).
- Use one of the actions available on the Edit > Bookmarks > Go to menu.

Tip:
The navigation shortcuts work even if the document where the bookmark was inserted has been closed. In this case, using the shortcut will automatically re-open the document.

Smart Editing in Text Mode

Oxygen XML Editor includes smart editing features to help you edit XML documents in Text mode. The following smart editing features are included:

- **Closing tag auto-expansion** - This feature helps save some keystrokes by automatically inserting a closing tag when you insert a complete start tag and the cursor is automatically placed in between the start and end tags. For instance, after entering a start `<tag>`, the corresponding closing `</tag>` is automatically inserted and the cursor is placed between the two (`<tag>` `</tag>`).
- **Auto-rename matching tag** - When you edit the name of a start tag, Oxygen XML Editor will mirror-edit the name of the matching end tag. This feature can be controlled from the Content Completion option page (on page 211).
- **Auto-breaking the edited line** - The Hard line wrap option (on page 204) automatically breaks the edited line when its length exceeds the maximum line length defined for the format and indent operation (on page 204).
- **Indent on Enter** - The Indent on Enter option (on page 204) indents the new line inserted when you press Enter.
- **Smart Enter** - The Smart Enter option (on page 204) inserts an empty line between the start and end tags. If you press Enter between a start and end tag, the action places the cursor in an indented position on the empty line between the lines that contain the start and end tag.
- **Double-click** - A double-click selects certain text, depending on the position of the click in the document:
  - If the click position is on a start tag or end tag, then the element name is selected.
  - If the click position is immediately after the opening quote or immediately before the closing quote of an attribute value, then the entire attribute value is selected.
  - Otherwise, a double-click selects contiguous text.
- **Triple-click** - A triple-click selects entire regions of text, depending on the click position:
  - If the click position is on a start or end tag, then the entire tag is selected, including the start and end tags, and the content in between.
  - If the click position is after a start tag or before an end tag, then the entire content of the element without the start and end tags is selected.
  - If the click position is before a start tag or after an end tag, then the entire tag is selected, including the start and end tags, and the content in between.
If the click position is immediately before an attribute, then the entire attribute and its value are selected.
If the click position is in between the opening and closing quotes of an attribute value, then the entire attribute value is selected.
Otherwise, it selects the entire current line.

Shortcut Actions in Text Mode

Oxygen XML Editor includes numerous shortcut actions to help you edit content in the Text editing mode.

Changing the Font Size (Zoom)

The font size of the editor panel can be changed with the following actions that are available with shortcuts or in the Document > Font size menu:

Increase editor font (Ctrl + NumPad+ (Command + NumPad+ on macOS) or Ctrl + MouseWheelForward (Windows/Linux)

Increases the font size (zooms in) with one point for each execution of the action.

Note:
For macOS, if you activate the Enable mouse-wheel zooming option (on page 171) in the Editor preferences page, you can use Command + MouseWheelForward to increase the font size (zoom in). It is disabled by default due to the way inertia affects the mouse wheel on macOS.

Decrease editor font (Ctrl + NumPad- (Command + NumPad- on macOS) or Ctrl + MouseWheelBackwards (Windows/Linux)

Decreases the font size (zooms out) with one point for each execution of the action.

Note:
For macOS, if you activate the Enable mouse-wheel zooming option (on page 171) in the Editor preferences page, you can use Command + MouseWheelBackwards to decrease the font size (zoom out). It is disabled by default due to the way inertia affects the mouse wheel on macOS.

Normal editor font (Ctrl + 0 (Command + 0 on macOS))

Resets the font size to the value of the editor font set in the Fonts preferences page (on page 133).

Undo/Redo Actions

The typical undo and redo actions are available with shortcuts or in the Edit menu:

Undo (Ctrl + Z (Command + Z on macOS))
Reverses a maximum of 200 editing actions (configurable with the **Undo history size option (on page 171)** in the **Editor** preferences page) to return to the preceding state.

**Note:**
Complex operations such as **Replace All** or **Indent selection** count as single undo events.

**Redo** *(Ctrl + Y (Command + Shift + Z on macOS, Ctrl + Shift + Z on Linux/Unix))*

Recreates a maximum of 100 editing actions that were undone by the **Undo** function.

**Copy and Paste Actions**

The typical copying and pasting actions are available with shortcuts or in the contextual menu (or the **Edit** menu):

- **Cut** *(Ctrl + X (Command + X on macOS))*
  Removes the currently selected content from the document and places it in the clipboard.

- **Copy** *(Ctrl + C (Command + C on macOS))*
  Places a copy of the currently selected content in the clipboard.

- **Paste** *(Ctrl + V (Command + V on macOS))*
  Inserts the current clipboard content into the document at the cursor position.

- **Select All** *(Ctrl + A (Command + A on macOS))*
  Selects the entire content of the current document.

**Moving XML Nodes**

You can use the following shortcuts to move XML elements or XSLT variables up or down in **Text** mode:

- **Ctrl + Alt + UpArrow (Command + Alt + UpArrow on macOS)**
  Moves the node up one line.

- **Ctrl + Alt + DownArrow (Command + Alt + DownArrow on macOS)**
  Moves the node down one line.

**Note:**
The requirements for these node moving actions to work are as follows:
• The mechanism is designed to work without a selection. If you use these actions on a selection of content, it moves the entire selection. To make this mechanism work as intended, simply position the cursor somewhere on the line that you want to move.
• A start tag must be the first text occurrence on the line where the cursor is positioned.
• On the line where the element ends, only whitespaces are allowed after the end tag.

Miscellaneous Shortcut Actions in Text Mode

Oxygen XML Editor also includes the following other miscellaneous shortcut actions in Text mode:

**Ctrl + Delete (Command + Delete on macOS)**

Deletes the next word.

**Ctrl + Backspace (Command + Backspace on macOS)**

Deletes the previous word.

**Ctrl + W (Command + W on macOS)**

Cuts the previous word.

**Ctrl + K (Command + K on macOS)**

Cuts to end of line.

**Ctrl + Single-Click (Command + Single-Click on macOS)**

Use this shortcut to open any of the following:

- Any absolute URL (URLs that have a protocol), regardless of their location in the document.
- URI attributes such as: `@schemaLocation`, `@noNamespaceSchemaLocation`, `@href` and others.
- Open the target for DITA references (such as a `@conref`, `@conkeyref`, `@keyref`, and more).
- Processing instructions used for associating resources, xml-models, xml-stylesheets.

**Ctrl + Shift + Y (Command + Shift + Y on macOS) (Document > Edit > Toggle Line Wrap)**

Enables or disables line wrapping. When enabled, if text exceeds the width of the displayed editor, content is wrapped so that you do not have to scroll horizontally.

Related information

Frequently Used Shortcut Keys (on page 55)

Editing XML Markup in Text Mode

Oxygen XML Editor includes some useful actions that allow you to easily edit XML markup in Text mode. These actions are available in the Refactoring submenu of the contextual menu and in the Document > Markup menu, and many of the actions can also be done with simple keyboard shortcuts.
Using the Breadcrumb

A breadcrumb on the top stripe indicates the path from the document root element to the current element. It can also be used as a helpful tool to insert and edit specific elements in the document structure.

**Figure 110. Breadcrumb in Text Mode**

<table>
<thead>
<tr>
<th>book</th>
<th>chapter</th>
<th>sect1</th>
<th>sect2</th>
<th>sect3</th>
<th>para</th>
<th>figure</th>
<th>title</th>
</tr>
</thead>
</table>

The last element listed in the breadcrumb is the element at the current cursor position. The current element is also highlighted by a thin light blue bar for easy identification. Clicking an element in the breadcrumb selects the entire element in the editor area. Also, each element provides a contextual menu with access to the following actions:

**Append Child**

Allows you to select an element (from a drop-down list) that is allowed by the associated schema and inserts it as a child of the current element.

**Insert Before**

Allows you to select an element (from a drop-down list) that is allowed by the associated schema and inserts it immediately before the current element, as a sibling.

**Insert After**

Allows you to select an element (from a drop-down list) that is allowed by the associated schema and inserts it immediately after the current element, as a sibling.

**Edit Attributes**

Opens an editing window that allows you to edit the attributes of the currently selected element.

**Toggle Comment**

Encloses the currently selected element in an XML comment, if the element is not already commented. If it is already commented, this action will remove the comment.

**Cut**

Removes the selected element and copies it to the clipboard.

**Copy**

Copies the selected element to the clipboard.

**Delete**

Deletes the currently selected element.

**Move Nodes**

You can easily move XML nodes in the current document by using the following shortcut keys:

**Alt + UpArrow**

Moves the current node or selected nodes in front of the previous node.
Alt + DownArrow

Moves the current node or selected nodes after the subsequent node.

Rename Elements

You can rename elements by using the following actions in the Refactoring submenu of the contextual menu (or from the Document > Markup menu):

- Rename Element

  The element from the cursor position, and any elements with the same name, can be renamed according with the options from the Rename dialog box.

- Rename Prefix (Alt + Shift + P (Command + Shift + P on macOS))

  The prefix of the element from the cursor position, and any elements with the same prefix, can be renamed according with the options from the Rename dialog box.

  - If you select the Rename current element prefix option, the application will recursively traverse the current element and all its children. For example, to change the xmlns:p1="ns1" association in the current element to xmlns:p5="ns1", if the xmlns:p1="ns1" association is applied on the parent element, then Oxygen XML Editor will introduce xmlns:p5="ns1" as a new declaration in the current element and will change the prefix from p1 to p5. If p5 is already associated with another namespace in the current element, then the conflict will be displayed in a dialog box. By pressing OK, the prefix is modified from p1 to p5 without inserting a new declaration.

  - If you select the Rename current prefix in all document option, the application will apply the change on the entire document.

  - To also apply the action inside attribute values, select the Rename also attribute values that start with the same prefix checkbox.

Surround Content with Tags (Wrap)

You can surround a selection of content with tags (wrap the content) by using the following action in the Refactoring submenu of the contextual menu (or from the Document > Markup menu):

- Surround with submenu

  Presents a drop-down menu that allows you to choose a tag to surround a selected portion of content.

- Surround with Tags (Ctrl + E (Command + E on macOS))

  Allows you to choose a tag that encloses a selected portion of content. If there is no selection, the start and end tags are inserted at the cursor position.
• If the Position cursor between tags option (on page 212) is selected in the Content Completion preferences page, the cursor is placed between the start and end tag.
• If the Position cursor between tags option (on page 212) is not selected in the Content Completion preferences page, the cursor is placed at the end of the start tag, in an insert-attribute position.

♂ Surround with `[tag]` (Ctrl + ForwardSlash (Command + ForwardSlash on macOS))

Surround the selected content with the last tag used.

Unwrap the Content of Elements

You can unwrap the content of an element by using the following action in the Refactoring submenu of the contextual menu (or from the Document > Markup menu):

♂ Delete element tags (Alt + Shift + X (Command + Alt + X on macOS))

Deletes the start and end tag of the current element.

Join or Split Elements

You can join or split elements in the current document by using the following actions in the Refactoring submenu of the contextual menu (or from the Document > Markup menu):

♂ Join elements (Alt + Shift + J (Command + Alt + J on macOS))

Joins the left and right elements relative to the current cursor position. The elements must have the same name, attributes, and attributes values.

♂ Split element (Alt + Shift + D (Ctrl + Alt + D on macOS))

Split the element from the cursor position into two identical elements. The cursor must be inside the element.

Other Refactoring Actions

You can also manage the structure of the markup by using the other specific XML refactoring actions that are available in the Refactoring submenu of the contextual menu:

Attributes Refactoring Actions

Contains built-in XML refactoring operations that pertain to attributes with some of the information preconfigured based upon the current context.

Add/Change attribute

Allows you to change the value of an attribute or insert a new one.

Convert attribute to element

Allows you to change an attribute into an element.

Delete attribute
Allows you to remove one or more attributes.

**Rename attribute**

Allows you to rename an attribute.

**Replace in attribute value**

Allows you to search for a text fragment inside an attribute value and change the fragment to a new value.

**Comments Refactoring Actions**

Contains built-in XML refactoring operations that pertain to comments with some of the information preconfigured based upon the current context.

**Delete comments**

Allows you to delete comments found inside one or more elements.

**Elements Refactoring Actions**

Contains built-in XML refactoring operations that pertain to elements with some of the information preconfigured based upon the current context.

**Delete element**

Allows you to delete elements.

**Delete element content**

Allows you to delete the content of elements.

**Insert element**

Allows you to insert new elements.

**Rename element**

Allows you to rename elements.

**Unwrap element**

Allows you to remove the surrounding tags of elements, while keeping the content unchanged.

**Wrap element**

Allows you to surround elements with element tags.

**Wrap element content**

Allows you to surround the content of elements with element tags.

**Fragments Refactoring Actions**

Contains built-in XML refactoring operations that pertain to XML fragments with some of the information preconfigured based upon the current context.

**Insert XML fragment**
Allows you to insert an XML fragment.

**Replace element content with XML fragment**

Allows you to replace the content of elements with an XML fragment.

**Replace element with XML fragment**

Allows you to replace elements with an XML fragment.

---

**Folding XML Elements in Text Mode**

When working with a large document, the folding (on page 3219) support in Oxygen XML Editor can be used to collapse some element content leaving only those that you need to edit in focus. Expanding and collapsing works on individual elements. Expanding an element leaves the child elements unchanged.

By default, the folding (on page 3219) feature is enabled in Oxygen XML Editor, but it can be disabled in the Text preferences page with the Enable folding option (on page 173).

The fact that the folds are persistent is a unique feature of Oxygen XML Editor. The next time you open the document the folds are restored to its last state.

**Folding Actions in Text Mode**

Element folds are marked with a small triangle (✓/✓) in the left stripe. To toggle the fold, simply click the icon. Also, if you right-click the icon, the following actions are available:

- **Collapse Other Folds (Ctrl + NumPad/ (Command + NumPad/ on macOS))**
  
  Folds all the elements except the current element.
Collapse Child Folds (Ctrl + NumPad. (Command + NumPad. on macOS))
Folds the child elements that are indented one level inside the current element.

Expand Child Folds
Unfolds all child elements of the currently selected element.

Expand All (Ctrl + NumPad* (Command + NumPad* on macOS))
Unfolds all elements in the current document.

For more information about the folding support in Oxygen XML Editor, watch our video demonstration:
https://www.youtube.com/embed/eR9HfN_peAE

Drag and Drop in Text Mode
To move a whole region of text to other location in the same edited document, just select the text, drag the selection by holding down the left mouse button and drop it to the target location.

You can also copy content from other applications and paste it into the document.

Selecting Content in Text Mode
Oxygen XML Editor includes a variety of keyboard shortcuts that allow you to select content in Text mode. These include numerous standard continuous selection possibilities that are common to many text editors, as well as a selection feature that allows you to select a rectangular area within a document in Text mode.

Standard Continuous Selection Shortcuts

- **Ctrl + A (Meta + A on macOS)**
  Selects all content in the document.

- **Shift + Left/Right Arrow Keys**
  Begins a continuous selection at the cursor position and extends it one character at a time in the direction that you press the arrow keys.

- **Shift + Up/Down Arrow Keys**
  Begins a continuous selection at the cursor position and extends it one line at a time in the direction that you press the arrow keys.

- **Ctrl + Shift + Left/Right Arrow Keys (Meta + Shift + Left/Right Arrow Keys on macOS)**
  Begins a continuous selection at the cursor position and extends it one word at a time in the direction that you press the arrow keys.

- **Shift + Home**
  Begins a continuous selection at the cursor position and extends it to the beginning of the current line (on macOS, it extends to the beginning of the document).
**Shift + End**

Begins a continuous selection at the cursor position and extends it to the end of the current line (on macOS, it extends to the end of the document).

**Ctrl + Shift + Home**

Begins a continuous selection at the cursor position and extends it to the beginning of the document.

**Ctrl + Shift + End**

Begins a continuous selection at the cursor position and extends it to the end of the document.

**Shift + PageUp**

Begins a continuous selection at the cursor position and extends it up one screen page.

**Shift + PageDown**

Begins a continuous selection at the cursor position and extends it down one screen page.

**Double-Click**

Selects certain text, depending on the position of the click in the document. See Smart Editing: Double-Click *(on page 520)* for the specifics.

**Triple-Click**

Selects entire regions of text, depending on the position of the click in the document. See the Smart Editing: Triple-Click *(on page 520)* for the specifics.

**Right-Click > Select > Element**

Selects the entire element at the current cursor position.

**Right-Click > Select > Content**

Selects the entire content of the element at the current cursor position, excluding the start and end tag. Performing this action repeatedly will result in the selection of the content of the ancestor of the currently selected element content.

**Right-Click > Select > Attributes**

Selects all the attributes of the element at the current cursor position.

**Right-Click > Select > Parent**

Selects the entire parent element at the current cursor position.

**Rectangular Selection Shortcuts**

Oxygen XML Editor also includes some keyboard shortcuts that allow you to select a rectangular block of content in Text mode and you can then copy, cut, paste, delete, or edit the selection.
Attention:
The rectangular selection shortcuts will not work if the Line Wrap option (on page 173) is selected in the Text preferences page.

The following shortcuts can be used to create a rectangular selection:

**Alt + Mouse Click + Mouse Movement** *(Alt + Meta + Mouse Click + Mouse Movement on macOS)*

Begin a rectangular selection at the mouse click position and extends it in the direction that you move the mouse. Release **Alt** *(Alt + Meta on macOS)* to enter the in-place editing mode *(on page 531).*

**Shift + Alt + Left/Right Arrow Keys** *(Shift + Alt + Meta + Left/Right Arrow Keys on macOS)*

Begin a rectangular selection at the cursor position and extends it one character at a time in the direction that you press the arrow keys (you can also use the mouse to extend the selection).

**Shift + Alt + Up/Down Arrow Keys** *(Shift + Alt + Meta + Up/Down Arrow Keys on macOS)*

Begin a rectangular selection at the cursor position and extends it one line at a time in the direction that you press the arrow keys (you can also use the mouse to extend the selection).

**Ctrl + Shift + Alt + Left/Right Arrow Keys** *(Ctrl + Shift + Alt + Meta + Left/Right Arrow Keys on macOS)*

Begin a rectangular selection at the cursor position and extends it one word at a time in the direction that you press the arrow keys.

**Shift + Alt + Home** *(Shift + Alt + Meta + Home on macOS)*

Begin a rectangular selection at the cursor position and extends it to the beginning of the current line.

**Shift + Alt + End** *(Shift + Alt + Meta + End on macOS)*

Begin a rectangular selection at the cursor position and extends it to the end of the current line.

**Shift + Alt + PageUp** *(Shift + Alt + Meta + PageUp on macOS)*

Begin a rectangular selection at the cursor position and extends it up one screen page.

**Shift + Alt + PageDown** *(Shift + Alt + Meta + PageDown on macOS)*

Begin a rectangular selection at the cursor position and extends it down one screen page.

You can then use standard editing actions to copy, cut, paste, or delete the entire selection.

**In-Place Editing Mode**

To edit the content of the rectangular selection, you can enter an in-place editing mode by releasing the **Alt** key (on macOS, release both **Alt & Meta**). Once you are in the editing mode, you can simply use your keyboard to edit the entire selection of content, or click anywhere inside the selection to edit the content at the cursor position for all lines within the selection at once (as if the rectangular selection is a selection of columns). To exit the editing mode, press either **Enter** or **Esc**.
Content Completion Assistant in Text Mode

Oxygen XML Editor includes an intelligent Content Completion Assistant that offers proposals for inserting structured language elements, attributes, and attribute values that are valid in the current editing context.

The Content Completion Assistant is enabled by default. To disable it, open the Preferences dialog box (Options > Preferences), go to Editor > Content Completion, and deselect the Enable content completion option.

Figure 112. Content Completion Assistant

Content Completion and the Associated Schema

The Content Completion Assistant feature is schema-driven and the list of proposals in the Content Completion Assistant depend on the associated schemas (DTD, XML Schema, Relax NG, or NVDL schema). For information about the various ways to associate a schema and the order of their precedence, see the Associating a Schema to XML Documents section.

Using the Content Completion Assistant in Text Mode

The feature is activated in Text mode in the following situations:

- After you enter the < character when inserting an element, it is automatically activated after a short delay. You can adjust the activation delay with the Activation delay of the proposals window (ms) option from the Content Completion preferences page.
- After typing a partial element or attribute name, you can manually activate it by pressing Ctrl + Space (Command + Space on macOS) or Alt + ForwardSlash (Command + Alt + ForwardSlash on macOS). If there is only one valid proposal at the current location, it is inserted without displaying the list of proposals.

You can navigate through the list of proposals by using the Up and Down keys on your keyboard. In some cases, the Content Completion Assistant displays a documentation window with information about the particular proposal and some of them have links to additional information (for example, DITA elements might have a link to the DITA Style Guide). You can also change the size of the documentation window by dragging its top, right, and bottom borders.
To insert the selected proposal in **Text** mode, do one of the following:

- Press **Enter** or **Tab** to insert both the start and end tags and position the cursor inside the start tag in a position suitable for inserting attributes.
- Press **Ctrl + Enter** (**Command + Enter** on macOS) to insert both the start and end tags and positions the cursor between the tags in a position where you can start typing content.

**Note:**

When the DTD, XML Schema or RELAX NG schema specifies required child elements for the newly added element, they are inserted automatically only if the Add Element Content option *(on page 212)* (in the Content Completion preferences page) is selected. The Content Completion Assistant can also add optional content and first choice particle, as specified in the DTD, XML Schema, or RELAX NG schema. To activate these features, select the Add optional content *(on page 212)* and Add first Choice particle *(on page 212)* options in the Content Completion preferences page.

After inserting an element, the cursor is positioned:

- Before the > character of the start tag, if the element allows attributes, to allow rapid insertion of any of the attributes supported by the element. Pressing the space bar displays the Content Completion list once again. This time it contains the list of allowed attribute names. If the attribute supports a fixed set of parameters, the assistant list displays the list of valid parameters. If the parameter setting is user-defined and therefore variable, the assistant is closed to allow manual insertion. The values of the attributes can be learned from the same elements in the current document.
- After the > character of the start tag, if the element has no attributes.

**Where the Content Completion Assistant is Displayed**

The Content Completion Assistant is displayed:

- Anywhere within a tag name or at the beginning of a tag name in an XML document, XML Schema, DTD, or Relax NG (full or compact syntax) schema.
- Anywhere within an attribute name or at the beginning of an attribute name in any XML document with an associated schema.
- Within attribute values or at the beginning of attribute values in XML documents where lists of possible values have been defined for that element in the schema associated with the document.

**Types of Proposals Listed in the Content Completion Assistant**

The following things are considered for determining the proposals that are listed in the content completion window:

- **Element Structure Specified in DTD or Schema**
The proposals that populate the **Content Completion Assistant** depend on the element structure specified in the DTD, XML Schema, Relax NG (full or compact syntax) schema, or NVDL schema associated with the edited document.

**Note:**

The **Content Completion Assistant** is able to offer elements defined both by XML Schemas version 1.0 and 1.1.

**Current Cursor Position**

The number and type of elements displayed by the **Content Completion Assistant** is dependent on the cursor’s current position in the structured document. The child elements displayed within a given element are defined by the structure of the specified DTD, XML Schema, Relax NG (full or compact syntax) schema, or NVDL schema.

**Unique ID Attribute Values**

A schema may declare certain attributes as **ID** or **IDREF/IDREFS**. When the document is validated, Oxygen XML Editor checks the uniqueness and correctness of the **@id** attributes. It also collects the attribute values declared in the document to prepare the list of proposals. This is available for documents that use DTD, XML Schema, and Relax NG schema.

**Values for xml:id Attributes**

Values of all the **@xml:id** attributes are handled as **@id** attributes. They are collected and displayed by the **Content Completion Assistant** as possible values for **anyURI** attributes defined in the schema of the edited document. This works only for XML Schema and Relax NG schemas.

**Links/References in DITA**

When entering values for the various types of links and references in DITA (for example, values for **@href** or **@conref** elements), the **Content Completion Assistant** will propose potential targets when you use the forward slash key (\/).

**ID Values for DITA Key References**

In DITA, when inserting key references (**@keyref**) or content key references (**@conkeyref**), the ID values that are defined in the key reference are presented as possible targets. The **Content Completion Assistant** will only propose targets that are valid in the current context.

**Element and Attribute Values**

For documents that use an XML Schema or Relax NG schema, the **Content Completion Assistant** offers proposals for attribute and element values as long as the allowed values are defined in the schema. Also, if a default value or fixed value is defined in the schema, then that value is offered in the **Content Completion Assistant**.

**Related information**

[Customizing the Content Completion Assistant Using a Configuration File](on page 2191)
Schema Annotations in Text Mode

A schema annotation is a documentation snippet associated with the definition of an element or attribute in a schema. If such a schema is associated with an XML document, the annotations are displayed in:

- The Content Completion Assistant (on page 3217).
- A small tooltip window shown when the mouse hovers over an element or attribute. The tooltip window can be invoked at any time by using the F2 shortcut.

The schema annotations support is available if the schema type is one of the following:

- XML Schema
- Relax NG
- NVDL schema
- DTD

This feature is enabled by default, but you can disable it by deselecting the Show annotations in Content Completion Assistant (on page 216) option in the Annotations preferences page.

Styling Annotations with HTML

You can use HTML format in the annotations you add in an XML Schema or Relax NG schema. This improves the visual appearance and readability of the documentation window displayed when editing XML documents validated against such a schema. An annotation is recognized and displayed as HTML if it contains at least one HTML element (such as `<div>`, `<body>`, `<p>`, `<br>`, `<table>`, `<ul>`, or `<ol>`).

The HTML rendering is controlled by the Show annotations using HTML format, if possible (on page 216) option in the Annotations preferences page. When this option is deselected, the annotations are converted and displayed as plain text and if the annotation contains one or more HTML tags (`<p>`, `<br>`, `<ul>`, `<li>`), they are rendered as an HTML document loaded in a web browser. For example, `<p>` begins a new paragraph, `<br>` breaks the current line, `<ul>` encloses a list of items, and `<li>` encloses an item of the list.

Collecting Annotations from XML Schemas

In an XML Schema, the annotations are specified in an `<xs:annotation>` element like this:

```xml
<xs:annotation>
  <xs:documentation>
    Description of the element.
  </xs:documentation>
</xs:annotation>
```

If an element or attribute does not have a specific annotation, then Oxygen XML Editor looks for an annotation in the type definition of that element or attribute.
Collecting Annotations from Relax NG Schemas

For Relax NG schema, element and attribute annotations are made using the `<documentation>` element from the http://relaxng.org/ns/compatibility/annotations/1.0 namespace like this:

```xml
<define name="person">
  <element name="person">
    <a:documentation xmlns:a="http://relaxng.org/ns/compatibility/annotations/1.0">
      Information about a person. </a:documentation>
    </ref>
    <zeroOrMore>
      <ref name="email"/>
    </zeroOrMore>
  </element>
</define>
```

However, any element outside the Relax NG namespace (http://relaxng.org/ns/structure/1.0) is handled as annotation and the text content is displayed in the annotation window. To activate this behavior, select the Use all Relax NG annotations as documentation (on page 217) option in the Annotations preferences page.

Collecting Annotations from Relax NG Compact Syntax Schemas

For Relax NG Compact Syntax schema, annotations are made using comments like this:

```plaintext
## Information about a person.

element person { name, email*}
```

Collecting Annotation from DTDs

For DTD, Oxygen XML Editor defines a custom mechanism for annotations using comments enabled by the Prefer DTD comments that start with "doc:" as annotations (on page 216) option in the Annotations preferences page. The following is an example of a DTD annotation:

```
<!--doc:Description of the element. -->
```

Content Completion Helper Views (Text Mode)

Information about the current element being edited is also available in various dockable (on page 3218) views, such as the Model view (on page 545), Attributes view (on page 542), Elements view (on page 546), and Entities view (on page 547). By default, they are located on the right-hand side of the main editor window. These views, along with the powerful Outline view (on page 539), provide spatial and insight information about the edited document and the current element. If any particular view is not displayed, it can be opened by selecting it from the Window > Show View menu.

Code Templates

Code templates are code fragments that can be inserted quickly at the current editing position. Oxygen XML Editor includes a set of built-in code templates for CSS, LESS, Schematron, XSL, XQuery, JSON, HTML, and
XML Schema document types. You can also define your own code templates for any type of file and share them with others.

Code templates are displayed with a symbol in the content completion list (Enter in Author mode or Ctrl + Space (Command + Space on macOS) in Text mode). Also, in Text mode you can press Ctrl + Shift + Space to see a complete list of the available code templates. To enter the code template at the cursor position, select it from the content completion list or type its code and press Enter. If a shortcut key has been assigned to the code template, you can also use the shortcut key to enter it.

How to Create Code Templates

To create a code template, follow these steps:

1. Open the Preferences dialog box (Options > Preferences) (on page 124) and go to Editor > Content Completion > Code Templates.

2. Click New to open a code template configuration dialog box.

   Tip:
   You can use one of the existing code templates as a starting point by selecting that template and clicking Duplicate.

   ![Figure 113. Code Template Configuration Dialog Box](image)

3. Configure your template using the fields in the code template configuration dialog box:
   - **Name** - The name of the code template.
   - **Description** - [Optional] The description of the code template that will appear in the Code Templates preferences page and in the tooltip message when selecting it from the Content Completion Assistant (on page 3217). HTML markup can be used for better rendering.
   - **Associate with** - You can choose to set the code template to be associated with a specific type of editor or for all editor types.
Shortcut key - [Optional] If you want to assign a shortcut key that can be used to insert the code template, place the cursor in the Shortcut key field and press the desired key combination on your keyboard. Use the Clear button if you make a mistake. If the Enable platform-independent shortcut keys checkbox is selected, the shortcut is platform-independent and the following modifiers are used:

- **M1** represents the Command key on macOS, and the Ctrl key on other platforms.
- **M2** represents the Shift key.
- **M3** represents the Option key on macOS, and the Alt key on other platforms.
- **M4** represents the Ctrl key on macOS, and is undefined on other platforms.

Content - Text box where you define the content that is used when the code template is inserted. An editor variable (on page 323) can be inserted in the text box using the Insert Editor Variables button.

4. Click OK to save your new code template.

Result: Your code template can now be selected using the Content Completion Assistant (on page 3217) (Enter in Author mode or Ctrl + Space (Command + Space on macOS) in Text mode). The code templates are displayed with a symbol.

**How to Share Code Templates**

There are two ways to easily share all of your code templates with other members of your team:

**Method 1: Export/Import**

1. Open the Preferences dialog box (Options > Preferences) (on page 124) and go to Editor > Templates > Code Templates.
2. Click the Export button to export all of your code templates into an XML file.
3. Save the XML file.
4. Share the XML file with other members of your team.

5. Instruct them to open the Preferences dialog box (Options > Preferences) (on page 124), go to Editor > Templates > Code Templates, click the Import button, and select the file you sent them.

**Result:** The code templates will be now available in their content completion list.

**Method 2: Share Project**

1. Open the Preferences dialog box (Options > Preferences) (on page 124) and go to Editor > Templates > Code Templates.
2. Select Project Options at the bottom of the dialog box. This stores the preferences in the project file (.xpr).
3. Share the project file with the other members of your team. For example, you can commit it to your version control system and have them update their working copy.

**Result:** When they open the updated project file in their Project view (on page 402), the code templates will be available in their content completion list.

**Tip:** It is also possible to configure certain actions that function similar to code templates and add them to the content completion list (on page 2191) for a particular framework. You could then share the whole framework (on page 2288) with other members of your team.

**Text Mode Views**

There is a variety of dockable (on page 3218) helper views that are displayed by default in Text mode. There are also a large selection of additional views available in the Window > Show View menu. This section presents some of the most helpful views for editing in Text mode.

**Outline View for XML Documents**

The Outline view displays a general tag overview of the currently edited XML document. When you edit a document, the Outline view dynamically follows the changes that you make, displaying the node that you modify. This functionality gives you great insight on the location of your modifications in the current document. It also shows the correct hierarchical dependencies between elements. This makes it easy for you to be aware of the document structure and the way element tags are nested.

**Outline View Features**

The Outline view allows you to:

- Quickly navigate through the document by selecting nodes in the Outline tree.
- Insert or delete nodes using contextual menu actions.
- Move elements by dragging them to a new position in the tree structure.
- Highlight elements in the editor area. It is synchronized with the editor area, so when you make a selection in the editor area, the corresponding nodes are highlighted in the Outline view, and vice versa.
- View document errors, as they are highlighted in the Outline view. A tooltip also provides more information about the nature of the error when you hover over the faulted element.

**Outline View Interface**

By default, it is displayed on the left side of the editor. If the view is not displayed, it can be opened by selecting it from the Window > Show View menu.

The upper part of the Outline view contains a filter box that allows you to focus on the relevant components. Type a text fragment in the filter box and only the components that match it are presented. For advanced usage you can use wildcard characters (such as * or ?) and separate multiple patterns with commas.
It also includes a Settings menu in the top-right corner that presents a variety of options to help you filter the view even further.

**Drag and Drop Actions in the Outline View**

Entire XML elements can be moved or copied in the edited document using only the mouse in the Outline view with drag-and-drop operations. Several drag and drop actions are possible:

- If you drag an XML element in the Outline view and drop it on another node, then the dragged element will be moved after the drop target element.
- If you hold the mouse pointer over the drop target for a short time before the drop then the drop target element will be expanded first and the dragged element will be moved inside the drop target element after its opening tag.
- You can also drop an element before or after another element if you hold the mouse pointer towards the upper or lower part of the targeted element. A marker will indicate whether the drop will be performed before or after the target element.
- If you hold down the Ctrl (Command on macOS) key after dragging, a copy operation will be performed instead of a move.

**Figure 114. Outline View**

![Outline View]

**Outline View Filters**

The upper part of the Outline view contains a filter box that allows you to focus on the relevant components. Type a text fragment in the filter box and only the components that match it are presented. For advanced usage you can use wildcard characters (such as * or ?) and separate multiple patterns with commas.

The following actions are available in the Settings menu of the Outline view:
Filter returns exact matches

The text filter of the Outline view returns only exact matches.

*Selection update on cursor move (Available in Text mode)*

Controls the synchronization between Outline view and source document. The selection in the Outline view can be synchronized with the cursor moves or the changes in the editor. Selecting one of the components from the Outline view also selects the corresponding item in the source document.

**Flat presentation mode of the filtered results**

When active, the application flattens the filtered result elements to a single level.

*Show comments and processing instructions*

Show/hide comments and processing instructions in the Outline view.

*Show element name*

Show/hide element name.

*Show text*

Show/hide additional text content for the displayed elements.

*Show attributes*

Show/hide attribute values for the displayed elements. The displayed attribute values can be changed from the Outline preferences panel (on page 306).

*Configure displayed attributes*

Displays the XML Structured Outline preferences page (on page 306).

Outline View Contextual Menu Actions

The contextual menu of the Outline view contains the following actions:

*Edit Attributes*

Displays an in-place attributes editor that allows you to edit the attributes of a selected node.

*Edit Profiling Attributes (Available in Author mode)*

Allows you to change the profiling attributes (on page 668) defined on all selected elements.

*Append Child*

Invokes a content completion list with the names of all the elements that are allowed by the associated schema and inserts your selection as a child of the current element.

*Insert Before*

Invokes a content completion list with the names of all the elements that are allowed by the associated schema and inserts your selection immediately before the current element, as a sibling.
Insert After

Invokes a content completion list with the names of all the elements that are allowed by the associated schema and inserts your selection immediately after the current element, as a sibling.

Cut, Copy, Paste, Delete common editing actions

Executes the typical editing actions on the currently selected elements. The Cut and Copy operations preserve the styles of the copied content.

Paste before (Available in Author mode)

Inserts a well-formed copied element before the currently selected element.

Paste after (Available in Author mode)

Inserts a well-formed copied element after the currently selected element.

Paste as XML (Available in Author mode)

Pastes copied content that is considered to be valid XML, preserving its XML structure.

Toggle Comment

Encloses the currently selected element in an XML comment, if the element is not already commented. If it is already commented, this action will remove the comment.

Rename Element (Available in Author mode)

Invokes a Rename dialog box that allows you to rename the currently selected element, siblings with the same name, or all elements with the same name.

Expand More

Expands the structure tree of the currently selected element.

Collapse All

Collapses all of the structure tree of the currently selected node.

Tip:

You can copy, cut or delete multiple nodes in the Outline by using the contextual menu after selecting multiple nodes in the tree.

Attributes View in Text Mode

The Attributes view presents all the attributes of the current element determined by the schema of the document. By default, it is located on the right side of the editor. If the view is not displayed, it can be opened from the Window > Show View menu.

You can use the Attributes view to insert attributes, edit their values, or add values to existing attributes.

The attributes are rendered differently depending on their state:
• The names of the attributes are rendered with a bold font, and their values with a plain font.
• Default values are rendered with a plain font, painted gray.
• Empty values display the text "[empty]", painted gray.
• Invalid attributes and values are painted red.

To edit the value of the corresponding attribute, double-click a cell in the Value column. If the possible values of the attribute are specified as list in the schema of the edited document, the Value column acts as a combo box that allows you to either select the value from a list or manually enter it.

You can sort the attributes table by clicking the Attribute column header. The table contents can be sorted as follows:

• By attribute name in ascending order.
• By attribute name in descending order.
• Custom order, where the used attributes are displayed at the beginning of the table sorted in ascending order, followed by the rest of the allowed elements sorted in ascending order.

Figure 115. Attributes View

Expand/Collapse Button

There is an Expand/Collapse (↑ / ↓) button at the top-right of the view. When expanded, this presents the following additional combo boxes:

Name Combo Box

Use this combo box to select an attribute. The drop-down list displays the list of possible attributes allowed by the schema of the document, as in the Attributes view. You can use the Remove button to delete an attribute and its value from the selected element.

Value Combo Box
Use this combo box to add, edit, or select the value of an attribute. If the selected attribute has predefined values in the schema, the drop-down list displays those possible values. You can use the Browse button to select a URL for the value of an attribute. You can also press Ctrl + Space to open a content completion window that offers a list of possible choices and allows you to select multiple values. After you have entered or selected a value, use the Update button (or press Enter) to add the value to the attribute.

**Note:**
For built-in frameworks, if the selected attribute in the Name field is an @id attribute, the Browse button is replaced by a Generate Unique ID Value button. Clicking this button will automatically generate a unique ID for the selected element.

### Contextual Menu Actions in the Attributes View

The following actions are available in the contextual menu of the Attributes view when editing in Text mode:

**Add**

Allows you to insert a new attribute. Adding an attribute that is not in the list of all defined attributes is not possible when the Allow only insertion of valid elements and attributes (on page 180) schema-aware option is selected.

**Set empty value**

Specifies the current attribute value as empty.

**Remove**

Removes the attribute (action available only if the attribute is specified). You can invoke this action by pressing the Delete or Backspace keys.

**Copy**

Copies the attrName="attrValue" pair to the clipboard. The attrValue can be:

- The value of the attribute.
- The value of the default attribute, if the attribute does not appear in the edited document.
- Empty, if the attribute does not appear in the edited document and has no default value set.

**Paste**

Depending on the content of the clipboard, the following cases are possible:
• If the clipboard contains an attribute and its value, both of them are introduced in the Attributes view. The attribute is selected and its value is changed if they exist in the Attributes view.
• If the clipboard contains an attribute name with an empty value, the attribute is introduced in the Attributes view and you can start editing it. The attribute is selected and you can start editing it if it exists in the Attributes view.
• If the clipboard only contains text, the value of the selected attribute is modified.

Model View

The Model view presents the structure of the currently selected tag, and its documentation, defined as annotation in the schema of the current document. By default, it is located on the right side of the editor. If the view is not displayed, it can be opened by selecting it from the Window > Show View menu.

Figure 116. Model View

The Model view is comprised of two sections, an element structure panel and an annotations panel.

Element Structure Panel

The element structure panel displays the structure of the currently edited or selected tag in a tree-like format. The information includes the name, model, and attributes of the current tag. The allowed attributes are shown along with imposed restrictions, if any.
Annotation Panel

The Annotation panel displays the annotation information for the currently selected element. This information is collected from the XML schema.

Elements View in Text Mode

The Elements view presents a list of all defined elements that are valid at the current cursor position according to the schema associated to the document. By default, it is located on the right side of the editor. If the view is not displayed, it can be opened by selecting it from the Window > Show View menu.

Double-clicking any of the listed elements inserts that element into the edited document, at the current cursor position. Pressing F2 with an element selected will display information about that particular element.
Entities View

*Entities* provide abbreviated entries that can be used in XML files when there is a need of repeatedly inserting certain characters or large blocks of information. An *entity* is defined using the `ENTITY` statement either in the DOCTYPE declaration or in a DTD file associated with the current XML file.

There are three types of entities:

- **Predefined** - Entities that are part of the predefined XML markup (`&lt;`, `&gt;`, `&amp;`, `&apos;`, `&quot;`).
- **Internal** - Defined in the DOCTYPE declaration header of the current XML.
- **External** - Defined in an external DTD module included in the DTD referenced in the XML DOCTYPE declaration.

**Note:**

If you want to add internal entities, you would need to switch to the Text editing mode and manually modify the DOCTYPE declaration. If you want to add external entities, you need to open the DTD module file and modify it directly.

The *Entities* view displays a list with all entities declared in the current document, as well as built-in ones. By default, it is located on the right side of the editor. If the view is not displayed, it can be opened by selecting it from the *Window > Show View* menu.

Double-clicking one of the entities will insert it at the current cursor position in the XML document. You can also sort entities by name and value by clicking the column headers.
The view features a filtering capability that allows you to search an entity by name, value, or both. Also, you can choose to display the internal or external entities.

**Note:**

When entering filters, you can use the `?` and `*` wildcards. Also, you can enter multiple filters by separating them with a comma.

---

### Results View

The Results view displays the messages generated as a result of user actions such as validations, transformations, search operations, and others. Each message is a link to the location related to the event that triggered the message. Double-clicking a message opens the file containing the location and positions the cursor at the location offset. The Results view is automatically opened when certain actions generate result messages. By default, the view normally opens at the bottom of the editor, but it is [dockable](on page 3218), so it can be moved to another UI location alongside other side views.

**Tip:**

To shift focus to the open Results view without using the mouse, there is an action in the **Window > Results** menu called **Focus Results** that can be used for this purpose and you can assign a keyboard shortcut (on page 294) to this action.

The actions that contribute messages to this view include:
• **Validation** actions *(on page 775)*

• **Transformation** actions *(on page 1432)*

• **Check Spelling in Files** action *(on page 458)*

• **Find All** action from the **Find/Replace** dialog box *(on page 432)*

• **Find/Replace in Files** dialog box *(on page 436)*

• **Search References** action *(on page 911)*

• **XPath expression results** *(on page 2000)*

• **SQL results** *(on page 2073)*

**Figure 121. Results View**

**Results View Toolbar Actions**

The view includes a toolbar with the following actions:

- **Settings drop-down menu**

  This drop-down menu also includes the following options:

  - **Group by "Severity"**
    
    Groups the results based upon the severity of the validation issues.

  - **Group by "Resource"**
    
    Groups the results based upon the type of resource.

  - **Group by "System ID"**
    
    Groups the results based upon the system ID of the resource.

  - **Group by "Operation description"**
    
    Groups the results based upon the description of the validation issue.

  - **Ungroup all**
    
    Removes the grouping rules so that the messages are presented in a continuous list.

- **Show group columns**

  If any of the **Group by** options are selected, you can use this option to show or hide grouping columns.

- **Restore default grouping**
Restores the column size for each column and the grouping rules that were saved in the user preferences the last time when this view was used. If it is the first time this view is used, the action sets an initial default column size for each column and a grouping rule that is appropriate for the type of messages. For example:

- Group the messages by the path of the validated file if there are error messages from a validation action or spelling errors reported by the Check Spelling in Files action (on page 458).
- No grouping rule for the results of applying an XPath expression (on page 1999).

Include problem ID in description

If this option is selected, validation issues will include the problem ID (as provided by the validation engine) in the Description column.

Show Ignored Problems

If you have ignored validation problems (on page 812), you can deselect this option to hide the ignored problems. Likewise, you can select this option to show the ignored problems.

👉 Highlight all results in editor

Oxygen XML Editor highlights all matches obtained after executing an XPath expression, or performing one of the following operations: Find All, Find in Files, Search References, and Search Declarations. Click Highlight all results in editor again to turn off highlighting.

Note:

To customize highlighting behavior, open the Preferences dialog box (Options > Preferences) (on page 124) and go to Editor > Highlights category. You can do the following customizations:

- Set a specific color of the highlights depending on the type of action you make.
- Set a maximum number of highlights that the application displays at any given time.

❌ Remove selected

Removes the current selection from the view. This can be helpful if you want to reduce the number of messages, or remove those that have already been addressed or not relevant to your task.

❌ Remove all

Removes all messages from the view.
Results View Contextual Menu Actions

The following actions are available when the contextual menu is invoked in this view:

**Learn Word(s) (Available when spelling errors are reported in the Results view)**

Adds the word(s) to a list of learned words to instruct the spell checker engine to not report the word(s) as spelling errors in the future.

**Show message**

Displays a dialog box with the full error message, which is useful for a long message that does not have enough room to be displayed completely in the view.

**Previous message**

Navigates to the message above the current selection.

**Next message**

Navigates to the message below the current selection.

**Remove selected**

Removes selected messages from the view.

**Remove all**

Removes all messages from the view.

**Copy**

Copies information associated with the selected messages. For example:

- The file path of the document that triggered the output message.
- The path of the main file (on page 3220) (in the case of a validation scenario on page 787), it is the path of the file where the validation starts and can be different from the validated file).
- Error severity (error, warning, info message, etc.)
- Name of validating processor.
- Name of validation scenario (on page 787).
- The line and column in the file that triggered the message.

**Copy Description**

Copies the description values for all selected items. It is possible to assign a shortcut key (on page 296) for this action.

**Select All**

Extends the selection to all the messages from the view.

**Print Results**
Sends the complete list of messages to a printer. For each message, the included details are the same as the ones for the Copy action (on page 551). This action is also available in the Window > Results menu.

Save Results

Saves the complete list of messages in a file in text format. For each message, the included details are the same as the ones for the Copy action (on page 551). This action is also available in the Window > Results menu.

Save Results as XML

Saves the complete list of messages in a file in XML format. For each message, the included details are the same as the ones for the Copy action (on page 551).

Save Results as HTML

Saves the complete list of messages in a file in HTML format. For each message, the included details are the same as the ones for the Copy action (on page 551).

Group by

A set of Group by toggle actions that allow you to group the messages according to a selected criteria so that they can be presented in a hierarchical layout. The criteria used for grouping can be the severity of the errors (error, warning, info message, etc.), the resource name, the description of the message, and so on.

Ungroup all

Removes the grouping rules so that the messages are presented in a continuous list.

Show group columns

If any of the Group by options are selected, you can use this option to show or hide grouping columns.

Restore default grouping

Restores the column size for each column and the grouping rules that were saved in the user preferences the last time when this view was used. If it is the first time this view is used, the action sets an initial default column size for each column and a grouping rule that is appropriate for the type of messages. For example:

• Group the messages by the path of the validated file if there are error messages from a validation action or spelling errors reported by the Check Spelling in Files action (on page 458).

• No grouping rule for the results of applying an XPath expression (on page 1999).

Expand All

Expands all the nodes of the tree, which is useful when the messages are presented in a hierarchical mode.

Collapse All
Collapses all the nodes of the tree, which is useful when the messages are presented in a hierarchical mode.

**Making a Persistent Copy of Results**

The Results view *(on page 548)* displays the results from the following operations:

- Document validation *(on page 775)*
- Checking the form of documents *(on page 773)*
- XSLT or FO transformations *(on page 1432)*
- Finding all occurrences of a string in a file *(on page 432)*
- Finding all occurrences of a string in multiple files *(on page 436)*
- Applying an XPath expression to the current document *(on page 2002)*

To make a persistent copy of the Results view *(on page 548)*, use one of these actions:

**File > Save Results**

Displays the Save Results dialog box, used to save the result list of the current message tab. The action is also available on the right-click menu of the Results panel.

**File > Print Results**

Displays the Page Setup dialog box used to define the page size and orientation properties for printing the result list of the current Results panel. The action is also available on the right-click menu of the Results panel.

**Save Results as XML from the contextual menu**

Saves the content of the Results panel in an XML file with the format:

```xml
<Report>
  <Incident>
    <engine>The engine reporting the error.</engine>
    <severity>The severity level</severity>
    <Description>Description of output message.</Description>
    <SystemID>The location of the file linked to the message.</SystemID>
    <Location>
      <start>
        <line>Start line number in file.</line>
        <column>Start column number in file</column>
      </start>
      <end>
        <line>End line number in file.</line>
        <column>End column number in file</column>
      </end>
    </Location>
  </Incident>
</Report>
```
Syntax Highlighting in XML Documents

Oxygen XML Editor supports syntax highlighting in Text mode to make it easier to read the semantics of the structured content by displaying each type of code in different colors and fonts.

To customize the colors or styles used for the syntax highlighting colors for XML files, follow these steps:

1. Open the Preferences dialog box (Options > Preferences) (on page 124).
2. Go to Editor > Syntax Highlight (on page 224).
3. Select and expand the XML section in the top pane.
4. Select the component you want to change and customize the colors or styles using the selectors to the right of the pane.
5. Select the XML tab in the Preview pane to see the effects of your changes.

Tip:
Oxygen XML Editor also allows you to specify syntax highlighting colors for specific XML elements and attributes with specific namespace prefixes. This can be done in the Editor > Syntax Highlight > Elements/Attributes by Prefix preferences page (on page 225).

Syntax Highlight Depending on Namespace Prefix

The syntax highlight scheme of an XML file type (on page 224) allows the configuration of a color per each type of token that can appear in an XML file. Distinguishing between the XML tag tokens based on the namespace prefix brings additional visual help in editing some XML file types. For example, in XSLT stylesheets, elements from various namespaces (such as XSLT, XHTML, XSL:FO, or XForms) are inserted in the same document and the editor panel can become cluttered. Marking tags with different colors based on the namespace prefix (on page 225) allows easier identification of the tags.
Figure 122. Example of Coloring XML Tags by Prefix

```xml
3  <xsl:template match="name">
4     <fo:list-item>
5         <fo:list-item-label end-indent="label-end0">
6             <fo:block text-align="end" font-weight="bold">Full Name</fo:block>
7             <fo:list-item-label>
8         </fo:list-item-body start-indent="body-start0">
9             <xsl:apply-templates select="*"/>
10        </fo:list-item-body>
11    </fo:list-item>
12  </xsl:template>
```

Related Information:
Changing the colors displayed in the Text Mode Editor (on page 224)

Formatting and Indenting XML Documents

In Text mode (on page 353), you can decide how the XML file is formatted and indented. In the other modes, and when you switch between modes, Oxygen XML Editor automatically formats and indents the XML.

You can trigger a format and indent operation for your XML document (in Text mode) using one of the following actions:

- Format and Indent toolbar button - Formats and indents the current document.
- Document > Source > Format and Indent - Formats and indents the whole document.
- Document > Source > Indent Selection - Indents the current selection (but does not add line breaks). This action is also available in the Source submenu of the contextual menu.
- Document > Source > Format and Indent Element - Formats and indents the current element (the inmost nested element that currently contains the cursor) and its child-elements. This action is also available in the Source submenu of the contextual menu.

Various settings affect how Oxygen XML Editor formats and indents XML. Many of these settings have to do with how whitespace is handled.

Significant and Insignificant Whitespace in XML

XML documents are text files that describe complex documents. Some of the white space (spaces, tabs, line feeds, etc.) in the XML document belongs to the document it describes (such as the space between words in a paragraph) and some of it belongs to the XML document (such as a line break between two XML elements). Whitespace belonging to the XML file is called insignificant whitespace. The meaning of the XML would be the same if the insignificant whitespace were removed. Whitespace belonging to the document being described is called significant whitespace.

Knowing when whitespace is significant or insignificant is not always easy. For instance, a paragraph in an XML document might be laid out like this:
NO Free man shall be taken or imprisoned, or be stripped of his Freedom, or Liberties, or free Customs, or be outlawed, or exiled, or any otherwise destroyed; nor will we not pass upon him, nor condemn him, but by lawful judgment of his Peers, or by the Law of the land. We will sell to no man, we will not deny to any man either Justice or Right.

By default, XML considers a single whitespace between words to be significant, and all other whitespace to be insignificant. The paragraph above could have been written on one line because the XML parser would see it as exactly the same paragraph since all multiple consecutive whitespaces will be replaced with a single whitespace. Removing the insignificant space in markup like this is called normalizing space.

In some cases, all the spaces inside an element should be treated as significant. For example, in a code sample:

```xml
<codeblock>
class HelloWorld
{
    public static void main(String args[])
    {
        System.out.println("Hello World");
    }
}
</codeblock>
```

Here every whitespace character between the `</codeblock>` tags should be treated as significant.

**How Oxygen XML Editor Determines When Whitespace is Significant**

When Oxygen XML Editor formats and indents an XML document, it introduces or removes insignificant whitespace to produce a layout with reasonable line lengths and elements indented to show their place in the hierarchy of the document. To correctly format and indent the XML source, Oxygen XML Editor needs to know when to treat whitespace as significant and when to treat it as insignificant. However it is not always possible to tell this from the XML source file alone. To determine what whitespace is significant, Oxygen XML Editor assigns each element in the document to one of four categories:

- **Ignore space**
  
  In the ignore space category, all whitespace is considered insignificant. This generally applies to content that consists only of elements nested inside other elements, with no text content.

- **Normalize space**
  
  In the normalize space category, a single whitespace character between character strings is considered significant and all other spaces are considered insignificant. Therefore, all
consecutive whitespaces will be replaced with a single space. This generally applies to elements that contain text content only.

**Mixed content**

In the mixed content category, a single whitespace between text characters is considered significant and all other spaces are considered insignificant.

**Notes:**

- Whitespace between two child elements embedded in the text is normalized to a single space (rather than to zero spaces as would normally be the case for a text node with only whitespace characters, or the space between elements generally).
- The lack of whitespace between a child element embedded in the text and either adjacent text or another child element is considered significant. That is, no whitespace can be introduced here when formatting and indenting the file.

For example:

```xml
<p>The file is located in <i>HOME</i>/USER/hello. This is a <strong>big</strong> deal.</p>
```

In this example, whitespace should not be introduced around the i tags as it would introduce extra significant whitespace into the document. The space between the end <strong>tag and the beginning <emphasis>tag should be normalized to a single space, not zero spaces.

**Preserve space**

In the preserve space category, all whitespace in the element is regarded as significant. No changes are made to the spaces in elements in this category. However, child elements may be in another category, and may be treated differently.

Attribute values are always in the preserve space category. The spaces between attributes in an element tag are always in the default space category.

Oxygen XML Editor evaluates several pieces of information to assign an element to one of these categories. An element is always assigned to the most restrictive category (from Ignore to Preserve) that it is assigned to by any of the sources Oxygen XML Editor consults. For instance, if the element is named on the Default elements list (as described below) but it has an `@xml:space="preserve"` attribute in the source file, it will be assigned to the preserve space category. If an element has the `@xml:space="default"` attribute in the source, but is listed on the Mixed content elements list, it will be assigned to the mixed content category.

To assign elements to these categories, Oxygen XML Editor consults information from the following sources:
xml:space

If the XML element contains the `@xml:space` attribute, the element is promoted to the appropriate category based on the value of the attribute.

CSS whitespace property

If the CSS stylesheet controlling the Author mode editor applies the `whitespace: pre` setting to an element, it is promoted to the preserve space category.

CSS display property

If a text node contains only white-spaces:

- If the node has a parent element with the CSS `display` property set to `inline` then the node is promoted to the mixed content category.
- If the left or right sibling is an element with the CSS `display` property set to `inline` then the node is promoted to the mixed content category.
- If one of its ancestors is an element with the CSS `display` property set to `table` then the node is assigned to the ignore space category.

Schema-aware formatting

If a schema is available for the XML document, Oxygen XML Editor can use information from the schema to promote the element to the appropriate category. For example:

- If the schema declares an element to be of type `xs:string`, the element will be promoted to the preserve space category because the string built-in type has the whitespace facet with the value preserve.
- If the schema declares an element to be mixed content, it will be promoted to the mixed content category.

Schema-aware formatting can be turned on and off.

- To turn it on or off for Author mode, open the Preferences dialog box (Options > Preferences) (on page 124), go to Editor > Edit modes > Author > Schema-Aware, and select/deselect the Schema-aware normalization, format and indent option (on page 180).
- To turn it on or off for the Text editing mode, open the Preferences dialog box (Options > Preferences) (on page 124), go to Editor > Format > XML, and select/deselect the Schema-aware format and indent option (on page 208).

Preserve space elements list

If an element is listed in the Preserve space tab of the Element Spacing list (on page 207) in the XML formatting preferences (on page 205), it is promoted to the preserve space category.

Default space elements list
If an element is listed in the **Default space** tab of the **Element Spacing** list (on page 207) in the XML formatting preferences (on page 205), it is promoted to the default space category.

**Mixed content elements list**

If an element is listed in the **Mixed content** tab of the **Element Spacing** list (on page 207) in the XML formatting preferences (on page 205), it is promoted to the mixed content category.

**Element content**

If an element contains mixed content, that is, a mix of text and other elements, it is promoted to the mixed content category. (Note that, in accordance with these rules, this happens even if the schema declares the element to have element only content.)

If an element contains text content, it is promoted to the default space category.

**Text node content**

If a text node contains any non-whitespace characters then the text node is promoted to the normalize space category.

**Exception to the Rule**

In general, an element can only be promoted to a more restrictive category (one that treats more whitespace as significant). However, there is one exception. In **Author** mode, if an element is marked as mixed content in the schema, but the actual element contains no text content, it can be demoted to the space ignore category if all of its child elements are displayed as **blocks** by the associated CSS (that is, they have a CSS property of display: block). For example, in some schemas, a section or a table entry can be defined as having mixed content but in many cases they contain only **block elements** (on page 3216). In these cases, any whitespace they contain cannot be significant and they can be treated as space ignore elements. This exception can be turned on or off using the **Schema-Aware Editing** option (on page 181) in the **Schema-Aware** preferences page.

**How Oxygen XML Editor formats and indents XML**

You can control how Oxygen XML Editor formats and indents XML documents. This can be particularly important if you store your XML document in a version control system, as it allows you to limit the number of trivial changes in spacing between versions of an XML document. The following preference pages include options that control how XML documents are formatted:

- **Format preferences page** (on page 202)
- **XML Formatting preferences page** (on page 205)
- **Whitespaces preferences page** (on page 209)

**When Oxygen XML Editor formats and indents XML**

Oxygen XML Editor formats and indents a document, or part of it, on the following occasions:
• In Text mode when you select one of the format and indent actions (Document > Source > Format and Indent, Document > Source > Indent Selection, or Document > Source > Format and Indent Element).
• When saving documents in Author mode.
• When switching from Author mode to another mode.
• When saving documents in Design mode.
• When switching from Design mode to another mode.
• When saving or switching to Text mode from Grid mode, if the Format and indent when passing from grid to text or on save option (on page 174) is selected in the Grid preferences page.

Setting an Indent Size to Zero

Oxygen XML Editor will automatically format and indent (on page 555) documents at certain times. This includes indenting the content from the margin to reflect its structure. In some cases, you may not want your content indented. To avoid your content being indented, you can set an indent size of zero.

Note:
Changing the indent size does not override the rules that Oxygen XML Editor uses for handling whitespace when formatting and indenting XML documents. Therefore, changing the indent size will have no effect on elements that require whitespaces to be maintained.

There are two cases to consider.

Maintaining zero indent in documents with zero indent

If you have existing documents with zero indent and you want Oxygen XML Editor to maintain a zero indent when editing or formatting those documents:

1. Open the Preferences dialog box (Options > Preferences) (on page 124) and go to Editor > Format (on page 202).
2. Select Detect indent on open.
3. Select Use zero-indent if detected.

Oxygen XML Editor will examine the indent of each document as it is opened and if the indent is zero for all lines, or for nearly all lines, a zero indent will be used when formatting and indenting the document. Otherwise, Oxygen XML Editor will use the indent closest to what it detects in the document.

Enforcing zero indent for all documents

If you want all documents to be formatted with zero indent, regardless of their current indenting:

1. Open the Preferences dialog box (Options > Preferences) (on page 124) and go to Editor > Format (on page 202).
2. Deselect Detect indent on open.
3. Set Indent size to 0.
All documents will be formatted and indented with an indent of zero.

**Warning:**

Setting the indent size to zero can change the meaning of some file types, such as Python source files.

**Format and Indent (Pretty-Print) Multiple Files**

Oxygen XML Editor provides support for formatting and indenting (*pretty-print* (on page 3222)) multiple files at once. This action is available for any document in XML format, as well as for XQuery, CSS, JavaScript, and JSON documents.

To format and indent multiple files, use the **Format and Indent Files** action that is available in the contextual menu of the **Project view** (on page 402) or from the **Tools** menu. This opens the **Format and Indent Files** dialog box that allows you to configure options for the action.

![Figure 123. Format and Indent Files Dialog Box](image)

The **Scope** section allows you choose from the following scopes:

- **All opened files** - The *pretty-print* (on page 3222) is performed in all opened files.
- **Directory of the current file** - All the files in the folder of the currently edited file.
- **Project files** - All files from the current project.
- **Selected project files** - The selected files from the current project.
- **Specified path** - the *pretty-print* (on page 3222) is performed in the files located at a specified path.

The **Options** section includes the following options:
• **File filter** - Allow you to filter the files from the selected scope.

• **Recurse subdirectories** - When selected, the *pretty-print (on page 3222)* is performed recursively for the specified scope. The one exception is that this option is ignored if the scope is set to All opened files.

• **Include hidden files** - When selected, the *pretty-print (on page 3222)* is also performed in the hidden files.

• **Make backup files with extension** - When selected, Oxygen XML Editor makes backup files of the modified files. The default extension is `.bak`, but you can change the extension as you prefer.

**Managing Highlighted Content**

While working with XML documents you often have frequent changes to the structure and content. You are often faced with a situation where you need to make a slight change in multiple places in the same document. Oxygen XML Editor includes a feature, *Manage Highlighted Content*, that is designed to help you achieve this.

When you are in **Text** mode and you perform a search operation or apply an XPath that highlights multiple results, you can access the *Manage Highlighted Content* submenu by right-clicking any of the highlights in the editing pane. If the results are displayed only in the **Results** panel at the bottom of the screen, you can use the ![Highlight all results in editor](note) button (on the right side of the **Results** panel) to display all the highlights in the editor (then you can access the *Manage Highlighted Content* submenu from the contextual menu of any highlight.

The following options are available in the *Manage Highlighted Content* submenu:

**Modify All**

Use this option to modify (in-place) all the occurrences of the selected content. When you use this option, a thin rectangle replaces the highlights and allows you to start editing. If matches with different letter cases are found, a dialog box is displayed that allows you select whether you want to modify only matches with the same letter case or all matches.

**Note:**

If you select a very large number of highlights that you want to modify using this feature, a dialog box informs you that you may experience performance issues. You have the option to either use the *Find/Replace operation (on page 432)*, or continue the operation.

**Surround All**

Use this option to surround the highlighted content with a specific tag. This option opens the **Tag** dialog box. The **Specify the tag** drop-down menu presents all the available elements that you can choose from.

**Remove All**

Removes all the highlighted content.
If you right-click content in another part of the document, other than a highlight, you have the option to select the following option:

**Modify All Matches**

Use this option to modify (in-place) all the occurrences of the selected content (or the contiguous fragment where the cursor is located). When you use this option, a thin rectangle replaces the highlights and allows you to start editing. If matches with different letter cases are found, a dialog box is displayed that allows you select whether you want to modify only matches with the same letter case or all matches.

### Adjusting the Transparency of XML Markup

Most of the time you want the content of a document displayed on screen with zero transparency. However, if you want to focus your attention only on editing text content inside XML markup, Oxygen XML Editor offers the option of reducing the visibility of the markup by increasing their transparency when displayed in Text mode. To change the level of transparency, use the Tags Transparency Selector drop-down menu that is available from the Source toolbar. By default, this drop-down menu is not visible. You can add it to the toolbar by using the Configure Toolbars action *(on page 365)*. There are several levels of transparency that can be adjusted to make the content more or less visible:

- **Normal Contrast** - Resets the transparency level back to normal.
- **Semi-transparent Text** - Slightly reduces the visibility of text to place greater emphasis on the visibility of the XML markup.
- **Transparent Text** - Greatly reduces the visibility of text to place even greater emphasis on the visibility of the XML markup.
- **Semi-transparent Markup** - Slightly reduces the visibility of the XML markup to place greater emphasis on the visibility of the text.
- **Transparent Markup** - Greatly reduces the visibility of the XML markup to place even greater emphasis on the visibility of the text.

*Figure 124. Tags Transparency Selector*
Locking and Unlocking XML Markup

For documents with fixed markup, such as forms that do not allow the XML tags to be modified (only their text content), the possibility to edit the XML tag names can be toggled on or off with the Lock / Unlock the XML tags action available in Text editing mode from the Source submenu from the contextual menu (or Document > Source menu).

You can set the default lock state for all opened editors using the Lock the XML tags option in the Text preferences page (on page 173).

Special Character Support in Text Mode

If bidirectional text, such as Arabic or Hebrew languages, certain Asian languages (such as Devanagari, Bengali, Gurmukhi, Gujarati, Oriya, Tamil, Telugu, Kannada, Malayalam, Sinhala, Thai, Khmer), or other special characters (such as combining characters) are detected in a document, Oxygen XML Editor displays a Special Characters Detected dialog box that prompts you to Enable or Disable support for these special characters (you can also enable or disable the support for special characters in the Open preferences page (on page 200).

Enabled

If you choose to enable support for special characters and as long as you chose a font (on page 133) that supports the particular special characters, this means that the glyphs will be rendered properly in Text mode and the cursor navigation mechanism will recognize them as they are shown.

Example: The ALLERY glyph could be inserted using a consecutive combination of two characters (U+00C2 followed by U+0323). With the special characters support enabled and the SansSerif font chosen, that glyph will be rendered properly (a capital letter A with a circumflex above it and a dot below) and you can navigate through the glyph in one step (pressing the right/left arrow key once).

Restriction:
When support for special characters is enabled, the folding support (on page 528) is not available.

Disabled

If you choose to disable support for special characters, it may affect text rendering, cursor navigation, and text management operations. However, this is helpful if you need to open very large documents (on page 469) since disabling the bidirectional editing support can enhance performance.

Example: The ALLERY glyph could be inserted using a consecutive combination of two characters (U+00C2 followed by U+0323). With the special characters support disabled, that glyph may or may not be rendered properly and when navigating through the glyph, it would take two steps (pressing the right/left arrow key twice).
Restriction:
Bidirectional content in the Text mode cannot be rendered using Bold or Italic.

Related Information:
Special Character Support in Author Mode (on page 752)
Special Character Support in Grid Mode (on page 586)
Inserting Special Characters with the Character Map (on page 465)

Inserting or Opening a File at Cursor Location

When editing content in Text mode, the following actions (with regard to inserting, opening, or comparing files) are available in the Document > File menu:

**Insert File**

Inserts the content of the file with the specified file path into the current document at the current position of the cursor.

**Open File at Cursor**

Opens the file at the cursor position in a new panel. If the file path represents a directory path, it will be opened in system file browser. If the file at the specified location does not exist, an error dialog box is displayed and it includes a Create new file button that starts the New document wizard. This allows you to choose the type or the template for the file. If the action succeeds, the file is created with the referenced location and name and is opened in a new editor panel. If the file is an image file, it will be opened in the Image Preview pane (on page 469).

**Open File at Cursor in System Application**

Opens the file (identified by its link) or web page (identified by a web link) found at the cursor position. The target is opened in the default system application associated with that file type.

**Compare**

Opens the current file in the Compare Files tool (on page 474).

**Ctrl + Single-Click (Command + Single-Click on macOS)**

Use this shortcut to open any of the following:

- Any absolute URL (URLs that have a protocol), regardless of their location in the document.
- URI attributes such as: `@schemaLocation`, `@noNamespaceSchemaLocation`, `@href` and others.
- Open the target for DITA references (such as a `@conref`, `@conkeyref`, `@keyref`, and more).
- Processing instructions used for associating resources, xml-models, xml-stylesheets.
Quick Assist Support for IDs and IDREFS

The Quick Assist support (on page 3223) is activated automatically when you place the cursor inside an ID or IDREF in Text mode. To access it, click the yellow bulb help marker placed on the current line, in the line number stripe of the editor. You can also invoke the Quick Assist menu from the contextual menu or by pressing Alt+1 (Command+Alt+1 on macOS) on your keyboard.

The following actions are available:

Rename
Renames the ID and all its occurrences. Selecting this action opens the Rename XML ID dialog box. This dialog box lets you insert the new ID value and choose the scope of the rename operation (on page 831). For a preview of the changes you are about to make, click Preview. This opens the Preview dialog box, which presents a list with the files that contain changes and a preview zone of these changes.

Search Declarations
Searches for the declaration of the ID reference. By default, the scope of this action is the current project. If you configure a scope using the Select the scope for the Search and Refactor operations (on page 831) dialog box, this scope will be used instead.

Search References
Searches for the references of the ID. By default, the scope of this action is the current project. If you configure a scope using the Select the scope for the Search and Refactor operations (on page 831) dialog box, this scope will be used instead.

Change scope
Opens the Select the scope for the Search and Refactor operations (on page 831) dialog box.

Rename in File
Renames the ID you are editing and all its occurrences from the current file.

Search Occurrences
Searches for the declaration an references of the ID located at the cursor position in the current document.

Related Information:
Modular Contextual XML Editing Using 'Main Files' Support (on page 829)

Highlight ID Occurrences in Text Mode

To see the occurrences of an ID in an XML document in the Text mode, place the cursor inside the ID declaration or reference. The occurrences are marked in the vertical side bar at the right of the editor. Click a marker on the side bar to jump to the occurrence that it corresponds to. The occurrences are also highlighted in the editing area.
Note:

Highlighted ID declarations are rendered with a different color than highlighted ID references. To customize these colors or disable this feature, open the Preferences dialog box (Options > Preferences) (on page 124) and go to Editor > Mark Occurrences (on page 225).

Related Information:

Modular Contextual XML Editing Using 'Main Files' Support (on page 829)

Contextual Menu Actions in Text Mode

When editing XML documents in Text mode, Oxygen XML Editor provides the following actions in the contextual menu (many of them also appear in the submenus of the Document menu):

Add File to Review Task

This action can be used to add the current document to a task in the Content Fusion Tasks Manager view. Oxygen Content Fusion is a flexible, intuitive collaboration platform designed to adapt to any type of documentation review workflow. This functionality is available through a pre-installed connector add-on (on page 2527). To fully take advantage of all of the benefits and features of Content Fusion, your organization will need an Oxygen Content Fusion Enterprise Server. For more information, see the Oxygen Content Fusion website.

Cut, Copy, Paste

Executes the typical editing actions on the currently selected content.

Copy XPath

Copies the XPath expression of the current element or attribute (or property for JSON documents) to the clipboard.

Toggle Line Wrap (Ctrl + Shift + Y (Command + Shift + Y on macOS))

Enables or disables line wrapping. When enabled, if text exceeds the width of the displayed editor, content is wrapped so that you do not have to scroll horizontally.

Toggle Comment (Ctrl + Shift + Comma (Command + Shift + M on macOS))

Comments the current selection of the current editor. If the selection already contains a comment the action removes the comment from around the selection. If there is no selection in the current editor and the cursor is not positioned inside a comment the current line is commented. If the cursor is positioned inside a comment then the commented text is uncommented.

Go to submenu

This submenu includes the following actions:

Go to Matching Tag (Ctrl + Shift + G (Command + Shift + G on macOS))
Moves the cursor to the end tag that matches the start tag, or vice versa.

**Go after Next Tag (Ctrl + CloseBracket (Command + CloseBracket on macOS))**

Moves the cursor to the end of the next tag.

**Go after Previous Tag (Ctrl + OpenBracket (Command + OpenBracket on macOS))**

Moves the cursor to the end of the previous tag.

**Select submenu**

This submenu allows you to select the following:

**Element**

Selects the entire element at the current cursor position.

**Content**

Selects the entire content of the element at the current cursor position, excluding the start and end tag. Performing this action repeatedly will result in the selection of the content of the ancestor of the currently selected element content.

**Attributes**

Selects all the attributes of the element at the current cursor position.

**Parent**

Selects the parent element at the current cursor position.

**Source submenu**

This submenu includes the following actions:

- **Shift Right (Tab)**
  Shifts the currently selected block to the right.

- **Shift Left (Shift + Tab)**
  Shifts the currently selected block to the left.

- **Indent selection (Ctrl + I (Command + I on macOS))**
  Corrects the indentation of the selected block of lines if it does not follow the current indenting preferences (on page 202).

- **Escape Selection**
  Escapes a range of characters by replacing them with the corresponding character entities.

- **Unescape Selection**
  Replaces the character entities with the corresponding characters.

- **Format and Indent Element (Ctrl + Shift + I (Command + Shift + I on macOS))**
Pretty-prints (on page 3222) the element that surrounds the current cursor position.

To Upper Case

Converts the selected content to upper case characters. This works with contiguous and multiple selections.

To Lower Case

Converts the content selection to lower case characters. This works with contiguous and multiple selections.

Capitalize Lines

It capitalizes the first letter found on every new line that is selected. Only the first letter is affected, the rest of the line remains the same. If the first character on the new line is not a letter then no changes are made.

Convert Hexadecimal Sequence to Character (Ctrl + Shift + X (Command + Shift + X on macOS))

Converts a sequence of hexadecimal characters to the corresponding Unicode character (on page 463). The action can be invoked if there is a selection containing a valid hexadecimal sequence or if the cursor is placed at the right side of a valid hexadecimal sequence. A valid hexadecimal sequence can be composed of 2 to 4 hexadecimal characters and may or may not be preceded by the 0x or 0X prefix. Examples of valid sequences and the characters they will be converted to:

- 0x0045 will be converted to E
- 0x0125 to ĥ
- 265 to ɥ
- 2190 to ←

Note:

For more information about finding the hexadecimal value of a character, see Finding the Decimal, Hexadecimal, or Character Entity Equivalent (on page 466).

Base64 Encode/Decode submenu

This submenu include the following actions for encoding or decoding base 64 schemes:

Import File to Encode and Insert

Encodes a file and then inserts the encoded content into the current document at the cursor position.

Decode Selection and Export to File
Decodes a selection of text from the current document and then exports (saves) the result to another file.

**Encode Selection**

Replaces a selection of text with the result of encoding that selection. By default, a dialog box is displayed that allows you to select the encoding to use. There is an option to choose to not show this dialog box in the future. In this case, the encoding that is specified in the **Encoding for Base64, Base32, Hex conversions** option in the **Encoding preferences page** (on page 169) will be used. Likewise, the same is true if the **Show the dialog box for choosing the encoding for Base64, Base32, Hex conversions** option is not selected in the **Messages preference page** (on page 308).

**Decode Selection**

Replaces a selection of text with the result of decoding that selection. By default, a dialog box is displayed that allows you to select the encoding to use. There is an option to choose to not show this dialog box in the future. In this case, the encoding that is specified in the **Encoding for Base64, Base32, Hex conversions** option in the **Encoding preferences page** (on page 169) will be used. Likewise, the same is true if the **Show the dialog box for choosing the encoding for Base64, Base32, Hex conversions** option is not selected in the **Messages preference page** (on page 308).

**Modify All Matches**

Use this option to modify (in-place) all the occurrences of the selected content (or the contiguous fragment where the cursor is located). When you use this option, a thin rectangle replaces the highlights and allows you to start editing. If matches with different letter cases are found, a dialog box is displayed that allows you to select whether you want to modify only matches with the same letter case or all matches.

**Base32 Encode/Decode submenu**

This submenu include the following actions for encoding or decoding **base32** schemes:

- **Import File to Encode and Insert**
  
  Encodes a file and then inserts the encoded content into the current document at the cursor position.

- **Decode Selection and Export to File**
Decodes a selection of text from the current document and then exports (saves) the result to another file.

**Encode Selection**

Replaces a selection of text with the result of encoding that selection. By default, a dialog box is displayed that allows you to select the encoding to use. There is an option to choose to not show this dialog box in the future. In this case, the encoding that is specified in the **Encoding for Base64, Base32, Hex conversions** option in the **Encoding preferences page (on page 169)** will be used. Likewise, the same is true if the **Show the dialog box for choosing the encoding for Base64, Base 32, Hex conversions** option is not selected in the **Messages preference page (on page 308)**.

**Decode Selection**

Replaces a selection of text with the result of decoding that selection. By default, a dialog box is displayed that allows you to select the encoding to use. There is an option to choose to not show this dialog box in the future. In this case, the encoding that is specified in the **Encoding for Base64, Base32, Hex conversions** option in the **Encoding preferences page (on page 169)** will be used. Likewise, the same is true if the **Show the dialog box for choosing the encoding for Base64, Base 32, Hex conversions** option is not selected in the **Messages preference page (on page 308)**.

**Modify All Matches**

Use this option to modify (in-place) all the occurrences of the selected content (or the contiguous fragment where the cursor is located). When you use this option, a thin rectangle replaces the highlights and allows you to start editing. If matches with different letter cases are found, a dialog box is displayed that allows you select whether you want to modify only matches with the same letter case or all matches.

**Hex Encode/Decode submenu**

This submenu include the following actions for encoding or decoding **hex** schemes:

**Import File to Encode and Insert**

Encodes a file and then inserts the encoded content into the current document at the cursor position.

**Decode Selection and Export to File**
Decodes a selection of text from the current document and then exports (saves) the result to another file.

**Encode Selection**

Replaces a selection of text with the result of encoding that selection. By default, a dialog box is displayed that allows you to select the encoding to use. There is an option to choose to not show this dialog box in the future. In this case, the encoding that is specified in the **Encoding for Base64, Base32, Hex conversions** option in the **Encoding preferences page** (on page 169) will be used. Likewise, the same is true if the **Show the dialog box for choosing the encoding for Base64, Base32, Hex conversions** option is not selected in the **Messages preference page** (on page 308).

**Decode Selection**

Replaces a selection of text with the result of decoding that selection. By default, a dialog box is displayed that allows you to select the encoding to use. There is an option to choose to not show this dialog box in the future. In this case, the encoding that is specified in the **Encoding for Base64, Base32, Hex conversions** option in the **Encoding preferences page** (on page 169) will be used. Likewise, the same is true if the **Show the dialog box for choosing the encoding for Base64, Base32, Hex conversions** option is not selected in the **Messages preference page** (on page 308).

**Modify All Matches**

Use this option to modify (in-place) all the occurrences of the selected content (or the contiguous fragment where the cursor is located). When you use this option, a thin rectangle replaces the highlights and allows you to start editing. If matches with different letter cases are found, a dialog box is displayed that allows you select whether you want to modify only matches with the same letter case or all matches.

**Join and Normalize Lines** *(Ctrl + J (Command + J on macOS))*

For the current selection, this action joins the lines by replacing the *line separator* with a single space character. It also normalizes the whitespaces by replacing a sequence of such characters with a single space.

**Insert new line after** *(Ctrl + Alt + Enter (Command + Alt + Enter on macOS))*

This action has the same result as moving the cursor to the end of the current line and pressing the *ENTER* key.

**Insert XInclude**
Displays a dialog box that allows you to browse and select the content to be included and automatically generates the corresponding XInclude instruction.

Note:
In the Author mode, this dialog box presents a preview of the inserted document as an author page in the Preview tab and as a text page in the Source tab. In the Text mode, the Source tab is presented.

& Import entities list
Displays a dialog box that allows you to select a list of files as sources for external DTD entities. The internal subset of the DOCTYPE declaration of your document will be updated with the chosen entities. For instance, choosing the files chapter1.xml and chapter2.xml inserts the following section in the DOCTYPE:

```xml
<!ENTITY chapter1 SYSTEM "chapter1.xml">
<!ENTITY chapter2 SYSTEM "chapter2.xml">
```

Lock / Unlock the XML Tags
Disables or enables the ability to edit XML tags.

Canonicalize
Opens the Canonicalize dialog box that allows you to select a canonicalization (on page 3217) algorithm to standardize the format of the document.

Sign
Opens the Sign dialog box that allows you to configure a digital signature for the document.

Verify Signature
Allows you to specify the location of a file to verify its digital signature.

Manage Highlighted Content submenu
This submenu is available from the contextual menu when it is invoked from a highlight after you perform a search operation or apply an XPath expression that highlights more than one result. The following options are available in this submenu:

Modify All
Allows you to modify (in-place) all the occurrences of the selected content. A thin rectangle replaces the highlights and allows you to start editing. If matches with different letter cases are found, a dialog box is displayed that allows you select whether you want to modify only matches with the same letter case or all matches.

Surround All
Surround the highlighted content with a specific tag. This option opens the 
Tag dialog box. The Specify the tag drop-down menu presents all the available 
elements that you can choose from.

Remove All

Removes all the highlighted content.

Modify All Matches

Use this option to modify (in-place) all the occurrences of the selected content (or the contiguous fragment where the cursor is located). When you use this option, a thin rectangle replaces the highlights and allows you to start editing. If matches with different letter cases are found, a dialog box is displayed that allows you select whether you want to modify only matches with the same letter case or all matches.

Go to Definition (Ctrl + Shift + Enter)

Navigates to the definition of the current element or attribute in the schema (DTD, XML Schema, Relax NG schema) associated with the edited XML document. If the current attribute is a "type" belonging to the "http://www.w3.org/2001/XMLSchema-instance" namespace, the cursor is moved in the XML schema to the definition of the type referenced in the value of the attribute. For JSON documents, it navigates to the definition of the current JSON property in the associated JSON Schema.

Refactoring submenu

This submenu includes the following actions:

 Rename Element

The element from the cursor position, and any elements with the same name, can be renamed according with the options from the Rename dialog box.

 Rename Prefix (Alt + Shift + P (Command + Shift + P on macOS))

The prefix of the element from the cursor position, and any elements with the same prefix, can be renamed according with the options from the Rename dialog box.

- If you select the Rename current element prefix option, the application will recursively traverse the current element and all its children. For example, to change the xmlns:p1="ns1" association in the current element to xmlns:p5="ns1", if the xmlns:p1="ns1" association is applied on the parent element, then Oxygen XML Editor will introduce xmlns:p5="ns1" as a new declaration in the current element and will change the prefix from p1 to p5. If p5 is already associated with another namespace in the current element, then the conflict will be displayed in a dialog box. By pressing OK, the prefix is modified from p1 to p5 without inserting a new declaration.
If you select the **Rename current prefix in all document** option, the application will apply the change on the entire document.

To also apply the action inside attribute values, select the **Rename also attribute values that start with the same prefix** checkbox.

**Surround with submenu**

Presents a drop-down menu that allows you to choose a tag to surround a selected portion of content.

**Surround with Tags (Ctrl + E (Command + E on macOS))**

Allows you to choose a tag that encloses a selected portion of content. If there is no selection, the start and end tags are inserted at the cursor position.

- If the **Position cursor between tags** option *(on page 212)* is selected in the **Content Completion** preferences page, the cursor is placed between the start and end tag.
- If the **Position cursor between tags** option *(on page 212)* is not selected in the **Content Completion** preferences page, the cursor is placed at the end of the start tag, in an insert-attribute position.

**Surround with ‘[tag]’ (Ctrl + ForwardSlash (Command + ForwardSlash on macOS))**

Surround the selected content with the last tag used.

**Delete element tags (Alt + Shift + X (Command + Alt + X on macOS))**

Deletes the start and end tag of the current element.

**Split element (Alt + Shift + D (Ctrl + Alt + D on macOS))**

Split the element from the cursor position into two identical elements. The cursor must be inside the element.

**Join elements (Alt + Shift + J (Command + Alt + J on macOS))**

Joins the left and right elements relative to the current cursor position. The elements must have the same name, attributes, and attributes values.

**Attributes Refactoring Actions**

Contains built-in XML refactoring operations that pertain to attributes with some of the information preconfigured based upon the current context.

**Add/Change attribute**

Allows you to change the value of an attribute or insert a new one.

**Convert attribute to element**

Allows you to change an attribute into an element.

**Delete attribute**
Allows you to remove one or more attributes.

**Rename attribute**

Allows you to rename an attribute.

**Replace in attribute value**

Allows you to search for a text fragment inside an attribute value and change the fragment to a new value.

**Comments Refactoring Actions**

Contains built-in XML refactoring operations that pertain to comments with some of the information preconfigured based upon the current context.

**Delete comments**

Allows you to delete comments found inside one or more elements.

**Elements Refactoring Actions**

Contains built-in XML refactoring operations that pertain to elements with some of the information preconfigured based upon the current context.

**Delete element**

Allows you to delete elements.

**Delete element content**

Allows you to delete the content of elements.

**Insert element**

Allows you to insert new elements.

**Rename element**

Allows you to rename elements.

**Unwrap element**

Allows you to remove the surrounding tags of elements, while keeping the content unchanged.

**Wrap element**

Allows you to surround elements with element tags.

**Wrap element content**

Allows you to surround the content of elements with element tags.

**Fragments Refactoring Actions**

Contains built-in XML refactoring operations that pertain to XML fragments with some of the information preconfigured based upon the current context.

**Insert XML fragment**
Oxygen XML Editor 24.1 | 8 - Editing Documents | 577

Allows you to insert an XML fragment.

**Replace element content with XML fragment**

Allows you to replace the content of elements with an XML fragment.

**Replace element with XML fragment**

Allows you to replace elements with an XML fragment.

**Manage IDs submenu**

This submenu is available for XML documents that have an associated DTD, XML Schema, or Relax NG schema (not available for DITA). It includes the following actions:

- **Rename in**
  
  Renames the ID and all its occurrences. Selecting this action opens the Rename XML ID dialog box. This dialog box lets you insert the new ID value and choose the scope of the rename operation (on page 831). For a preview of the changes you are about to make, click Preview. This opens the Preview dialog box, which presents a list with the files that contain changes and a preview zone of these changes.

- **Rename in File**
  
  Renames the ID you are editing and all its occurrences from the current file.

- **Search References**
  
  Searches for the references of the ID. By default, the scope of this action is the current project. If you configure a scope using the Select the scope for the Search and Refactor operations (on page 831) dialog box, this scope will be used instead.

- **Search References in**
  
  Searches for the references of the ID. Selecting this action opens the Select the scope for the Search and Refactor operations (on page 831).

- **Search Declarations**
  
  Searches for the declaration of the ID reference. By default, the scope of this action is the current project. If you configure a scope using the Select the scope for the Search and Refactor operations (on page 831) dialog box, this scope will be used instead.

- **Search Declarations in**
  
  Searches for the declaration of the ID reference. Selecting this action opens the Select the scope for the Search and Refactor operations (on page 831).

- **Search Occurrences in file**
  
  Searches for the declaration and references of the ID in the current document.
Quick Assist (Alt + 1 (Command + Alt + 1 on macOS))

Available when the cursor is inside an ID or IDREF, this action opens the Quick Assist (on page 3223) window that allows you to select some search and refactoring actions for the selected ID or IDREF.

Open submenu

The following actions are available in this submenu:

**Open File at Cursor**

Opens the file at the cursor position in a new panel. If the file path represents a directory path, it will be opened in system file browser. If the file at the specified location does not exist, an error dialog box is displayed and it includes a Create new file button that starts the New document wizard. This allows you to choose the type or the template for the file. If the action succeeds, the file is created with the referenced location and name and is opened in a new editor panel. If the file is an image file, it will be opened in the Image Preview pane (on page 469).

**Open File at Cursor in System Application**

Opens the file (identified by its link) or web page (identified by a web link) found at the cursor position. The target is opened in the default system application associated with that file type.

**Compare**

Opens the current file in the Compare Files tool (on page 474).

**Resource Hierarchy**

Opens the Resource Hierarchy/Dependencies view (on page 832) that allows you to see the resource hierarchy for an XML document.

**Resource Dependencies**

Opens the Resource Hierarchy/Dependencies view (on page 832) that allows you to see the resource dependencies for an XML document.

**Editing XML Documents in Grid Mode**

This section includes topics that describe how to work with XML documents in Grid mode, including various features, actions that are available, and much more.

The Grid mode in Oxygen XML Editor displays the XML document as a structured grid of nested tables where the text content can be modified without directly interacting with the XML markup. This is helpful for non-technical users who want to edit text content without modifying the XML markup.

To switch to this mode, select Grid at the bottom of the editing area.
You can easily expand or collapse elements within the table and the document structure can be changed with simple contextual menu actions, drag/drop, or copy/paste operations. The text content can be modified simply by editing the value of cells that contain the text and a useful Content Completion Assistant is also available to help you edit or insert XML elements.

For more information about some of the features available in the Grid editor, watch our video demonstration:

https://www.youtube.com/embed/PoYm2VqisWk

**Layouts: Grid and Tree**

The Grid editor offers two layout modes. The default one is the grid layout. This smart layout detects the recurring elements in the XML document and creates tables having the children (including the attributes) of these elements as columns. This way, it is possible to have tables nested in other tables, reflecting the structure of your document.

![Figure 125. Grid Layout](image)

The other layout mode is tree-like. It does not create any tables and it only presents the structure of the document.

![Figure 126. Tree Layout](image)

To switch between the two modes, select Document > Grid Layout > Grid mode/Tree mode.

**Grid Mode Navigation**

When you first open a document in Grid mode, the content is collapsed. Only the root element and its attributes are displayed. An arrow sign displayed at the left of the node name indicates that this node has child nodes. To display the children, click this arrow sign. To collapse a node, click the reverse arrow sign. 
The expand/collapse actions can also be invoked with the `NumPad+` and `NumPad-` keys, or from the Expand/Collapse submenu of the contextual menu or from Document > Grid Expand/Collapse.

**Expand/Collapse Submenu**

The following actions are available on the Expand/Collapse submenu:

- **Expand All**
  Expands the selection and all its children.

- **Collapse All**
  Collapses the selection and all its children.

**Expand Children**

Expands all the children of the selection but not the selection.

**Collapse Children**

Collapses all the children of the selection but not the selection.

**Collapse Others**

Collapses all the siblings of the current selection but not the selection.

**Keyboard Shortcuts**

A variety of other keyboard shortcuts are also available in Grid mode:

<table>
<thead>
<tr>
<th>Key</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tab</td>
<td>Moves the cursor to the next editable value in a table row.</td>
</tr>
<tr>
<td>Shift + Tab</td>
<td>Moves the cursor to the previous editable value in a table row.</td>
</tr>
<tr>
<td>Enter</td>
<td>Begins editing and lets you insert a new value. Also commits the changes after you finish editing.</td>
</tr>
<tr>
<td>UpArrow/PageUp</td>
<td>Navigates toward the beginning of the document.</td>
</tr>
<tr>
<td>DownArrow/PageDown</td>
<td>Navigates toward the end of the document.</td>
</tr>
<tr>
<td>Shift</td>
<td>Used in conjunction with the navigation keys to create a continuous selection area.</td>
</tr>
<tr>
<td>Ctrl (Command on macOS key)</td>
<td>Used in conjunction with the mouse cursor to create discontinuous selection areas.</td>
</tr>
</tbody>
</table>

The following key combinations can be used to scroll the grid:

- **Ctrl + UpArrow (Command + UpArrow on macOS)** - scrolls the grid upwards.
- **Ctrl + DownArrow (Command + DownArrow on macOS)** - scrolls the grid downwards.
• Ctrl + LeftArrow (Command + LeftArrow on macOS) scrolls the grid to the left.
• Ctrl + RightArrow (Command + RightArrow on macOS) scrolls the grid to the right.

Related Information:
Editing Actions in Grid Mode (on page 581)

Editing Actions in Grid Mode

Since Grid mode presents XML content in a structured grid of nested tables, editing content in this mode can be done with a combination of the Content Completion Assistant (on page 586) and actions that allow you to work with the structure or content of the nested tables much like you would with any table. Oxygen XML Editor provides ways to edit content in the cells of the nested tables or to edit the structure of the tables.

Tip:
There are two different types of layouts available in Grid mode. Most people prefer to leave it on the default Grid mode layout, but there is also a Tree mode layout that presents the structure of the document in more of a vertical tree-like manner. You can switch between the two layouts to see which one works best for you particular situation from the Document > Grid Layout menu.

Expanding/Collapsing Nodes

An arrow sign (↑) displayed at the left of a node indicates that it has child nodes. To display the children, click this arrow sign. To collapse a node, click the reverse arrow sign (↓). The expand/collapse actions can also be invoked with the NumPad+ and NumPad- keys, or from the Expand/Collapse submenu of the contextual menu.

To expand all child nodes, right-click the cell that contains the parent node and select Expand All from the Expand/Collapse submenu. To collapse all node, right-click any cell and select Collapse All from the Expand/Collapse submenu.

Editing Elements or Attributes

To edit elements or attributes in Grid mode, simply double-click the cell that contains the element or attribute (or select the cell and press Enter) to invoke the Content Completion Assistant (on page 586). This opens a pop-up window that offers a list of proposals that are valid for that particular node.

Editing Text Content in Cells

To edit the text value of a cell, simply select the grid cell and press Enter (or double-click the cell), and start editing.

To stop editing a cell value, press Enter again.

To cancel the editing without saving the current changes in the document, press the Esc key.
Editing the Structure of the Nested Tables

To edit the structure of the nested tables in Grid mode, Oxygen XML Editor provides the following actions in the contextual menu (many of them also appear in the submenus of the Document menu, or the toolbar):

- **Cut, Copy, Paste, Delete** common editing actions
  
  Executes the typical editing actions on the currently selected elements. The Cut and Copy operations preserve the styles of the copied content.

- **Paste as Child**
  
  Pastes the copied content as the last child of the current selection.

- **Duplicate**
  
  Creates a new node by duplicating the currently selected one.

- **Insert Before**
  
  Offers a list of valid nodes, depending on the context, and inserts your selection before the currently selected node, as a sibling.

- **Insert After**
  
  Offers a list of valid nodes, depending on the context, and inserts your selection after the currently selected node, as a sibling.

- **Append Child**
  
  Offers a list of valid nodes, depending on the context, and appends your selection as a child of the currently selected node.

- **Sort Ascending, Sort Descending**
  
  The sorting result depends on the data type of the column content. It could be a numerical sorting for numbers or an alphabetical sorting for text information. The editor automatically analyzes the content and decides what type of sorting to apply. When a mixed set of values is present in the sorted column, a dialog box is displayed that allows you to choose the desired type of sorting between numerical and alphabetical.

- **Insert Row**
  
  Inserts a new row below the current selection. To insert a new row, you could also select the row header (the zone to the left of the row that holds the row number) and press Enter.

- **Insert Column**
  
  Inserts a column after the current selection.

- **Clear Content**
  
  Removes all content from the current cell.

- **Expand/Collapse > Expand All**
  
  Expands the selection and all its children.
Expand/Collapse > Collapse All
Collapses the selection and all its children.

Expand/Collapse > Expand Children
Expands all the children of the selection but not the selection.

Expand/Collapse > Collapse Children
Collapses all the children of the selection but not the selection.

Expand/Collapse > Collapse Others
Collapses all the siblings of the current selection but not the selection.

_refreshSelected
Forces the layout to be recomputed.

Related Information:
Grid Mode Navigation (on page 579)
Copy and Paste in the Grid Editing Mode (on page 584)
Drag and Drop in the Grid Editing Mode (on page 583)
Content Completion Assistant in Grid Mode (on page 586)

Drag and Drop in the Grid Editing Mode

You can easily arrange sections in your XML document in the Grid mode by using drag and drop actions.

You can do the following with drag and drop:

• Copy or move a set of nodes.
• Change the order of columns in the tables.
• Move the rows from the tables.

These operations are available for both single and multiple selections. To deselect one of the selected fragments, use Ctrl + Single-Click (Command + Single-Click on macOS).

While dragging, the editor paints guide-lines showing the locations where you can drop the nodes. You can also drag nodes outside the Grid editor and text from other applications into the Grid.

Tip:
When using drag and drop to reorganize the document, the resulting layout can be different from what you expected. For instance, the layout can contain a set of sibling tables that can be joined together.
To force the layout to be recomputed, you can use the Refresh Selected action that is available in the contextual menu and in the Document > Grid Edit menu.
Copy and Paste in the Grid Editing Mode

Selecting content in the Grid mode is similar to working with any table with a little more complexity. Specifically, depending on the type of node, when you select a cell, the selection may automatically include additional cells that are implied by the currently selected node. For example, if you click a node that contains any child nodes, all cells that contain the parent and child nodes will be selected. In this case, the currently selected cell is painted with a color that is different from the rest of the selection.

You can also select discontinuous regions of nodes and place them in the clipboard with the copy action. To deselect one of the selected fragments, use Ctrl + Single-Click (Command + Single-Click on macOS).

Pasting Content Within Grid Mode

You can paste copied nodes relative to the currently selected cell using one of the following actions (available in the contextual menu):

- **Paste (Ctrl + V (Command + V on macOS))** - Pastes copied content, as a sibling, just below (after) the current selection.
- **Paste as Child** - Pastes copied content as the last child of the current selection.

Pasting Content from Grid Mode to Other Editors

Nodes that are copied from the Grid editor can also be pasted into Text mode or other external applications. When pasting copied content from Grid mode, the inserted string represents the nodes serialization. The nodes from tables can be copied using HTML or RTF in table format. The resulting cells contain only the concatenated values of the text nodes.
Pasting Content from Other Editors into Grid Mode

You can also paste well-formed XML content or tab-separated values from other editors into the Grid editor. If you paste XML content, the result will be the insertion of the nodes obtained by parsing this content.

Figure 128. Pasting XML Data into Grid

If the pasted text contains multiple lines of tab-separated values, it can be considered as a matrix of values. By pasting this matrix of values into the Grid editor, the result will be a matrix of cells. If the operation is performed inside existing cells, the existing values will be overwritten and new cells will be created when needed.
If you need to add copied content to your existing content (rather than overwriting existing cells), you need to first insert new cells by using the `Insert row` or `Insert column` actions from the contextual menu. This is useful, for example, when trying to transfer data from spreadsheet-like editors to the Grid editor.

### Content Completion Assistant in Grid Mode

If the edited document is associated with a schema (DTD, XML Schema, Relax NG, etc.), the Grid editing mode offers a Content Completion Assistant (on page 3217) for the names and values of elements and attributes. If you choose to insert an element that has required content, the sub-tree of needed elements and attributes are also automatically included.

To display the content completion pop-up menu, simply double-click a cell that contains an element or attribute (or press `Enter` on your keyboard).

### Special Character Support in Grid Mode

If you are editing documents with a bidirectional text orientation or other special characters (such as combining characters), you can change the way the text is rendered and edited in the grid cells by using the Change Text Orientation (`Ctrl + Shift + O` (Command + Shift + O on macOS)) action that is available from the Edit menu in the Grid editing mode. Use this action to switch from the default left to right text orientation to the right to left orientation, and vice versa.

**Note:**

This change applies only to the text from the cells, and not to the layout of the grid editor.
Exporting XML Content to Excel

For use-cases where you have XML content that needs to be exported to Excel (or any other spreadsheet application) but the content is not already in some sort of table format, Grid mode offers you a way to display the content of an XML document as a structured grid of nested tables and you can work with the cells in those tables much like you would with any spreadsheet application. This makes it possible to export content to Excel by copying cells that contain the specific content and then pasting the copied cells in Excel the same as you would when working with any table or spreadsheet.

To export XML content from Grid mode to Excel or other spreadsheet applications, follow this procedure:
1. Open the XML document in Oxygen XML Editor and switch to **Grid** mode.

2. **Expand the nodes (on page 581)** to gain access to the particular nested table that contains the content you want to export.

3. Copy the cells that contain the content you want to export (Copy from the contextual menu or **Ctrl +C**).

4. Switch to your spreadsheet application and paste the copied cells.

5. You may need to make some manual adjustments depending on the complexity of the structure in the original XML document.

Note that Oxygen XML Editor also supports the reverse scenario (copying cells from a spreadsheet application and pasting them in **Grid** mode). For more information, see Import from MS Excel Files – **Grid Mode Method (on page 2093)**.

For more information about exchanging data between Oxygen XML Editor and spreadsheet applications, watch our video demonstration:

https://www.youtube.com/embed/8VwsF58zLkU

**Related Information:**
- Import from MS Excel Files - Grid Mode Method (on page 2093)
- Pasting Content from Other Editors into Grid Mode (on page 585)

**Editing XML Documents in Author Mode**

This section includes topics that describe how to work with XML documents in **Author** mode, including its various features, actions that are available, and much more.

The **Author** editing mode in Oxygen XML Editor allows you to visually edit XML documents in a user-friendly interface that is similar to a WYSIWYG word processor. This makes structured authoring easier for people who are not familiar with XML and it also provides easier access to the XML structure for XML experts. Oxygen XML Editor provides support for visually editing the most commonly used XML vocabularies in **Author** mode, including DITA, Doc Book, TEI, and XHTML.

Adding text content in **Author** mode is as simple as doing so in a standard text editor but the content is rendered similar to how you see it in the output. Tables, images, and media objects (such as videos) are also rendered comparable to the output. You can even play audio and video objects directly in **Author** mode and it includes an intuitive **Image Map Editor (on page 725)**. You can easily change the rendering by selecting one of the preset **main styles (on page 3220)** from the **Styles drop-down menu (on page 590)** (available on the toolbar) and combine multiple **alternate styles (on page 3216)** that behave like layers. You can also use the options in the **Tags Display Mode** drop-down menu (on page 594) to control how much XML markup is displayed in **Author** mode and there are various features and views that provide information about the XML structure based on your current location within the document.
Author mode provides numerous helpful editing actions, many of which are specific to the type of document you are editing and it includes a variety of other powerful editing features, such as keyboard shortcuts, drag and drop support (on page 612), a Smart Paste mechanism (on page 613), and an intelligent Content Completion Assistant (on page 616). Author mode also allows you to visualize and manage profiled content (on page 668), you can collaborate with others with various review features (on page 642) (such as the ability to add comments, track changes, or highlight content), and includes many other unique features.

To switch to this mode, click the Author button at the bottom of the editing area.

For more information about some of the features available in the visual Author editing mode, watch our video demonstration:

https://www.youtube.com/embed/bnQwJZD58wY

Author Mode User Roles

There are two main types of users for the Author mode: framework developers and content authors.

Framework Developers

A framework developer is a technical person with advanced XML knowledge who defines the framework (on page 3219) for authoring XML documents in the visual editor. Once the framework is created or edited by the developer, it is distributed as a deliverable component ready to plug into the application for the content authors.

The framework (document type) configuration defines a type of XML document by specifying all the details needed for editing the content of XML documents in Author mode.

The framework details that are created and customized by the developer include:

- The CSS stylesheet that drives the visual rendering of the document.
- The rules for associating an XML schema with the document, which is needed for the content completion assistance and validation of the document.
- Transformation scenarios for the document.
- Configuration of XML Catalogs (on page 3225).
- Custom actions available as buttons on the toolbar or in menus.

Oxygen XML Editor includes some ready-to-use built-in document types for XML frameworks, such as DocBook, DITA, TEI, JATS, and XHTML.

Content Authors

A content author does not need to have advanced knowledge about XML markup, operations such as validation of XML documents, or applying XPath expressions to an XML document. The content author just uses the framework set up by the developer in the application and starts editing the content of XML documents without editing the XML tags directly.
Changing the Look of Documents in Author Mode Using the Styles Menu

The **Author** mode renders the content of the XML documents visually, based on CSS stylesheets associated with the document.

Oxygen XML Editor provides a **Styles** drop-down menu on the toolbar that allows you to select one main (non-alternate) CSS style (on page 3220) and multiple alternate CSS styles (on page 3216). This makes it easy to change the look of the document as it appears in **Author** mode.

The list of CSS styles that are available in the **Styles** menu depend on the **framework** (on page 3219) (document type).

![Figure 133. Styles Drop-down Menu in a DITA Document](image)

You can use the **Styles** drop-down menu to select a main css style (on page 3220) that applies to the whole document and then select one or more alternate css styles (on page 3216) that behave like layers and are merged sequentially with the main style. Each of the styles that are listed in this drop-down menu have a corresponding CSS file that defines how your documents are rendered in **Author** mode and in the output. Also, the selections from this drop-down menu are persistent, meaning that Oxygen XML Editor remembers them when subsequent documents are opened.

**Main CSS Styles**

The **main styles** are listed in the top section and each of their corresponding CSS files are primarily used to render basic things such as the font and background of the document. You can only select one main style at a time.
Alternate CSS Styles

The alternate styles are listed in the bottom section and their corresponding CSS files are used to render numerous different types of things in the document as they are merged with the selected main styles. You can select as many alternate styles as you wish. If you are unsure about how each of the styles with change the look of your documents based solely upon their name, there is no harm in selecting them to see the difference. You can simply deselect them to revert to the previous look.

Note:
If you deselect the Enable multiple selection of alternate CSSs option (on page 146) in the CSS subtab of the Document Type configuration dialog box (on page 140), the alternate styles are treated like main CSS styles and you can only select one at a time.

Unique CSS Styles for DITA

Oxygen XML Editor comes with a set of built-in CSS layer stylesheets for DITA documents (as well as some that are specifically for DITA maps (on page 3218)). For example, the main styles for DITA documents include Cursive, Century, Facet Design, Roboto, Trip Journal, Papyrus, and more.

Some of these unique alternate styles for DITA documents include:

- **Hints** - Displays tooltips throughout DITA documents that offer additional information to help you with the DITA structure.
- **Inline actions** - Displays possible elements that are allowed to be inserted at various locations throughout DITA documents.
- **Inline insertion actions** - Displays a widget (➕) near each empty paragraph that makes it easy to insert DITA elements (for example, to insert lists, notes, or tables).

Tip:
For information about configuring the Styles drop-down menu, see Configuring and Managing Multiple CSS Styles for a Framework (on page 2145).

Related information
- Associating a Schema to XML Documents (on page 816)
- Configuring and Managing Multiple CSS Styles for a Framework (on page 2145)

Navigating the Document Content in Author Mode

Oxygen XML Editor includes some useful features to help you navigate XML documents.

Navigation Keyboard Shortcuts

Tab
Navigate to the next XML node.

**Tip:**
If you encounter a space-preserved element ([on page 3224](#)) when you navigate through a document and you do not press another key, pressing the **Tab** key will continue the navigation. However, if the cursor is positioned in a space-preserved element and you press another key or you position the cursor inside such an element using the mouse, the **Tab** key can be used to arrange the text.

**Shift + Tab**
Navigate to the previous XML node.

**Ctrl + RightArrow (Command + RightArrow on macOS)**
Navigate one word forward.

**Ctrl + LeftArrow (Command + LeftArrow on macOS)**
Navigate one word backward.

**Ctrl + Home (Command + Home on macOS)**
Position the cursor at the beginning of the document.

**Ctrl + End (Command + End on macOS)**
Position the cursor at the end of the document.

**Navigating to a Modification**

Oxygen XML Editor includes some actions that help you to quickly navigate to a particular modification. These navigation buttons are available in the main toolbar (they can also be accessed from the **Find** menu):

- **Last Modification**
  Navigates to the last modification in any open tab.

- **Back**
  Navigates to the last selected editor tab or to the last selected element/content in the current tab. You can also go back after clicking on links in **Text** or **Author** mode.

- **Forward**
  Available after you use the **Back** button at least once, and it navigates in the opposite direction as the **Back** button.

**Navigating with the Outline View**

Oxygen XML Editor includes an **Outline view ([on page 539](#))** that displays a hierarchical tag overview of the currently edited XML Document.
You can use this view to quickly navigate through the current document by selecting nodes in the outline tree. It is synchronized with the editor area, so when you make a selection in the Outline view, the corresponding nodes are highlighted in the editor area.

**Figure 134. Outline View Navigation in Author Mode**

Using the Breadcrumb to Navigate

A breadcrumb on the stripe at the top of the document indicates the path from document root to the current element. It can also be used as a helpful tool to navigate to specific elements throughout the structure of the document.

**Figure 135. Breadcrumb in Author Mode**

The last element listed in the breadcrumb is the element at the current cursor position. The last element is also highlighted by a thin light blue bar for easier identification. Clicking an element from the breadcrumb selects the entire element and navigates to it in the editor area.

Using the Linking Support

When working on multiple documents that reference each other (references, external entities, XInclude, DITA conref, etc.), the linking support is useful for navigating between the documents. In the built-in frameworks that are bundled with Oxygen XML Editor, links are marked with the icon (or the icon for key-based references). When hovering over the icon, the mouse pointer changes its shape to indicate that the link can be accessed and a tooltip presents the destination location. Click the link to open the referenced resource in the editor or system browser. The same effect can be obtained by using the Document > File > Open file at cursor (Ctrl + Enter (Command + Enter on macOS)) action when the cursor is inside a link element.

**Note:**

Depending on the referenced file type, the target link will either be opened in the Oxygen XML Editor or in the default system application. If the target file does not exist, Oxygen XML Editor prompts you to create it.

Navigating with Bookmarks

A position in a document can be marked with a bookmark. You can then quickly go to the marked position with a keyboard shortcut or a menu action. This is useful when navigating large documents or working on multiple documents where the cursor needs to move between several marked positions. The bookmarks are displayed
with a small icon on the vertical strip to the left of the editor. You can place up to nine distinct **bookmarks** in any document. Shortcut keys are available to navigate to any of the marked positions (**Ctrl+1** through **Ctrl+9**). There are also shortcuts for creating bookmarks (**Ctrl+Shift+1** through **Ctrl+Shift+9**). You can also configure these shortcut keys in the **Options > Menu Shortcut Keys** (on page 294) menu.

Figure 136. Editor Bookmarks

To insert a **bookmark** in **Author** mode, do any of the following:

• Click in the vertical stripe on the left side of the editor (to the left of the line number).
• Press **F9** on your keyboard or use any of the specific bookmark creation shortcuts (**Ctrl+Shift+1** through **Ctrl+Shift+9**).
• Select the **Create Bookmark** action from the **Edit > Bookmarks** menu.

To remove **bookmark** in **Author** mode, do either of the following:

• Left-click its icon in the vertical stripe.
• Right-click its icon on the vertical stripe and select **Remove** or **Remove all** (**Ctrl+F7** (**Command+F7** on macOS)).

To navigate to a specific **bookmark**, do either of the following:

• Use any of the specific bookmark navigation shortcuts (**Ctrl+1** through **Ctrl+9**).
• Use one of the actions available on the **Edit > Bookmarks > Go to** menu.

**Tip:**
The navigation shortcuts work even if the document where the bookmark was inserted has been closed. In this case, using the shortcut will automatically re-open the document.

**Displaying the Markup**

You can control the amount of markup that is displayed in the **Author** mode with various levels of tag modes for both **block** and **in-line** elements.

**Tags Display Mode**

The following dedicated tag modes are available from the **Tags Display Mode** drop-down menu (available on the toolbar):
Full Tags with Attributes
Displays full tag names with attributes for both block (on page 3216) and inline elements (on page 3220).

Full Tags
Displays full tag names without attributes for both block elements and inline elements.

Block Tags
Displays full tag names for block elements and simple tags without names for inline elements.

Block Tags without Element Names
Displays tags for block elements but without element names for a more compact version of Block Tags mode. You can still see the element names by hovering over the tags.

Inline Tags
Displays full tag names for inline elements, while block elements are not displayed.

Partial Tags
Displays simple tags without names for inline elements, while block elements are not displayed.

No Tags
No tags are displayed. This is the most compact mode and is as close as possible to a word-processor view.

Configure Tags Display Mode
Use this option to go to the Author preferences page (on page 176) where you can configure the Tags Display Mode options.

Note:
The associated CSS information is used to determine whether a tag should be considered inline or block. If the current document does not have an associated CSS stylesheet, then the Full Tags mode will be used.

Displaying Referenced Content
The references to entities, XInclude, DITA conrefs, and constructs in other vocabularies with displayable referenced content (on page 2241) are expanded by default in Author mode and the referenced content is displayed. You can control this behavior from the Author preferences page (on page 176). The referenced resources are loaded and displayed inside the element or entity that references them, but the displayed content cannot be modified directly in the document.
If the referenced resource cannot be resolved, an error will be presented inside the element that refers them instead of the content.

If you want to make modifications to the referenced content, you must open the source where the referenced resource resides. The referenced resource can be opened quickly by clicking the link (marked with the $ icon, or the $ icon for key-based references) that is displayed before the referenced content or by using the Edit Reference action from the contextual menu (in this case, the cursor is placed at the precise location where the action was invoked). The referenced resource is resolved through the XML Catalog (on page 3225) set in the XML Catalog preferences page (on page 235).

The referenced content is refreshed as follows:

- Automatically, when it is modified and saved from Oxygen XML Editor.
- On demand, by using the Refresh references action (on page 754). This is useful when the referenced content is modified outside the Oxygen XML Editor scope.

Related Information:
- Configuring a Reference Resolver (on page 2241)

Visual Hints for the Cursor Position

When the cursor is positioned inside a new context, a tooltip will be shown for a couple of seconds displaying the position of the cursor relative to the context of the current element.

Here are some of the common situations that can be encountered:
Before first block - The cursor is positioned before the first block (on page 3216) child of the current node.

Between two block elements - The cursor is positioned between two block elements (on page 3216).

After last block - The cursor is positioned after the last block element (on page 3216) child of the current node.

Inside a node - The cursor is positioned inside a node.

Before an inline element - The cursor is positioned inside an element, before a child inline element (on page 3220).

Between two inline elements - The cursor is positioned between two inline elements (on page 3220).

After an inline element - The cursor is positioned inside an element, after a child inline element (on page 3220).

The nodes in these cases are displayed in the tooltip window using the element names.
To deactivate this feature, open the Preferences dialog box (Options > Preferences) (on page 124), go to Author > Cursor Navigation, and deselect the Show cursor position tooltip option (on page 179). Even if this option is deselected, you can still display the position tooltip by pressing Shift+F2.

Note:
The position information tooltip is not displayed if Full Tags with Attributes or Full Tags is selected in the Tags display mode drop-down menu (on page 594).

Location Tooltip

When editing XML documents in a visual environment, you might find it difficult to position the cursor between certain tags that do not have a visual representation. To counterbalance this, Oxygen XML Editor displays a transparent preview of the position information, called the Location Tooltip:

Figure 139. Location Tooltip

Oxygen XML Editor displays a Location Tooltip when the following conditions are met:

• You are editing the document in one of the following tags display modes (on page 594): Inline Tags, Partial Tags, No Tags.
• The mouse pointer is moved between block elements (on page 3216).

To activate or deactivate this feature, use the Show location tooltip on mouse move option (on page 180) in the Cursor Navigation preferences page (on page 179).

Whitespace Handling in Author Mode

When you edit a document in Author mode, Oxygen XML Editor must serialize the resulting document as XML. Oxygen XML Editor serializes the document when you save it or switch to another editing mode. When the document is serialized, Oxygen XML Editor formats and indents the XML document (on page 555) according to the current format and indent settings (on page 202).

Minimizing Whitespace Differences Between Versions

When serializing a document to XML, Author mode will only format and indent those elements of the document that have been edited. Any element that has not been edited will be serialized exactly as it was loaded from disk. This is useful when your content is managed in a version control systems, as it avoids introducing insignificant whitespace differences between version, which in turn makes diff output easier to read.
Entering Whitespace in Author Mode

Oxygen XML Editor controls the entry of whitespace characters in Author mode according to the XML whitespace rules (on page 555), which means it will not let you insert insignificant whitespace. This means that it will not let you insert extra line-breaks or spaces inside a typical paragraph element, for instance. (Any such whitespace would be normalized away when the document was serialized to XML, so Oxygen XML Editor is saving you from any surprises when this happens.)

Of course, you will legitimately want to enter additional spaces and returns in some cases, such as code samples. Oxygen XML Editor will allow this in elements that are configured as preserve space elements according to the XML whitespace rules. For all of its built-in document types (on page 1291), Oxygen XML Editor is correctly configured to recognize preserve space elements (on page 205) and to allow you to enter additional spaces in them.

If you are using a built-in document type and you are unable to enter additional whitespace, make sure that you are using an element from that document type that is intended to be a preserve-space element.

If you are using a custom document type, make sure that it is configured correctly (on page 2133) so that Oxygen XML Editor recognizes that the current element is a preserve-space element.

Serialization Options for Author Mode

The Options > Preferences > Editor > Edit modes > Author > Serialization page contains some options that control how the formatting and indenting is applied when a document is saved in Author mode or when switching from Author to Text mode. It also includes a Compatibility with other tools option (on page 197) that controls how line breaks are handled when a document is serialized to help obtain better compatibility with other applications.

Editing Content in Author Mode

The Author mode includes a large variety of user-friendly authoring features to help you work with XML content, including numerous toolbar, menu, and shortcut actions and some specialized content editing features.

Undo/Redo Actions

The typical undo and redo actions are available with shortcuts or in the Edit menu:

_undo (Ctrl + Z (Command + Z on macOS))_

Reverses a maximum of 200 editing actions (configurable with the Undo history size option (on page 171) in the Editor preferences page) to return to the preceding state.

_Note:_

Complex operations such as Replace All or Indent selection count as single undo events.
Redo (Ctrl + Y (Command + Shift + Z on macOS, Ctrl + Shift + Z on Linux/Unix))

Recreates a maximum of 100 editing actions that were undone by the Undo function.

Copy and Paste Actions

The typical copying and pasting actions are available with shortcuts or in the contextual menu (or the Edit menu):

- Cut (Ctrl + X (Command + X on macOS))
  Removes the currently selected content from the document and places it in the clipboard.

- Copy (Ctrl + C (Command + C on macOS))
  Places a copy of the currently selected content in the clipboard.

- Paste (Ctrl + V (Command + V on macOS))
  Inserts the current clipboard content into the document at the cursor position.

- Select All (Ctrl + A (Command + A on macOS))
  Selects the entire content of the current document.

Entering Text in Elements

By default, you can only enter text in elements that accept text content. If the element is declared as empty or element only in the associated schema, you are not allowed to insert text in it. Instead, a warning message is displayed.

To allow text to be inserted in these instances, go to the Schema-Aware preferences page and deselect the Reject action when its result is invalid option in the Typing actions section (on page 182).

Editing Text Content Without Modifying the XML Markup

You can use the options in the Tags Display Mode drop-down menu (on page 594) (available on the toolbar) to control how tags are displayed in Author mode. This can help you to clearly see where the current cursor position is within the tag structure so that you can avoid making unintended modifications to the XML markup. You can also switch to the Grid editing mode (on page 578) to modify text content without affecting the XML tags.

Changing the Font Size (Zoom)

The font size of the editor panel can be changed with the following actions that are available with shortcuts or in the Document > Font size menu:
Increase editor font (Ctrl + NumPad+ (Command + NumPad+ on macOS) or Ctrl + MouseWheelForward (Windows/Linux)

Increases the font size (zooms in) with one point for each execution of the action.

Note:
For macOS, if you activate the Enable mouse-wheel zooming option (on page 171) in the Editor preferences page, you can use Command + MouseWheelForward to increase the font size (zoom in). It is disabled by default due to the way inertia affects the mouse wheel on macOS.

Decrease editor font (Ctrl + NumPad- (Command + NumPad- on macOS) or Ctrl + MouseWheelBackwards (Windows/Linux)

Decreases the font size (zooms out) with one point for each execution of the action.

Note:
For macOS, if you activate the Enable mouse-wheel zooming option (on page 171) in the Editor preferences page, you can use Command + MouseWheelBackwards to decrease the font size (zoom out). It is disabled by default due to the way inertia affects the mouse wheel on macOS.

Normal editor font (Ctrl + 0 (Command + 0 on macOS))

Resets the font size to the value of the editor font set in the Fonts preferences page (on page 133).

Related Information:
Editing XML Markup in Author Mode (on page 601)
Drag and Drop in Author Mode (on page 612)
Smart Paste in Author Mode (on page 613)
Content Completion Assistant in Author Mode (on page 616)
Contextual Menu Actions in Author Mode (on page 759)
Frequently Used Shortcut Keys (on page 55)

Editing XML Markup in Author Mode

Oxygen XML Editor includes some useful actions that allow you to easily edit XML markup in Author mode. Most of these actions are available in the contextual menu and some of them have simple keyboard shortcuts.

Selecting XML Markup in Author Mode

Selecting XML tags in Oxygen XML Editor is very simple with several methods for selecting entire elements:
• **Breadcrumb** - Click the element (XML tag) on the breadcrumb (on page 602) displayed at the top of the editing window.

• **Outline View** - Click the element name in the Outline view (on page 539).

• **Full Tags Mode** - While editing in Full Tags mode (on page 594), click the start or end tag of the element in the editor.

• **Mouse Selection** - While editing in Full Tags mode (on page 594), click before the start tag of the element, drag the selection, and release the mouse button after the end tag.

• **Shift + Arrow Keys** - While editing in Full Tags mode (on page 594), place the cursor before the start tag of the element, press and hold Shift, and use the arrow keys to make the selection (including the end tag).

---

**Note:**

If the selection does not include the entire element (for example you do not include the end tag of the element), Oxygen XML Editor will automatically close the appropriate tags when pasting the copied selection. This ensures that the pasted content will always result in well-formed XML (on page 773).

---

### Using the Breadcrumb in Author Mode

A breadcrumb on the top stripe indicates the path from document root to the current element. It can also be used as a helpful tool to insert and edit specific elements in the document structure.

![Figure 141. Breadcrumb in Author Mode](image)

The last element listed in the breadcrumb is the element at the current cursor position. The last element is also highlighted by a thin light blue bar for easier identification. Clicking an element from the breadcrumb selects the entire element in the editor area and each element provides a contextual menu with access to the following actions:

- **Edit Attributes**
  
  Opens the in-place attributes editor (on page 630) that allows you to easily edit the attributes of an element.

- **Edit Profiling Attributes**

  Allows you to select the profiling attributes (on page 668) that apply to a certain element.

- **Append child**

  Opens a content completion list that allows you to select an element to be inserted as a child of the selected element.

- **Insert before**

  Opens a content completion list that allows you to select an element to be inserted (as a sibling) before the selected element.

- **Insert after**
Opens a content completion list that allows you to select an element to be inserted (as a sibling) after the selected element.

**Cut**

Removes the selected element and copies it to the clipboard, while preserving the styles of the content.

**Copy**

Copies the selected element to the clipboard, while preserving the styles of the copied content.

**Paste**

Pastes a well-formed element from the clipboard at currently selected position in the breadcrumb.

**Paste before**

Insert a well-formed element (from the clipboard) before the currently selected element.

**Paste after**

Insert a well-formed element (from the clipboard) after the currently selected element.

**Paste as XML**

Inserts clipboard content that is considered to be well-formed XML content, preserving its XML structure.

**Delete**

Deletes the currently selected element.

**Toggle Comment**

Encloses the currently selected element in an XML comment, if the element is not commented, or removes the comment if it is commented.

**Rename Element**

Opens the Rename dialog box that allows you to rename the currently selected element and other elements with the same name.

**Tip:**

The tag names displayed in the breadcrumb can be customized with an Author mode extension class that implements the AuthorBreadCrumbCustomizer API. See the Oxygen SDK for more details.

**Move Nodes**

You can move XML nodes in the current document by using the following actions in the Refactoring submenu of the contextual menu (or from the Document > Markup menu):

**Move Up (Alt + UpArrow)**

Moves the current node or selected nodes in front of the previous node.
Move Down (Alt + DownArrow)

Moves the current node or selected nodes after the subsequent node.

Tip:
The easiest way to move nodes is to use the Alt + UpArrow and Alt + DownArrow shortcut keys.

Promote/Demote Nodes

You can easily promote or demote selected nodes (for example, within ordered lists or unordered lists) by using the following keyboard shortcuts:

Promote (Shift + Tab)

Promotes an entirely selected node to be a sibling of its parent node (the list item is moved to the left). It also works for selections of multiple nodes as long as all the selected nodes are siblings (on the same hierarchical level).

Demote (Tab)

Demotes an entirely selected node (the list item is moved to the right). It also works for selections of multiple nodes as long as all the selected nodes are siblings (on the same hierarchical level).

Join or Split Elements

You can join or split elements in the current document by using the following actions in the Refactoring submenu of the contextual menu (or from the Document > Markup menu):

Join Elements

Joins two adjacent block elements (on page 3216) that have the same name. The action is available only when the cursor position is between the two adjacent block elements. Also, joining two block elements can be done by pressing the Delete or Backspace keys and the cursor is positioned between the boundaries of these two elements.

Tip:
Specifically, the Delete or Backspace keys can be used to join block elements in the following situations:

- The cursor is located before the end position of the first element and Delete key is pressed.
- The cursor is located after the end position of the first element and Backspace key is pressed.
- The cursor is located before the start position of the second element and Delete key is pressed.
- The cursor is located after the start position of the second element and Backspace key is pressed.
If the element has no sibling or the sibling element has a different name, an **Unwrap** operation will be performed.

**Split Element (Alt + Shift + D (Ctrl + Alt + D on macOS))**

Splits the content of the closest element that contains the position of the cursor. Thus, if the cursor is positioned at the beginning or at the end of the element, the newly created sibling will be empty.

**Rename Elements**

You can rename elements by using the following action in the **Refactoring** submenu of the contextual menu (or from the **Document > Markup** menu):

**Rename Element**

The element from the cursor position, and any elements with the same name, can be renamed according with the options from the **Rename** dialog box.

**Surround Content with Tags (Wrap)**

You can surround a selection of content with tags (**wrap** the content) by using the following action in the **Refactoring** submenu of the contextual menu (or from the **Document > Markup** menu):

**Surround with Tags (Ctrl + E (Command + E on macOS))**

Allows you to choose a tag to enclose a selected portion of content. If there is no selection, the start and end tags are inserted at the cursor position.

- If the **Position cursor between tags** option ([on page 212](#)) is selected in the **Content Completion** preferences page, the cursor is placed between the start and end tag.
- If the **Position cursor between tags** option ([on page 212](#)) is not selected in the **Content Completion** preferences page, the cursor is placed at the end of the start tag, in an insert-attribute position.

**Surround with '[tag]' (Ctrl + ForwardSlash (Command + ForwardSlash on macOS))**

Surround the selected content with the last tag used.

**Unwrap the Content of Elements**

You can unwrap the content of an element by using the following action in the **Refactoring** submenu of the contextual menu (or from the **Document > Markup** menu):

**Delete Element Tags**

Deletes the tags of the closest element that contains the position of the cursor. This operation is also executed if the start or end tags of an element are deleted by pressing the **Delete** or **Backspace** keys.
Tip:
Specifically, the **Delete** or **Backspace** keys can be used to unwrap the content of an element in the following situations:

- The cursor is located before the start position of the element and **Delete** key is pressed.
- The cursor is located after the start position of the element and **Backspace** key is pressed.
- The cursor is located before the end position of the element and **Delete** key is pressed.
- The cursor is located after the end position of the element and **Backspace** key is pressed.

If the element has no sibling or the sibling element has a different name, an **Unwrap** operation will be performed.

**Remove Markup from Blocks of Content**

You can remove the markup from the current element by highlighting the appropriate block of content and using the following action in the **Refactoring** submenu of the contextual menu (or from the **Document > Markup** menu):

- **Remove All Markup**
  Removes all the XML markup inside the selected block of content and keeps only the text content.

Tip:
You can use the **Delete** or **Backspace** keys to remove markup, in which case the elements in the selected block will be unwrapped or joined with their sibling, or if the current element is empty, the element tags will be deleted.

**Remove Text from Selected Markup**

You can remove the text from elements by highlighting the appropriate block of content and using the following action in the **Refactoring** submenu of the contextual menu (or from the **Document > Markup** menu):

- **Remove Text**
  Removes the text content of the selected block of content and keeps the markup intact with empty elements.

**Other Refactoring Actions**

You can also manage the structure of the markup by using the other specific XML refactoring actions that are available in the **Refactoring** submenu of the contextual menu:

- **DITA-related Refactoring Actions**
A variety of built-in XML refactoring operations that pertain to DITA documents with some of the information preconfigured based upon the current context.

**Change Topic ID to File Name**

Use this operation to change the ID of a topic to be the same as its file name.

**Convert CALS Tables to Simple Tables**

Use this operation to convert DITA CALS tables to simple tables. If you invoke this operation from a nested table (a table inside a table), only the nested table will be affected. If it is invoked on a parent table that contains nested tables, all of the contained tables will be converted.

**Convert conrefs to conkeyrefs**

Use this operation to convert `@conref` attributes to `@conkeyref` attributes.

**Convert Simple Tables to CALS Tables**

Use this operation to convert DITA simple tables to CALS tables. If you invoke this operation from a nested table (a table inside a table), only the nested table will be affected. If it is invoked on a parent table that contains nested tables, all of the contained tables will be converted.

**Convert to Concept**

Use this operation to convert a DITA topic (of any type) to a DITA Concept topic type (for example, Topic to Concept).

**Convert to General Task**

Use this operation to convert a DITA topic (of any type) to a DITA General Task topic type (for example, Task to General Task). A DITA General Task is a less restrictive alternative to the Strict Task information type.

**Convert to Reference**

Use this operation to convert a DITA topic (of any type) to a DITA Reference topic type (for example, Topic to Reference).

**Convert to Task**

Use this operation to convert a DITA topic (of any type) to a DITA Task topic type (for example, Topic to Task).

**Convert to Topic**

Use this operation to convert a DITA topic (of any type) to a DITA Topic (for example, Task to Topic).

**Convert to Troubleshooting**

Use this operation to convert a DITA topic (of any type) to a DITA Troubleshooting topic type (for example, Topic to Troubleshooting).

**Rename Key**
Available when invoked on a key, and can be used to quickly rename a key. It also updates all references to it. Note that it does not work on DITA 1.3 key scopes.

**Generate IDs**

Use this operation to automatically generate unique IDs for elements.

**Attributes Refactoring Actions**

Contains built-in XML refactoring operations that pertain to attributes with some of the information preconfigured based upon the current context.

- **Add/Change attribute**
  
  Allows you to change the value of an attribute or insert a new one.

- **Convert attribute to element**
  
  Allows you to change an attribute into an element.

- **Delete attribute**
  
  Allows you to remove one or more attributes.

- **Rename attribute**
  
  Allows you to rename an attribute.

- **Replace in attribute value**
  
  Allows you to search for a text fragment inside an attribute value and change the fragment to a new value.

**Comments Refactoring Actions**

Contains built-in XML refactoring operations that pertain to comments with some of the information preconfigured based upon the current context.

- **Delete comments**
  
  Allows you to delete comments found inside one or more elements.

**Elements Refactoring Actions**

Contains built-in XML refactoring operations that pertain to elements with some of the information preconfigured based upon the current context.

- **Delete element**
  
  Allows you to delete elements.

- **Delete element content**
  
  Allows you to delete the content of elements.

- **Insert element**
  
  Allows you to insert new elements.

- **Rename element**
Allows you to rename elements.

**Unwrap element**

Allows you to remove the surrounding tags of elements, while keeping the content unchanged.

**Wrap element**

Allows you to surround elements with element tags.

**Wrap element content**

Allows you to surround the content of elements with element tags.

**Fragments Refactoring Actions**

Contains built-in XML refactoring operations that pertain to XML fragments with some of the information preconfigured based upon the current context.

**Insert XML fragment**

Allows you to insert an XML fragment.

**Replace element content with XML fragment**

Allows you to replace the content of elements with an XML fragment.

**Replace element with XML fragment**

Allows you to replace elements with an XML fragment.

**Copying XML Content in Author Mode to the Clipboard**

It is possible to copy the XML structure of a document to the system clipboard. Simply select the XML content in **Author** mode (for example, by selecting an element in the breadcrumb), and select **Document > Edit > Copy as XML**. The system clipboard will now contain the corresponding XML structure.

**Related Information:**

- Editing Content in Author Mode *(on page 599)*
- Displaying the Markup *(on page 594)*
- Refactoring XML Documents *(on page 840)*
- Selecting Content in Author Mode *(on page 614)*
- Content Completion Assistant in Author Mode *(on page 616)*
- Contextual Menu Actions in Author Mode *(on page 759)*
- Frequently Used Shortcut Keys *(on page 55)
Editing Attributes in Author Mode

You can easily edit attributes in Author mode by using the Attributes View (on page 627) and Oxygen XML Editor also allows you to edit attribute and element values in-place, directly in the Author mode, using an in-place attribute editor.

In-place Attributes Editor

Oxygen XML Editor includes an in-place attributes editor in Author mode. To edit the attributes of an XML element in-place, do one of the following:

- Select an element or place the cursor inside it and then press the Alt + Enter keyboard shortcut.
- Double-click any named start tag when the document is edited in one of the following display modes (on page 594): Full Tags with Attributes, Full Tags, Block Tags, or Inline Tags.

This opens an in-place attributes editor that contains the same content as the Attributes view. By default, this editor presents the Name and Value fields, with the list of all the possible attributes collapsed.

Figure 142. In-place Attributes Editor

<table>
<thead>
<tr>
<th>Name Combo Box</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use this combo box to select an attribute. The drop-down list displays the list of possible attributes allowed by the schema of the document, as in the Attributes view.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Value Combo Box</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use this combo box to add, edit, or select the value of an attribute. If the selected attribute has predefined values in the schema, the drop-down list displays those possible values. You can use the Browse button to select a URL for the value of an attribute. You can also press Ctrl + Space to open a content completion window that offers a list of possible choices and allows you to select multiple values.</td>
</tr>
</tbody>
</table>

Note:

For built-in frameworks, if the selected attribute in the Name field is an @id attribute, the Browse button is replaced by a Generate Unique ID Value button. Clicking this button will automatically generate a unique ID for the selected element.

If you click More while in the collapsed version, it is expanded to the full version of the in-place attribute editor.
The full version includes a table grid, similar to the Attributes view, that presents all the attributes for the selected element.

**Note:**
If the cursor is located inside read-only content, the attribute names and values are faded and you cannot add, edit, or remove values.

**Related Information:**
Attributes View in Author Mode (on page 627)

**Folding XML Elements in Author Mode**

When working with a large document, the folding (on page 3219) support in Oxygen XML Editor can be used to collapse some element content leaving only the parts that you need to edit in focus. Expanding and collapsing works on individual elements. Expanding an element leaves the child elements unchanged.

The fact that the folds are persistent is a unique feature of Oxygen XML Editor, meaning the next time you open the document the folds are restored to its last state.
Folding Actions in Author Mode

Foldable elements (on page 3219) are marked with a small triangle (\(\wedge\) /) on the left side of the editor panel. If you hover over that arrow, the entire content of the element is highlighted by a dotted border for quick identification of the foldable area. To toggle the fold, simply click the icon. Also, the following actions are available in the Folding sub-menu of the contextual menu or from the Document > Folding menu:

- **Toggle Fold** (or you can simply click on the \(\wedge\) / arrow)
  Toggles the state of the current fold.

- **Collapse Other Folds** (Ctrl + NumPad/ (Command + NumPad/ on macOS))
  Folds all the elements except the current element.

- **Collapse Child Folds** (Ctrl + NumPad. (Command + NumPad. on macOS))
  Folds the child elements that are indented one level inside the current element.

- **Expand Child Folds**
  Unfolds all child elements of the currently selected element.

- **Expand All** (Ctrl + NumPad* (Command + NumPad* on macOS))
  Unfolds all elements in the current document.

For more information about the folding support in Oxygen XML Editor, watch our video demonstration:

https://www.youtube.com/embed/eR9HfN_peAE

Related Information:
Folding Elements: -oxy-foldable Property (on page 2348)

Drag and Drop in Author Mode

The Oxygen XML Editor Author mode includes support for dragging and dropping content in XML documents.

When editing content in Author mode, entire sections or chunks of data can be moved or copied by using the drag and drop feature. The following situations can be encountered:

- When both of the drag and drop sources are from the Author mode editor, a well-formed XML fragment is transferred. The section is balanced before dropping it by adding matching tags when needed.
- When the drag source is from the Author mode editor but the drop target is a text-based editor, only the text inside the selection is transferred as it is.
- The text dropped from another text editor or another application into the Author mode editor is inserted without changes.

Related Information:
Smart Paste in Author Mode (on page 613)
Smart Paste in Author Mode

The **Author** editing mode includes a *Smart Paste* feature that preserves certain style and structure information when copying content and pasting it into document types that support the feature. You can copy content from various sources, including web pages, external applications (such as Office-type applications), or other documents in Oxygen XML Editor, and then paste it into DITA, TEI, DocBook, JATS, and XHTML documents. Oxygen XML Editor preserves the original text styling (such as bold, italics, underline) and formatting (such as lists, tables, paragraphs) and considers various pasting solutions to keep the resulting document valid.

The styles and general layout of the pasted content are converted to the equivalent XML markup for the target document type while preserving certain style and structure information. For example, if you copy content that includes multiple paragraphs and then paste it in **Author** mode, the multiple paragraph structure is preserved. If you paste the content in a location where the resulting XML would not be valid, Oxygen XML Editor will attempt to place it in a valid location, and may prompt you with one or more choices for where to place it.

---

**Tip:**
When pasting text fragments formatted with the *Courier New* font, the *Smart Paste* mechanism will wrap it in an inline code element (for example, in DITA it would be wrapped in a `<codeph>` element).

---

Smart Paste Options

By default, the *Smart Paste* feature is enabled in Oxygen XML Editor. There are several options in the **Schema Aware** preferences page (on page 180) that control the *Smart Paste* mechanism:

- **Smart paste and drag and drop** (on page 181) - This option determines whether or not Oxygen XML Editor will try to find an appropriate insert position when the current location is not valid for the pasted content. This option is selected by default.

- **Reject action when its result is invalid** (on page 182) - If you select this option, Oxygen XML Editor will not let you paste content into a position where it would be invalid. This option is deselected by default.

- **Convert external content on paste** (on page 183) - This option determines whether or not Oxygen XML Editor will convert the styling and formatting of copied content from external sources when pasting it into a document type that supports the feature. This option is selected by default.

- **Convert even when pasting inside space-preserve elements** (on page 183) - If you select this option, the *Smart Paste* feature will also work when pasting external content into a *space-preserve* element (such as a `<codeblock>`). This option is deselected by default.

---

Smart Paste Supported Document Types

The *Smart Paste* feature is supported for the following document types (frameworks (on page 3219)):

- DITA
- DocBook 4
- DocBook 5
- TEI
For more information about the *Smart Paste* support, watch our video demonstration:

https://www.youtube.com/embed/bpiXZQwzBfA

**Related Information:**
- Customizing Smart Paste Support *(on page 2188)*
- Migrating MS Office Documents to DITA *(on page 3188)*
- Oxygen Batch Converter add-on (Convert Markdown/HTML to DITA or DocBook)

**Selecting Content in Author Mode**

Oxygen XML Editor includes a variety of features and keyboard shortcuts to help you select content in **Author** mode.

**Selection Shortcuts in Author Mode**

- **Ctrl + A (Meta + A on macOS)**
  Selects all content in the document.

- **Shift + Left/Right Arrow Keys**
  Begins a continuous selection at the cursor position and extends it one character at a time in the direction that you press the arrow keys.

- **Shift + Up/Down Arrow Keys**
  Begins a continuous selection at the cursor position and extends it one line at a time in the direction that you press the arrow keys.

- **Ctrl + Shift + Left/Right Arrow Keys (Meta + Shift + Left/Right Arrow Keys on macOS)**
  Begins a continuous selection at the cursor position and extends it one word at a time in the direction that you press the arrow keys.

- **Shift + Home**
  Begins a continuous selection at the cursor position and extends it to the beginning of the current line (on macOS, it extends to the beginning of the document).

- **Shift + End**
  Begins a continuous selection at the cursor position and extends it to the end of the current line (on macOS, it extends to the end of the document).

- **Ctrl + Shift + Home**
  Begins a continuous selection at the cursor position and extends it to the beginning of the document.
Ctrl + Shift + End

Begins a continuous selection at the cursor position and extends it to the end of the document.

Shift + PageUp

Begins a continuous selection at the cursor position and extends it up one screen page.

Shift + PageDown

Begins a continuous selection at the cursor position and extends it down one screen page.

Double-Click

Selects the word at the cursor position.

Triple-Click

Selects the node at the cursor position.

Right-Click > Select > Element

Selects the entire element at the current cursor position.

Right-Click > Select > Content

Selects the entire content of the element at the current cursor position, excluding the start and end tag. Performing this action repeatedly will result in the selection of the content of the ancestor of the currently selected element content.

Right-Click > Select > Parent

Selects the entire parent element at the current cursor position.

Intelligent Selection in Author Mode

Oxygen XML Editor supports all usual content selection methods using the mouse or keyboard shortcuts (for example Shift + Arrow Keys). When selecting content in Author mode, there may be instances where you want to select an element (along with its content) rather than just inline text content. There are several ways to select an element, including:

- You can select an element by clicking the element name in the breadcrumb (on page 593).
- You can select an element using the Outline view (on page 539).
- If you have the Tags Display Mode (on page 594) set to any of the modes other than Partial Tags or No Tags, you can select an element directly in the main editing pane by clicking on the element's tag.
- If you have the Tags Display Mode (on page 594) set to Partial Tags (and the Compact tag layout option is enabled in the Author preferences page (on page 179)), you can still select certain block elements directly in the main editing pane by spanning the selection to the right of the end of the content that is inside a block element so that the selection includes the element's invisible end tag. In this mode, the end tag is normally not visible, but when your selection includes the invisible end tag, a tooltip displays the selected element name and the selection includes the whole element (along with its content).
Content Completion Assistant in Author Mode

One of the most useful features in Author mode is the Content Completion Assistant (on page 3217). It offers a list of elements, attributes, attribute values, and other options that are valid in the current editing context.

Figure 145. Content Completion Assistant in Author Mode

The Content Completion Assistant is enabled by default. To disable it, open the Preferences dialog box (Options > Preferences) (on page 124), go to Editor > Content Completion, and deselect the Enable content completion option (on page 211).

Using the Content Completion Assistant in Author Mode

To activate the feature in Author mode, use any of the following shortcut keys:

- Enter
- Ctrl + Space (Command + Space on macOS)
- Alt + ForwardSlash (Command + Alt + ForwardSlash on macOS)

You can navigate through the list of proposals by using the Up and Down keys on your keyboard. In same cases, the Content Completion Assistant displays a documentation window with information about the particular proposal and some of them have links to additional information (for example, DITA elements might have a link to the DITA Style Guide). You can use Tab and Shift + Tab to navigate to those links and Space to trigger them. You can also change the size of the documentation window by dragging its top, right, and bottom borders.

To insert the selected proposal in Author mode, simply press Enter.

Types of Proposals Listed in the Content Completion Assistant

The Content Completion Assistant offers the following types of proposed actions depending on the current context:

- Insert allowed elements for the current context schema and the list of proposals contains elements depending on the elements inserted both before and after the cursor position.
- Insert element values if such values are specified in the schema for the current context.
- Insert new undeclared elements by entering their name in the text field.
• Insert CDATA sections, comments, processing instructions.

• Insert code templates (on page 536).

• If invoked on a selection that only contains an element start or end tag (remember that you can see all element tags while working in Full Tags mode (on page 594)), it will allow you to rename the element.

• If invoked on a selection of multiple elements or other content, it will allow you to surround the content with certain tags.

• If invoked on an empty list item that is the last element of the list, it will allow you to convert the list item to a paragraph.

• If the Show all possible elements in the content completion list option from the Schema-Aware preferences page (on page 182) is selected, the content completion pop-up window will present all the elements defined by the schema. When choosing an element from this section, the insertion will be performed using the schema-aware smart editing features.

**Note:**

By default, you are not allowed to insert element names that are not defined by the schema. This can be changed by deselecting the Allow only insertion of valid elements and attributes check box from the Schema-Aware preferences page (on page 182).

### Examples of How the Content Completion Assistant Works

To illustrate how the feature works, consider the following examples of invoking the Content Completion Assistant in certain contexts:

• If the cursor is positioned at the beginning or at the end of the element, the first item offered in the Content Completion Assistant is a New <Element> item. Selecting this item will insert an empty element.

  ![Figure 146. Example (New [Element Name])](image)

• If the cursor is positioned somewhere inside the element, the first entry in the Content Completion Assistant is a Split <Element> item. In most cases, you can only split the closest block element (on page 3216) to the cursor position, but if it is inside a list item, the list item will also be proposed for split. Selecting Split <Element> splits the content of the specified element around the cursor position.
• If the cursor is positioned inside a space-preserved element (on page 3224) (for example, a codeblock), the first choice in the Content Completion Assistant is Enter, which will insert a new line in the content of the element, followed by New <Element>.

• If invoked on a selection that only contains an element start or end tag (remember that you can see all element tags while working in Full Tags mode (on page 594)), it will allow you to rename the element.
• If invoked on a selection of multiple elements or other content, it will allow you to surround the content with certain tags.

![Figure 150. Example (Surround)](image)

**Set the Schema to be Used for Content Completion**

The proposals that are presented in the *Content Completion Assistant (on page 3217)* depend on the associated schemas. The DTD, XML Schema, Relax NG, or NVDL schema used to populate the *Content Completion Assistant* is specified in the following methods, in the order of their precedence:

- The *schema specified explicitly in the document (on page 823)*. In this case, Oxygen XML Editor reads the beginning of the document and resolves the location of the DTD, XML Schema, Relax NG schema, or NVDL schema.
- The *default schema declared (on page 825)* in the *Schema tab of the Document Type configuration dialog box (on page 144)* for the particular document type.

**Schema Annotations in Author Mode**

A *schema annotation* is a documentation snippet associated with the definition of an element or attribute in a schema. If such a schema is associated with an XML document, the annotations are displayed in the *Content Completion Assistant (on page 3217).*

![Figure 151. Schema Annotation in the Content Completion Assistant](image)

The schema annotations support is available if the schema type is one of the following:
• XML Schema
• Relax NG
• NVDL schema
• DTD

This feature is enabled by default, but you can disable it by deselecting the Show annotations in Content Completion Assistant (on page 216) option in the Annotations preferences page.

Styling Annotations with HTML

You can use HTML format in the annotations you add in an XML Schema or Relax NG schema. This improves the visual appearance and readability of the documentation window displayed when editing XML documents validated against such a schema. An annotation is recognized and displayed as HTML if it contains at least one HTML element (such as `<div>`, `<body>`, `<p>`, `<br>`, `<table>`, `<ul>`, or `<ol>`).

The HTML rendering is controlled by the Show annotations using HTML format, if possible (on page 216) option in the Annotations preferences page. When this option is deselected, the annotations are converted and displayed as plain text and if the annotation contains one or more HTML tags (<p>, <br>, <ul>, <li>), they are rendered as an HTML document loaded in a web browser. For example, `<p>` begins a new paragraph, `<br>` breaks the current line, `<ul>` encloses a list of items, and `<li>` encloses an item of the list.

Collecting Annotations from XML Schemas

In an XML Schema, the annotations are specified in an `<xs:annotation>` element like this:

```xml
<xs:annotation>
  <xs:documentation>
    Description of the element.
  </xs:documentation>
</xs:annotation>
```

If an element or attribute does not have a specific annotation, then Oxygen XML Editor looks for an annotation in the type definition of that element or attribute.

Collecting Annotations from Relax NG Schemas

For Relax NG schema, element and attribute annotations are made using the `<documentation>` element from the http://relaxng.org/ns/compatibility/annotations/1.0 namespace like this:

```xml
<define name="person">
  <element name="person">
    <a:documentation xmlns:a="http://relaxng.org/ns/compatibility/annotations/1.0">
      Information about a person.  
    </a:documentation>
    <ref name="name"/>
    <zeroOrMore>
      <ref name="email"/>
    </zeroOrMore>
  </element>
</define>
```
However, any element outside the Relax NG namespace (http://relaxng.org/ns/structure/1.0) is handled as annotation and the text content is displayed in the annotation window. To activate this behavior, select the **Use all Relax NG annotations as documentation** (on page 217) option in the **Annotations** preferences page.

### Collecting Annotations from Relax NG Compact Syntax Schemas

For Relax NG Compact Syntax schema, annotations are made using comments like this:

```xml
<!-- Information about a person. -->
<define person { name, email*}
```

### Collecting Annotation from DTDs

For DTD, Oxygen XML Editor defines a custom mechanism for annotations using comments enabled by the **Prefer DTD comments that start with "doc:" as annotations** (on page 216) option in the **Annotations** preferences page. The following is an example of a DTD annotation:

```xml
<!--doc:Description of the element. -->
```

**Related Information:**
- Customizing the Rendering of Elements (on page 2207)
- Customizing Annotations in the Content Completion Assistant (on page 2212)

### Content Completion Helper Views (Author Mode)

Information about the current element being edited is also available in various dockable (on page 3218) views, such as the **Model view** (on page 545), **Attributes view** (on page 627), **Elements view** (on page 633), and **Entities view** (on page 547). By default, they are located on the right-hand side of the main editor window. These views, along with the powerful **Outline view** (on page 539), provide spatial and insight information about the edited document and the current element. If any particular view is not displayed, it can be opened by selecting it from the **Window > Show View** menu.

### Code Templates

Code templates are code fragments that can be inserted quickly at the current editing position. Oxygen XML Editor includes a set of built-in code templates for CSS, LESS, Schematron, XSL, XQuery, JSON, HTML, and XML Schema document types. You can also define your own code templates for any type of file and share them with others.

Code templates are displayed with a symbol in the content completion list (Enter in Author mode or Ctrl + Space (Command + Space on macOS) in Text mode). Also, in Text mode you can press Ctrl + Shift + Space to see a complete list of the available code templates. To enter the code template at the cursor position, select
it from the content completion list or type its code and press Enter. If a shortcut key has been assigned to the code template, you can also use the shortcut key to enter it.

**How to Create Code Templates**

To create a code template, follow these steps:

1. Open the Preferences dialog box (Options > Preferences) (on page 124) and go to Editor > Content Completion > Code Templates.

2. Click New to open a code template configuration dialog box.

   ![Figure 152. Code Template Configuration Dialog Box](image)

   Tip:
   You can use one of the existing code templates as a starting point by selecting that template and clicking Duplicate.

3. Configure your template using the fields in the code template configuration dialog box:
   - **Name** - The name of the code template.
   - **Description** - [Optional] The description of the code template that will appear in the Code Templates preferences page and in the tooltip message when selecting it from the Content Completion Assistant (on page 3217). HTML markup can be used for better rendering.
   - **Associate with** - You can choose to set the code template to be associated with a specific type of editor or for all editor types.
   - **Shortcut key** - [Optional] If you want to assign a shortcut key that can be used to insert the code template, place the cursor in the Shortcut key field and press the desired key combination on your keyboard. Use the Clear button if you make a mistake. If the Enable platform-independent shortcut keys checkbox is selected, the shortcut is platform-independent and the following modifiers are used:

   - **Ctrl**
   - **Shift**
   - **Alt**
   - **Win**
   - **Mac**
- **M1** represents the **Command** key on macOS, and the **Ctrl** key on other platforms.
- **M2** represents the **Shift** key.
- **M3** represents the **Option** key on macOS, and the **Alt** key on other platforms.
- **M4** represents the **Ctrl** key on macOS, and is undefined on other platforms.

  - **Content** - Text box where you define the content that is used when the code template is inserted.

  An editor variable (on page 323) can be inserted in the text box using the **Insert Editor Variables** button.

4. Click **OK** to save your new code template.

**Result:** Your code template can now be selected using the **Content Completion Assistant (on page 3217)** (Enter in **Author** mode or **Ctrl + Space (Command + Space on macOS)** in **Text** mode). The code templates are displayed with a ⚪ symbol.

### How to Share Code Templates

There are two ways to easily share all of your code templates with other members of your team:

**Method 1: Export/Import**

1. Open the **Preferences** dialog box (Options > Preferences) (on page 124) and go to **Editor > Templates > Code Templates**.
2. Click the **Export** button to export all of your code templates into an XML file.
3. Save the XML file.
4. Share the XML file with other members of your team.
5. Instruct them to open the **Preferences** dialog box (Options > Preferences) (on page 124), go to **Editor > Templates > Code Templates**, click the **Import** button, and select the file you sent them.

   **Result:** The code templates will be now available in their content completion list.

**Method 2: Share Project**

1. Open the **Preferences** dialog box (Options > Preferences) (on page 124) and go to **Editor > Templates > Code Templates**.
2. Select **Project Options** at the bottom of the dialog box. This stores the preferences in the project file (**.xpr**).
3. Share the project file with the other members of your team. For example, you can commit it to your version control system and have them update their working copy.

   **Result:** When they open the updated project file in their **Project view (on page 402)**, the code templates will be available in their content completion list.
Tip:
It is also possible to configure certain actions that function similar to code templates and add them to the content completion list (on page 2191) for a particular framework. You could then share the whole framework (on page 2288) with other members of your team.

Author Mode Views

The content author is supported by a variety of dockable (on page 3218) helper views that are displayed by default when editing in Author mode. These views are automatically synchronized with the current editing context of the editor panel. They present additional information about this context thus helping the author to see quickly the current location in the overall document structure and the available editing options.

There is also a large selection of additional useful views available in the Window > Show View menu. This section presents some of the most helpful views for editing in Author mode.

Outline View for XML Documents

The Outline view displays a general tag overview of the currently edited XML document. When you edit a document, the Outline view dynamically follows the changes that you make, displaying the node that you modify. This functionality gives you great insight on the location of your modifications in the current document. It also shows the correct hierarchical dependencies between elements. This makes it easy for you to be aware of the document structure and the way element tags are nested.

Outline View Features

The Outline view allows you to:

- Quickly navigate through the document by selecting nodes in the Outline tree.
- Insert or delete nodes using contextual menu actions.
- Move elements by dragging them to a new position in the tree structure.
- Highlight elements in the editor area. It is synchronized with the editor area, so when you make a selection in the editor area, the corresponding nodes are highlighted in the Outline view, and vice versa.
- View document errors, as they are highlighted in the Outline view. A tooltip also provides more information about the nature of the error when you hover over the faulted element.

Outline View Interface

By default, it is displayed on the left side of the editor. If the view is not displayed, it can be opened by selecting it from the Window > Show View menu.

The upper part of the Outline view contains a filter box that allows you to focus on the relevant components. Type a text fragment in the filter box and only the components that match it are presented. For advanced usage you can use wildcard characters (such as * or ?) and separate multiple patterns with commas.
It also includes a settings menu in the top-right corner that presents a variety of options to help you filter the view even further.

**Drag and Drop Actions in the Outline View**

Entire XML elements can be moved or copied in the edited document using only the mouse in the Outline view with drag-and-drop operations. Several drag and drop actions are possible:

- If you drag an XML element in the Outline view and drop it on another node, then the dragged element will be moved after the drop target element.
- If you hold the mouse pointer over the drop target for a short time before the drop then the drop target element will be expanded first and the dragged element will be moved inside the drop target element after its opening tag.
- You can also drop an element before or after another element if you hold the mouse pointer towards the upper or lower part of the targeted element. A marker will indicate whether the drop will be performed before or after the target element.
- If you hold down the Ctrl (Command on macOS) key after dragging, a copy operation will be performed instead of a move.

**Figure 153. Outline View**

**Outline View Filters**

The upper part of the Outline view contains a filter box that allows you to focus on the relevant components. Type a text fragment in the filter box and only the components that match it are presented. For advanced usage you can use wildcard characters (such as * or ?) and separate multiple patterns with commas.

The following actions are available in the settings menu of the Outline view:
Filter returns exact matches
The text filter of the Outline view returns only exact matches.

Selection update on cursor move (Available in Text mode)
Controls the synchronization between Outline view and source document. The selection in the Outline view can be synchronized with the cursor moves or the changes in the editor. Selecting one of the components from the Outline view also selects the corresponding item in the source document.

Flat presentation mode of the filtered results
When active, the application flattens the filtered result elements to a single level.

Show comments and processing instructions
Show/hide comments and processing instructions in the Outline view.

Show element name
Show/hide element name.

Show text
Show/hide additional text content for the displayed elements.

Show attributes
Show/hide attribute values for the displayed elements. The displayed attribute values can be changed from the Outline preferences panel (on page 306).

Configure displayed attributes
Displays the XML Structured Outline preferences page (on page 306).

Outline View Contextual Menu Actions
The contextual menu of the Outline view contains the following actions:

Edit Attributes
Displays an in-place attributes editor that allows you to edit the attributes of a selected node.

Edit Profiling Attributes (Available in Author mode)
Allows you to change the profiling attributes (on page 668) defined on all selected elements.

Append Child
Invokes a content completion list with the names of all the elements that are allowed by the associated schema and inserts your selection as a child of the current element.

Insert Before
Invokes a content completion list with the names of all the elements that are allowed by the associated schema and inserts your selection immediately before the current element, as a sibling.
Insert After

Invokes a content completion list with the names of all the elements that are allowed by the associated schema and inserts your selection immediately after the current element, as a sibling.

Cut, Copy, Paste, Delete common editing actions

Executes the typical editing actions on the currently selected elements. The Cut and Copy operations preserve the styles of the copied content.

Paste before (Available in Author mode)

Inserts a well-formed copied element before the currently selected element.

Paste after (Available in Author mode)

Inserts a well-formed copied element after the currently selected element.

Paste as XML (Available in Author mode)

Pastes copied content that is considered to be valid XML, preserving its XML structure.

Toggle Comment

Encloses the currently selected element in an XML comment, if the element is not already commented. If it is already commented, this action will remove the comment.

Rename Element (Available in Author mode)

Invokes a Rename dialog box that allows you to rename the currently selected element, siblings with the same name, or all elements with the same name.

Expand More

Expands the structure tree of the currently selected element.

Collapse All

Collapses all of the structure tree of the currently selected node.

Tip:

You can copy, cut or delete multiple nodes in the Outline by using the contextual menu after selecting multiple nodes in the tree.

Attributes View in Author Mode

The Attributes view presents all the attributes of the current element determined by the schema of the document. By default, it is located on the right side of the editor. If the view is not displayed, it can be opened from the Window > Show View menu.

You can use this view to edit or add attribute values. The attributes of an element are editable if any one of the following is true:
• The CSS stylesheet associated with the document does not specify a false value for the -oxy-editable (on page 2347) property associated with the element.
• The element is entirely included in a deleted Track Changes (on page 642) marker.
• The element is part of a content fragment that is referenced in Author mode from another document.

The attributes are rendered differently depending on their state:

• The names of the attributes are rendered with a bold font, and their values with a plain font.
• Default values are rendered with a plain font, painted gray.
• Empty values display the text "[empty]", painted gray.
• Invalid attributes and values are painted red.

To edit the value of the corresponding attribute, double-click a cell in the Value column. If the possible values of the attribute are specified as list in the schema of the edited document, the Value column acts as a combo box that allows you to either select the value from a list or manually enter it.

Note:
If the cursor is located inside read-only content, the attribute names and values are faded and you cannot add, edit, or remove values.

You can sort the attributes table by clicking the Attribute column header. The table contents can be sorted as follows:

• By attribute name in ascending order.
• By attribute name in descending order.
• Custom order, where the used attributes are displayed at the beginning of the table sorted in ascending order, followed by the rest of the allowed elements sorted in ascending order.

Figure 154. Attributes View
A drop-down list located in the upper part of the view allows you to select the current element or its ancestors.

**Expand/Collapse Button**

There is an Expand/Collapse ( ▼ / ▲ ) button at the top-right of the view. When expanded, this presents the following additional combo boxes:

**Name Combo Box**

Use this combo box to select an attribute. The drop-down list displays the list of possible attributes allowed by the schema of the document, as in the Attributes view. You can use the **Remove** button to delete an attribute and its value from the selected element.

**Value Combo Box**

Use this combo box to add, edit, or select the value of an attribute. If the selected attribute has predefined values in the schema, the drop-down list displays those possible values. You can use the **Browse** button to select a URL for the value of an attribute. You can also press **Ctrl + Space** to open a content completion window that offers a list of possible choices and allows you to select multiple values. After you have entered or selected a value, use the **Update** button (or press **Enter**) to add the value to the attribute.

*Note:* For built-in frameworks, if the selected attribute in the Name field is an @id attribute, the **Browse** button is replaced by a **Generate Unique ID Value** button. Clicking this button will automatically generate a unique ID for the selected element.

**Contextual Menu Actions in the Attributes View**

The following actions are available in the contextual menu of the Attributes view when editing in Author mode:

**Set empty value**

Specifies the current attribute value as empty.

**Remove**

Removes the attribute (action available only if the attribute is specified). You can invoke this action by pressing the **Delete** or **Backspace** keys.

**Copy**

Copies the attrName="attrValue" pair to the clipboard. The attrValue can be:

- The value of the attribute.
- The value of the default attribute, if the attribute does not appear in the edited document.
- Empty, if the attribute does not appear in the edited document and has no default value set.

**Paste**
Depending on the content of the clipboard, the following cases are possible:

- If the clipboard contains an attribute and its value, both of them are introduced in the **Attributes** view. The attribute is selected and its value is changed if they exist in the **Attributes** view.
- If the clipboard contains an attribute name with an empty value, the attribute is introduced in the **Attributes** view and you can start editing it. The attribute is selected and you can start editing it if it exists in the **Attributes** view.
- If the clipboard only contains text, the value of the selected attribute is modified.

**In-place Attributes Editor**

Oxygen XML Editor includes an in-place attributes editor in **Author** mode. To edit the attributes of an XML element in-place, do one of the following:

- Select an element or place the cursor inside it and then press the **Alt + Enter** keyboard shortcut.
- Double-click any named start tag when the document is edited in one of the following display modes *(on page 594)*: **Full Tags with Attributes**, **Full Tags**, **Block Tags**, or **Inline Tags**.

This opens an in-place attributes editor that contains the same content as the **Attributes** view. By default, this editor presents the **Name** and **Value** fields, with the list of all the possible attributes collapsed.

![Figure 155. In-place Attributes Editor](image)

**Name Combo Box**

Use this combo box to select an attribute. The drop-down list displays the list of possible attributes allowed by the schema of the document, as in the **Attributes** view.

**Value Combo Box**

Use this combo box to add, edit, or select the value of an attribute. If the selected attribute has predefined values in the schema, the drop-down list displays those possible values. You can use the **Browse** button to select a URL for the value of an attribute. You can also press **Ctrl + Space** to open a content completion window that offers a list of possible choices and allows you to select multiple values.
Note: For built-in frameworks, if the selected attribute in the Name field is an @id attribute, the Browse button is replaced by a Generate Unique ID Value button. Clicking this button will automatically generate a unique ID for the selected element.

If you click More while in the collapsed version, it is expanded to the full version of the in-place attribute editor.

Figure 156. In-place Attributes Editor (Full Version)

The full version includes a table grid, similar to the Attributes view, that presents all the attributes for the selected element.

Model View

The Model view presents the structure of the currently selected tag, and its documentation, defined as annotation in the schema of the current document. By default, it is located on the right side of the editor. If the view is not displayed, it can be opened by selecting it from the Window > Show View menu.
The Model view is comprised of two sections, an element structure panel and an annotations panel.

**Element Structure Panel**

The element structure panel displays the structure of the currently edited or selected tag in a tree-like format. The information includes the name, model, and attributes of the current tag. The allowed attributes are shown along with imposed restrictions, if any.
Annotation Panel

The Annotation panel displays the annotation information for the currently selected element. This information is collected from the XML schema.

Figure 159. Annotation panel

Elements View in Author Mode

The Elements view presents a list of all defined elements that are valid at the current cursor position according to the schema associated to the document. By default, it is located on the right side of the editor. If the view is not displayed, it can be opened by selecting it from the Window > Show View menu.

The upper part of the view features a combo box that contains the ordered ancestors of the current element. Selecting a new element in this combo box updates the list of the allowed elements. By default, only the elements that are allowed at the current cursor position are listed. However, if the Show only allowed items option (on page 306) is not selected in the Views preferences page (on page 306), all elements allowed by the schema will be listed.

Double-clicking any of the listed elements inserts that element into the edited document at the current cursor position.

Pressing F2 with an element selected will display information about that particular element.

Figure 160. Elements View in Author Mode
Entities View

Entities provide abbreviated entries that can be used in XML files when there is a need of repeatedly inserting certain characters or large blocks of information. An entity is defined using the ENTITY statement either in the DOCTYPE declaration or in a DTD file associated with the current XML file.

There are three types of entities:

- **Predefined** - Entities that are part of the predefined XML markup (\ `<`, \ `>`, \ `&`, \ `apos`, \ `quot`).
- **Internal** - Defined in the DOCTYPE declaration header of the current XML.
- **External** - Defined in an external DTD module included in the DTD referenced in the XML DOCTYPE declaration.

**Note:**

If you want to add internal entities, you would need to switch to the Text editing mode and manually modify the DOCTYPE declaration. If you want to add external entities, you need to open the DTD module file and modify it directly.

The Entities view displays a list with all entities declared in the current document, as well as built-in ones. By default, it is located on the right side of the editor. If the view is not displayed, it can be opened by selecting it from the Window > Show View menu.

Double-clicking one of the entities will insert it at the current cursor position in the XML document. You can also sort entities by name and value by clicking the column headers.

![Figure 161. Entities View](image-url)
The view features a filtering capability that allows you to search an entity by name, value, or both. Also, you can choose to display the internal or external entities.

**Note:**
When entering filters, you can use the `?` and `*` wildcards. Also, you can enter multiple filters by separating them with a comma.

**Results View**

The **Results** view displays the messages generated as a result of user actions such as validations, transformations, search operations, and others. Each message is a link to the location related to the event that triggered the message. Double-clicking a message opens the file containing the location and positions the cursor at the location offset. The **Results** view is automatically opened when certain actions generate result messages. By default, the view normally opens at the bottom of the editor, but it is **dockable** (on page 3218), so it can be moved to another UI location alongside other side views.

**Tip:**
To shift focus to the open **Results** view without using the mouse, there is an action in the **Window > Results** menu called **Focus Results** that can be used for this purpose and you can specify a keyboard shortcut (on page 294) to this action.

The actions that contribute messages to this view include:

- **Validation** actions (on page 775)
- **Transformation** actions (on page 1432)
- **Check Spelling in Files** action (on page 458)
- **Find All** action from the **Find/Replace** dialog box (on page 432)
- **Find/Replace in Files** dialog box (on page 436)
- **Search References** action (on page 911)
- **XPath expression results** (on page 2000)
- **SQL results** (on page 2073)

**Figure 162. Results View**
Results View Toolbar Actions

The view includes a toolbar with the following actions:

⚙️ Settings drop-down menu

This drop-down menu also includes the following options:

- **Group by "Severity"**
  
  Groups the results based upon the severity of the validation issues.

- **Group by "Resource"**
  
  Groups the results based upon the type of resource.

- **Group by "System ID"**
  
  Groups the results based upon the system ID of the resource.

- **Group by "Operation description"**
  
  Groups the results based upon the description of the validation issue.

≡ Ungroup all

Removes the grouping rules so that the messages are presented in a continuous list.

Show group columns

If any of the **Group by** options are selected, you can use this option to show or hide grouping columns.

Restore default grouping

Restores the column size for each column and the grouping rules that were saved in the user preferences the last time when this view was used. If it is the first time this view is used, the action sets an initial default column size for each column and a grouping rule that is appropriate for the type of messages. For example:

- Group the messages by the path of the validated file if there are error messages from a validation action or spelling errors reported by the **Check Spelling in Files** action (on page 458).
- No grouping rule for the results of applying an XPath expression (on page 1999).

Include problem ID in description

If this option is selected, validation issues will include the problem ID (as provided by the validation engine) in the **Description** column.

Show Ignored Problems
If you have ignored validation problems (on page 812), you can deselect this option to hide the ignored problems. Likewise, you can select this option to show the ignored problems.

**Highlight all results in editor**

Oxygen XML Editor highlights all matches obtained after executing an XPath expression, or performing one of the following operations: Find All, Find in Files, Search References, and Search Declarations. Click Highlight all results in editor again to turn off highlighting.

**Note:**

To customize highlighting behavior, open the Preferences dialog box (Options > Preferences) (on page 124) and go to Editor > Highlights category. You can do the following customizations:

- Set a specific color of the highlights depending on the type of action you make.
- Set a maximum number of highlights that the application displays at any given time.

**Remove selected**

Removes the current selection from the view. This can be helpful if you want to reduce the number of messages, or remove those that have already been addressed or not relevant to your task.

**Remove all**

Removes all messages from the view.

**Results View Contextual Menu Actions**

The following actions are available when the contextual menu is invoked in this view:

- **Learn Word(s) (Available when spelling errors are reported in the Results view)**
  
  Adds the word(s) to a list of learned words to instruct the spell checker engine to not report the word(s) as spelling errors in the future.

- **Show message**
  
  Displays a dialog box with the full error message, which is useful for a long message that does not have enough room to be displayed completely in the view.

- **Previous message**
  
  Navigates to the message above the current selection.

- **Next message**
  
  Navigates to the message below the current selection.
**Remove selected**
Removes selected messages from the view.

**Remove all**
Removes all messages from the view.

**Copy**
Copies information associated with the selected messages. For example:

- The file path of the document that triggered the output message.
- The path of the main file (on page 3220) (in the case of a validation scenario (on page 787), it is the path of the file where the validation starts and can be different from the validated file).
- Error severity (error, warning, info message, etc.)
- Name of validating processor.
- Name of validation scenario (on page 787).
- The line and column in the file that triggered the message.

**Copy Description**
Copies the description values for all selected items. It is possible to assign a shortcut key (on page 296) for this action.

**Select All**
Extends the selection to all the messages from the view.

**Print Results**
Sends the complete list of messages to a printer. For each message, the included details are the same as the ones for the Copy action (on page 638). This action is also available in the Window > Results menu.

**Save Results**
Saves the complete list of messages in a file in text format. For each message, the included details are the same as the ones for the Copy action (on page 638). This action is also available in the Window > Results menu.

**Save Results as XML**
Saves the complete list of messages in a file in XML format. For each message, the included details are the same as the ones for the Copy action (on page 638).

**Save Results as HTML**
Saves the complete list of messages in a file in HTML format. For each message, the included details are the same as the ones for the Copy action (on page 638).
A set of **Group by** toggle actions that allow you to group the messages according to a selected criteria so that they can be presented in a hierarchical layout. The criteria used for grouping can be the severity of the errors (error, warning, info message, etc.), the resource name, the description of the message, and so on.

**Ungroup all**

Removes the grouping rules so that the messages are presented in a continuous list.

**Show group columns**

If any of the **Group by** options are selected, you can use this option to show or hide grouping columns.

**Restore default grouping**

Restores the column size for each column and the grouping rules that were saved in the user preferences the last time when this view was used. If it is the first time this view is used, the action sets an initial default column size for each column and a grouping rule that is appropriate for the type of messages. For example:

- Group the messages by the path of the validated file if there are error messages from a validation action or spelling errors reported by the **Check Spelling in Files** action (on page 458).
- No grouping rule for the results of applying an XPath expression (on page 1999).

**Expand All**

Expands all the nodes of the tree, which is useful when the messages are presented in a hierarchical mode.

**Collapse All**

Collapses all the nodes of the tree, which is useful when the messages are presented in a hierarchical mode.

**Making a Persistent Copy of Results**

The **Results view** (on page 548) displays the results from the following operations:

- Document validation (on page 775)
- Checking the form of documents (on page 773)
- XSLT or FO transformations (on page 1432)
- Finding all occurrences of a string in a file (on page 432)
- Finding all occurrences of a string in multiple files (on page 436)
- Applying an XPath expression to the current document (on page 2002)

To make a persistent copy of the **Results view** (on page 548), use one of these actions:

**File > Save Results**
Displays the **Save Results** dialog box, used to save the result list of the current message tab. The action is also available on the right-click menu of the **Results** panel.

**File > Print Results**

Displays the **Page Setup** dialog box used to define the page size and orientation properties for printing the result list of the current **Results** panel. The action is also available on the right-click menu of the **Results** panel.

**Save Results as XML from the contextual menu**

Saves the content of the **Results** panel in an XML file with the format:

```xml
<Report>
  <Incident>
    <engine>The engine reporting the error.</engine>
    <severity>The severity level</severity>
    <Description>Description of output message.</Description>
    <SystemID>The location of the file linked to the message.</SystemID>
    <Location>
      <start>
        <line>Start line number in file.</line>
        <column>Start column number in file</column>
      </start>
      <end>
        <line>End line number in file.</line>
        <column>End column number in file</column>
      </end>
    </Location>
  </Incident>
</Report>
```

**Related Information:**

[Results View (on page 548)](#)

**CSS Inspector View**

The purpose of the **CSS Inspector** view is to display information about the styles applied to the currently selected element. You can use this view to examine the structure and layout of the CSS rules that match the element. The matching rules displayed in this view include a link to the line in the CSS file that defines the styles. With this tool you can see how the CSS rules were applied and the properties defined, and use the link to open the associated CSS for editing purposes.
Displaying the CSS Inspector View

You can open this view by selecting the **Inspect Styles** action from the contextual menu in **Author** mode, or selecting the **CSS Inspector** view in the **Window > Show View** menu. This action makes the view visible and also initializes it for the currently selected element.

Displaying Rules

All rules that apply to the current element are displayed in sections, which are listed in order of importance (from most specific to least specific). Rules that are overridden by other rules are crossed out. If you click the link in the top-right corner of a rule Oxygen XML Editor opens the associated CSS file at the line number where the properties of the rule are defined.

The **CSS Inspector** view has six possible tabs (depending on the current context):
• **Element** - Displays the CSS rules matching the currently selected element in the **Author** page (ordered from most-specific to least-specific).
• **:marker** - Displays the rules matching the :marker pseudo-element.
• **:before** - Displays the rules matching the :before pseudo-element.
• **:after** - Displays the rules matching the :after pseudo-element.
• **Computed** - Displays all the styling properties that apply to the current element, as a result of all the CSS rules matching the element.
• **Path** - Displays the path for the current element, and its attributes, allowing you to quickly see the attributes on all parent elements, and allows you to copy fragments from this view and paste it into the associated CSS to easily create new rules.

The information displayed in each of the five tabs is updated when you click other elements in the **Author** editing view. The first three tabs include the link to the associated CSS source, while the other two tabs simply display the style properties that match the current element.

Each of the tabbed panes include a contextual menu with the following actions:

• **Copy** - copies the current selection
• **Select all** - selects all information listed in the pane
• **Show empty rules** - forces the view to show all the matching rules, even if they do not declare any CSS properties (by default, the empty rules are not displayed)

**Reviewing Documents**

Oxygen XML Editor includes a variety of helpful review tools that improve your ability to collaborate with other members of your team, track changes, mark content for various reasons, add comments in your content, and to manage the review features.

**Tracking Document Changes**

The **Track Changes feature (on page 3224)** is a way to keep track of the changes you make in a document. The **Track Changes** feature highlights changes that you make to the content in a document, as well as changes to attributes. Changes can be tracked for insertions and deletions. When the **Track Changes feature is activated (on page 644)**, insertions are rendered in **Author** mode with an underline while deletions are rendered with a strike through.

The tracked changes are also displayed in the **Review view (on page 664)** and you can also choose to present the changes in **callouts (on page 658)** by selecting the **Track Changes Deletions (on page 186)** and **Track Changes Insertions (on page 186)** options in the **Callouts preferences page (on page 186)**.

**Adding Comments in Documents**

You can associate a comment to a selected area of content. Comments can highlight virtually any content from your document, with the exception of **read-only** text. The difference between using comments and **change tracking** is that a comment can be associated to an area of text without modifying or deleting the text.
Comments are presented in callouts (on page 658) with persistent highlights and a colored background. The background color is assigned automatically by the application, but it can also be customized from the Review preferences page (on page 183).

**Highlighting Content**

Oxygen XML Editor includes a highlighting feature that allows you to create digital markers to emphasize important fragments of your documents. This is especially useful when you want to mark content that needs additional work or the attention of others.

**Using the Review View**

Oxygen XML Editor includes a Review view (on page 664) that provides a simplified way of monitoring all the insertions, deletions, comments, and highlights in an XML document. This handy tool is especially useful for large teams that need to gather and manage all the edits from all team members who are working on the same project.

The Review view is also useful for managing tracked changes and comments in a single panel. In this view, the changes and comments are presented in a compact form, in the order they appear in the document, and they are synchronized with the changes and comments in the main editing area.

You can use this view to quickly navigate through changes, accept or reject them, or to view and manage comments or highlights. You can also search for specific changes or comments and it includes some filtering options (for example, you can filter it to only show certain types of edits or to only show edits for a particular author).

**Printing Review Information**

When you print a document from Author mode, whatever review information is shown in the main editing area will be included in the printed output. For example, tracked changes will be included and as long as the Comments option (on page 186) is selected in the Callouts preferences page (on page 186), comment callouts will also be included (same with tracked change callouts if their corresponding options are selected in the Callouts preferences page (on page 186).

**Managing Tracked Changes**

Oxygen XML Editor includes a Track Changes feature (on page 3224) that allows you to review changes that you or other authors have made and then accept or reject them. You can also manage the visualization mode of the tracked changes, add comments to changes, and mark them as being done. These actions are easily accessible from contextual menus, the toolbar, or the Review view (on page 664).

The Track Changes feature is also able to keep track of changes you make to attributes in a document and the changes are presented in the Review view (on page 664) and Attributes view (on page 627).

**Types of Tracked Changes**

The types of tracked changes include:
- Inserting, deleting content (text or elements)
- Drag and drop content (text or elements)
- Cutting or pasting content (text or elements)
- Inserting, deleting, and changing the structure of tables
- Inserting and editing lists and their content
- Inserting and deleting entities
- Inserting and deleting element tags
- Editing attributes
- Performing a **Split** operation
- Performing a **Surround with** operation
- Changes in referenced content (for example, XInclude fragments or DITA conrefs)

**Important:**
If you copy content in **Author** mode that contains tracked changes, the changes will automatically be accepted prior to the content being copied to the clipboard. This filtering is performed only if the selection is not entirely inside a tracked change.

**Activating the Change Tracking Feature**

To activate the **Track Changes** feature for the current document, use any of the following methods:

- Click the ![Track Changes](image) button on the toolbar.
- Select ![Track Changes](image) from the **Review** submenu of the contextual menu in the main editing area in **Author** mode.
- Select ![Track Changes](image) from the **Edit > Review** menu.

To activate the **Track Changes** feature globally for all documents that you open in Oxygen XML Editor, change the **Initial State** option to **Always On** (on page 184) in the **Review preferences page** (on page 183).

**Rendering Tracked Changes in Author Mode**

When **Track Changes** (on page 3224) is enabled, your modifications are highlighted using a distinctive color. The colors can be customized from the **Review preferences page** (on page 183), along with the name of the author and the initial state of the feature when you open a document. Insertions are rendered with an underline while deletions are rendered with a strike through.
When hovering over a change a tooltip displays information about the author and modification time.

**Change Tracking Contextual Menu Actions**

You can right-click any change in **Author** mode to access the following contextual menu actions:

- **Accept Change(s)**
  
  Accepts the Tracked Change (on page 3224) located at the cursor position or all of the changes in a selection. If you select a part of a deletion or insertion change, only the selected content is accepted. If you select multiple changes, all of them are accepted. For an insertion change, it keeps the inserted text and for a deletion change, it removes the content from the document.

- **Reject Change(s)**
  
  Rejects the Tracked Change (on page 3224) located at the cursor position or all of the changes in a selection. If you select a part of a deletion or insertion change, only the selected content is rejected. If you select multiple changes, all of them are rejected. For an insertion change, it removes the inserted text and for a deletion change, it preserves the original content.

- **Comment Change**
  
  Opens a dialog box that allows you to add a comment to an existing Tracked Change (on page 3224). The comment will appear in a callout and a tooltip when hovering over the change. If the action is selected on an existing commented change, the dialog box will allow you to edit the comment.
Change Tracking Toolbar Actions

By default, the toolbar includes the following actions and options for reviewing or tracking changes (similar actions are also available in the Edit > Review menu and the Review submenu of the contextual menu):

![Track Changes](image)

Enables or disables the Track Changes (similar actions are also available in the Edit > Review menu and the Review submenu of the contextual menu):

- **Accept Change(s) combo box**

  This combo box is both a button and a drop-down menu that includes the following actions (when you select an action from the drop-down menu, that action becomes the default action for the combo box button):

  - **✓ Accept Change(s) and Move to Next** - Accepts the Tracked Change located at the cursor position or all of the changes in a selection and then moves to the next change. If you select a part of a deletion or insertion change, only the selected content is accepted.
  - **✓ Accept Change(s)** - Accepts the Tracked Change located at the cursor position or all of the changes in a selection.
  - **✓ Accept All Changes** - Accepts all Tracked Changes in the current document.

**Tip:**

You can assign shortcut keys for these actions to make it easier to access them.

- **✓ Reject Change(s) combo box**

  This combo box is both a button and a drop-down menu that includes the following actions (when you select an action from the drop-down menu, that action becomes the default action for the combo box button):

  - **✗ Reject Change(s) and Move to Next** - Rejects the Tracked Change located at the cursor position or all of the changes in a selection and then moves to the next change. If you select a part of a deletion or insertion change, only the selected content is rejected.
  - **✗ Reject Change(s)** - Rejects the Tracked Change located at the cursor position or all of the changes in a selection.
  - **✗ Reject All Changes** - Rejects all Tracked Changes in the current document.
Tip:
You can assign shortcut keys (on page 294) for these actions to make it easier to access them.

Comment Change

Opens a dialog box that allows you to add a comment to an existing Tracked Change (on page 3224). The comment will appear in a callout and a tooltip when hovering over the change. If the action is selected on an existing commented change, the dialog box will allow you to edit the comment.

Track Changes Visualization Modes Drop-Down Menu

This drop-down menu includes specialized actions that allow you to switch between the following visualization modes:

- View All Changes/Comments - This mode is active by default. When you use this mode, all tracked changes are represented in the Author mode.

- View only Changes/Comments by - Only the tracked changes made by the author you select are presented.

- View Final - This mode offers a preview of the document as if all tracked changes (both inserted and deleted) were accepted.

- View Original - This mode offers a preview of the document as if all tracked changes (both inserted and deleted) were rejected. If you attempt to edit the document in this mode, the view mode will switch to View All Changes/Comments.

- Review Settings - Opens the Review Preferences (on page 183) page where the Initial display mode setting can be configured.

Note:
If you use View Final mode and View Original mode, callouts (on page 3217) are not displayed for comments or changes. To display callouts, use the View All Changes/Comments mode.

Highlight

Enables or disables the Highlight tool (on page 656). Use the Highlight drop-down menu to select a new color.

Add Comment

Inserts a comment at the cursor position. The comment appears in a callout box and a tooltip (when hovering over the change).

Show/Edit Comment
Opens a dialog box that displays the discussion thread and allows the current user to edit comments that do not have replies. If you are not the author who inserted the original comment, the dialog box just displays the comment without the possibility of editing it.

Remove Comment

Removes a selected comment. If you remove a comment that contains replies, all of the replies will also be removed.

Manage Reviews

Opens the Review view (on page 664).

Tracked Change Callouts

You can also choose to display insertion and deletion changes in callouts (on page 3217) in Author mode. By default, tracked changes are not displayed in callouts, but you can change this behavior by selecting the Track Changes Deletions (on page 186) and Track Changes Insertions (on page 186) options in the Callouts preferences page (on page 186). You can also choose to display the actual content of the deletion or insertion.

By displaying the changes in callouts, you then have access to even more actions, such as the ability to reply or mark them as being done. For more information, see Author Callouts (on page 658).

Tracked Changes in the Review View

The Review view (on page 664) is also useful for managing tracked changes and comments. In this view, the edits are presented in a compact form, in the order they appear in the document and each edit is marked with a type-specific icon. You can use this view to quickly navigate through changes, accept or reject them, or to add and manage comments for the changes. You can also search for specific changes and it includes some filtering options (for example, you can filter it to only show certain types of changes or to only show changes for a particular author).

For more information, see Review View (on page 664).

Tracked Changes XML Source Code

The changes are stored in the document source code as processing instructions and they do not interfere with validation or transformations. For each change, the author name and the modification time are preserved.

Example - Insertion Change: The following processing instruction is an example of how an insertion change is stored in a document:

```xml
<?oxy_insert_start author="John Doe" timestamp="20090408T164459+0300"?>all<?oxy_insert_end?>
```

Example - Deletion Change: The following processing instruction is an example of how a deletion change is stored in a document:

```xml
<?oxy_delete author="John Doe" timestamp="20090508T164459+0300" content="belong"/>
```
For more information about the Track Changes support, watch our video demonstration:

https://www.youtube.com/embed/L_ESxRMfnek

Related Information:
Managing Comments (on page 653)
Author Callouts (on page 658)
Review View (on page 664)

**Tracked Changes Behavior**

The behavior of the Track Changes feature (on page 3224) depends on the context, the type of change, and whether or not it is activated.

**Inserting Content**

If the Track Changes feature is disabled and you insert content, the following behavior is possible:

- Making an insertion in a *Delete* change results in the change being split in two and the content is inserted without being marked as change.
- Making an insertion in an *Insert* change results in the change being split in two and the content is inserted without being marked as change.
- Making an insertion in regular content results in a regular insertion.

If the Track Changes feature is enabled and you insert content, the following behavior is possible:

- Making an insertion in a *Delete* change results in the change being split in two and the current inserted content appears marked as an INSERT.
- Making an insertion in an *Insert* change results in the following:
  - If the original insertion was made by another user, the change is split in two and the current inserted content appears marked as an INSERT by the current author.
  - If the original *Insert* change was made by the same user, the change is just expanded to contain the inserted content. The creation time-stamp of the previous insert is preserved.
- If inserted in regular content, the current inserted content appears marked as an *Insert* change.

**Surrounding Content**

If the Track Changes feature is enabled and you surround content in a new XML element, the following behavior is possible:

- Making a surround in a *Delete* change results in nothing happening.
- Making a surround in an *Insert* change results in the following:
If the original insertion was made by another user, the change is split in two and the surround operation appears marked as being performed by the current author.

- If the original Insert change was made by the same user, the existing change is just expanded to contain the surrounded content.

- Making a surround in regular content results in the operation being marked as a surround change.

### Deleting Characters

If the Track Changes feature is disabled and you delete content character by character, the following behavior is possible:

- Deleting content in an existing Delete change results in nothing happening.
- Deleting content in an existing Insert change results in the content being deleted without being marked as a deletion and the INSERT change shrinks accordingly.
- Deleting in regular content results in a regular deletion.

If the Track Changes feature is enabled and you delete content character by character, the following behavior is possible:

- Deleting content in an existing Delete change results in the following:
  - If the same author created the Delete change, the previous change is marked as deleted by the current author.
  - If another author created the Delete change, nothing happens.
- Deleting content in an existing Insert change results in the following:
  - If the same author created the Insert change, the content is deleted and the Insert change shrinks accordingly.
  - If another author created the Insert change, the Insert change is split in two and the deleted content appears marked as a Delete change by the current author.
- Deleting in regular content results in the content being marked as a Delete change by the current author.

### Deleting Selections of Content

If the Track Changes feature is disabled and you delete a selection of content, the following behavior is possible:

- If the selection contains an entire Delete change, the change disappears and the content is deleted.
- If the selection intersects with a Delete change (starts or ends in one), it results in nothing happening.
- If the selection contains an entire Insert change, the change disappears and the content is deleted.
- If the selection intersects with an Insert change (starts or ends in one), the Insert change is shrunk and the content is deleted.

If the Track Changes feature is enabled and you delete a selection of content, the following behavior is possible:
• If the selection contains an entire **Delete** change, the change is considered as rejected and then marked as deleted by the current author, along with the other selected content.
• If the selection intersects a **Delete** change (starts or ends in one), the change is considered as rejected and marked as deleted by the current author, along with the other selected content.
• If the selection contains an entire **Insert** change, the following is possible:
  ◦ If the **Insert** is made by the same author, the change disappears and the content is deleted.
  ◦ If the **Insert** is made by another author, the change is considered as accepted and then marked as deleted by the current author, along with the other selected content.
• If the selection intersects an **Insert** change (starts or ends in one), the **Insert** change shrinks and the part of the **Insert** change that intersects with the selection is deleted.

**Deleting Tags**

Assuming you are using any of the **Tag Display Modes (on page 594)** other than No Tags and the **Track Changes** feature is disabled, if you delete a start or end tag, both the start and end tag will be removed, while any content that was inside the element is preserved.

Assuming you are using any of the **Tag Display Modes (on page 594)** other than No Tags and the **Track Changes** feature is enabled, if you delete a start tag of an **inline element (on page 3220)**, both the start and end tag are marked as a **Delete** change by the current author, while any content that was inside the element is preserved.

**Copying Content**

If the **Track Changes** feature is disabled and you copy content, if the copied area contains **Insert** or **Delete** changes (or attribute edits), these are also copied to the clipboard.

If the **Track Changes** feature is enabled and you copy content, if the copied area contains **Insert** or **Delete** changes (or attribute edits), these are all accepted in the content of the clipboard (the changes will no longer be in the clipboard).

**Pasting Content**

If the **Track Changes** feature is disabled and you paste content, if the clipboard content contains **Insert** or **Delete** changes (or attribute edits), they will be preserved on paste.

If the **Track Changes** feature is enabled and you paste content, if the clipboard content contains **Insert** or **Delete** changes (or attribute edits), all the changes are accepted and then the paste operation proceeds according to the insertion rules.

**Tracked Changes Limitations**

There are some inherent limitations to the **Change Tracking (on page 3224)** feature. These limitations include the following:
- **Limitations to rejected changes** - Recording changes has limitations and there is no guarantee that rejecting all changes will return the document exactly to its original state.

- **Limitations to hierarchical changes** - Recorded changes are not hierarchical, a change cannot contain other changes inside. For example, if you delete an insertion made by another user, then reject the deletion, the information about the author who made the previous insertion is not preserved.

- **Limitations to using certain actions** - Some actions cannot be implemented with the *Track Changes feature (on page 3224)* enabled. For example, some table-related actions (>Delete Row(s), Delete Column(s), Join Cells, Split Cell) ignore the *Track Changes* feature. The Rename Element action also does not record tracked changes.

- **Possible Serialization Limitation** - If you have equivalent adjacent tracked changes, for example, you see two back-to-back changes in the Review pane that have identical properties (the same user, timestamp, content, etc.), when you save the document, it is sometimes possible for the document to only contain a single processing instruction.

### Tracked Changes XML Markup

Depending on the type of edits, the following markup appears in the document source code when you activate the *Track Changes feature (on page 3224)*:

<table>
<thead>
<tr>
<th>Edit Type</th>
<th>Processing Instruction Start Marker</th>
<th>Processing Instruction End Marker</th>
<th>Attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Deletion</td>
<td>&lt;?oxy_delete?&gt;</td>
<td></td>
<td>Common Attributes <em>(on page 653)</em>, content</td>
</tr>
<tr>
<td>Comment</td>
<td>&lt;?oxy_comment_start?&gt;</td>
<td>&lt;?oxy_comment_end?&gt;</td>
<td>Common Attributes <em>(on page 653)</em>, mid, parentID</td>
</tr>
<tr>
<td>Attribute Change</td>
<td>&lt;?oxy_attributes?&gt;</td>
<td></td>
<td>Attributes of the <em>Processing Instruction</em> have the name as the attribute that was changed on the <em>XML element</em>. The value is an attribute change descriptor <em>(on page 653)</em>.</td>
</tr>
</tbody>
</table>

If a comment intersects another, the @mid attribute is used to correctly identify start and end processing instruction markers.
Common Attributes

The following attributes can be added on both change tracking and comment processing instructions:

- **@id** - Used to link a reply to its parent comment or change.
- **@comment** - The comment message associated with a comment or change.
- **@timestamp** - The time when the change or comment was created.
- **@author** - The name of the author that created the change or comment.
- **@flag** - The value *done* means that the item is *Marked as Done*.

Attribute Change Descriptor

The value of the attributes for a *Processing Instruction* is an (escaped) XML element as in:

```xml
<change type="modified" oldValue="word" author="John" timestamp="20210520T091038+0000"/>
```

Related information

dita-classic-pdf-review - an open-source project that contains XSLT stylesheets to process the review markup

Managing Comments

You can add comments to any selected area of content within XML documents, with the exception of *read-only* content. The difference between using comments and *tracked changes (on page 3224)* is that a comment is associated to a selection without modifying or deleting the content.

By default, when you annotate your XML documents, the comments are displayed in the *Author* mode as callouts (on page 3217) (balloons) and they are rendered with a unique name and background for each user. If comments are not currently displayed in callouts, select the *Comments option (on page 186)* in the *Callouts preferences page (on page 186)*. Comments are also displayed in the *Review view (on page 664)*.

![Figure 166. Comments in Author Mode](image)

Managing Comments in the Main Editor

You can insert and manage comments directly in the main editing area in *Author* mode.

Add Comment
To insert a comment at the cursor position or on a specific selection of content, select the Add Comment action from the toolbar (or in the Review submenu of the contextual menu).

**Tip:**
When adding or editing a comment, you can use Enter to insert line breaks and Oxygen XML Editor will take the line breaks into account when presenting the callout. You can also use Ctrl + Enter to accept your changes and close the dialog box.

Show/Edit Comments

To edit an existing comment that you have added in the main editing area in Author mode, select the Show/Edit Comments action from the toolbar (or in the Review submenu of the contextual menu). The action opens a dialog box that allows you to see and edit your comment at the cursor position. Note that you cannot edit a comment that was added by another user, so in that case, the dialog box just displays the comment without the possibility of editing it.

Remove Comments

To remove a comment at the cursor position or multiple comments in a selection, select Remove Comment(s) from the toolbar (or in the Review submenu of the contextual menu).

Copy/Paste

If you copy content that includes comments, they will be preserved when you paste it.

Managing Comments in Callouts

As long as the Comments option (on page 186) is selected in the Callouts preferences page (on page 186), comments are also displayed in callouts (on page 658). By displaying the comments in callouts, you then have access to even more actions, such as the ability to reply or mark them as being done. When you right-click a specific comment in its callout, the contextual menu includes the following actions.

**Reply**

Opens a dialog box that allows you to add a reply to a comment or Tracked Changes (on page 3224). When replying to a comment, the dialog box shows the entire conversation in the comment thread, starting with the first comment added in the particular thread, followed by all the replies. After replies are added to a comment thread, they are displayed with an indentation in the callouts and Review view (on page 664).

**Mark as Done**

A toggle action that marks or unmarks a comment or comment thread as being done. It is also available for Tracked Changes (on page 3224) that are displayed in a callout. When a comment or change is marked as done, the callout is grayed out and cannot be edited unless the action is toggled to the unmarked state. The action applies to the particular comment and all of its descendents. This is useful for marking comments or changes that have been addressed, leaving only those that still need some attention.
Opens a dialog box that displays the discussion thread and allows the current user to edit comments that do not have replies. If you are not the author who inserted the original comment, the dialog box just displays the comment without the possibility of editing it.

**Remove Comment**

Removes a selected comment. If you remove a comment that contains replies, all of the replies will also be removed.

**Callouts Options**

Select this option to open the Callouts preference page (on page 186) where you can configure various callout options.

**Tip:**

When adding, editing, or replying to a comment, you can use Enter to insert line breaks and Oxygen XML Editor will take the line breaks into account when presenting the callout. You can also use Ctrl + Enter to accept your changes and close the dialog box.

**Managing Comments in the Review View**

The Review view (on page 664) is also useful for managing comments. In this view, comments are presented in a compact form, in the order they appear in the document, along with tracked changes. You can also use this view to search for specific comments and it includes some filtering options (for example, you can filter it to only show comments for a particular author). When you right-click a specific comment in the Review view, the contextual menu includes the following actions.

**Reply**

Opens a dialog box that allows you to add a reply to a comment or Tracked Changes (on page 3224). When replying to a comment, the dialog box shows the entire conversation in the comment thread, starting with the first comment added in the particular thread, followed by all the replies. After replies are added to a comment thread, they are displayed with an indentation in the callouts and Review view (on page 664).

**Mark as Done**

A toggle action that marks or unmarks a comment or comment thread as being done. It is also available for Tracked Changes (on page 3224) that are displayed in a callout. When a comment or change is marked as done, the callout is grayed out and cannot be edited unless the action is toggled to the unmarked state. The action applies to the particular comment and all of its descendents. This is useful for marking comments or changes that have been addressed, leaving only those that still need some attention.

**Show/Edit Comment**

Opens a dialog box that displays the discussion thread and allows the current user to edit comments that do not have replies. If you are not the author who inserted the original comment, the dialog box just displays the comment without the possibility of editing it.
**Remove Comment**

Removes a selected comment. If you remove a comment that contains replies, all of the replies will also be removed.

**Show only reviews by '<author name>'**

Filters the comments to only show comments for the particular author.

**Remove all Comments**

Removes all comments from the document.

**Comments XML Source Code**

The comments are stored in the document source code as processing instructions that contain information about the author name and the comment time:

```xml
<?oxy_comment_start author="John Doe" timestamp="20090508T164459+0300" comment="Do not change this content"?>
  Important content
<?oxy_comment_end?>
```

Replies to comments are stored in the document source code as a comment (with information about the author name and time), but with a `@parentID` attribute and its value is the same as the `@id` value of the parent comment.

```xml
<?oxy_comment_start author="Tom" timestamp="20160217T102630+0200" comment="We should not forget about recycling the oil and oil filter!" parentID="vws_x4l_1v" mid="4"?>
```

**Related Information:**

- Author Callouts *(on page 658)*
- Review View *(on page 664)*

**Managing Highlights**

Use the ✈️ Highlight tool to mark fragments in your document using various colors. This is especially useful when you want to mark sections that need additional editing or to draw the attention of others to particular content.

**Using the Highlight Tool**

You can find the ✈️ Highlight action on the main toolbar, in the Edit > Review menu, or in the Review submenu of the contextual menu of a document. You can also choose the color to use for the highlight or choose to Stop highlighting from the same menus.

To highlight content, follow these steps:
1. Click the 🟢 Highlight icon on the toolbar.

   **Step Result:** The highlighting mode is on and the cursor changes to a dedicated symbol.

2. Click the small arrow next to the 🟢 Highlight icon and select the color that you want to use for the highlighting.

3. Select the content you want to highlight. To mark multiple parts of a document, press and hold **Ctrl** (Meta on macOS) and select the parts you want to highlight.

4. To exit the highlighting mode, press **Esc**, click the 🟢 Highlight icon, or start editing the document.

To remove highlighting from a document, follow these steps:

1. Either select the text you want to remove highlighting from using your cursor, or press **Ctrl + A** (Command + A on macOS) if you want to select all of the text.

2. Click the small arrow next to the 🟢 Highlight icon and select **No color (erase)**, or right-click the highlighted content and select **Remove highlight(s)**.

3. To exit the highlighting mode, press **Esc**, click the 🟢 Highlight icon, or start editing the document.

   **Note:**

   Oxygen XML Editor preserves the highlighting of a document between working sessions. Also, if you copy content that includes highlights, the highlighting will be preserved when you paste it.

**Review View**

The **Review view** (on page 664) is also useful for managing highlights. In this view, the highlights are presented in a compact form, in the order they appear in the document, along with **tracked changes** (on page 3224) and comments. The following actions are available in the contextual menu of each highlight in the **Review** view:

- **Change Color**
  Allows you to change the color of an existing highlight by selecting the new color from this menu.

- **Remove Highlight**
  Removes the selected highlight.

- **Remove Highlights with the Same Color**
  Removes the selected highlight and all others that have the same color.

- **Remove All Highlights**
  Removes all highlights from the document.
Highlights XML Source Code

The highlights are stored in the document source code as processing instructions that contain information about the color:

```xml
<?oxy_custom_start type="oxy_content_highlight" color="0,128,255"?>
The highlights are stored.<?oxy_custom_end?>
```

For more information about using the Highlight tool, watch our video demonstration:

https://www.youtube.com/embed/RMFyuMfisZY

Related Information:
- Review View (on page 664)

Author Callouts

Oxygen XML Editor uses callouts (on page 3217) to present comments and tracked change (on page 3224) modifications that you or other members of your team have added to the document.

Displaying Callouts in Author Mode

The callouts are displayed in the right side of the editing area in Author mode. They are decorated with a colored border and also have a colored background. The background color is assigned automatically by the application depending on the user who is editing the document and the type of change, but it can also be customized from the Review preferences page (on page 183). This preferences page allows you to configure the colors for tracked change insertions or deletions, and for comments.

You can also choose to use the same color for all changes of that particular type of change, regardless of who makes the change. To do this, select the Fixed option for the particular type of change and choose a color from the color box. If the Automatic option is selected, Oxygen XML Editor automatically assigns a color based upon the Colors for automatic assignment list (on page 186).

The horizontal line that connects the callouts to their corresponding text fragments has the same color as the border. If this horizontal line is not visible, select the Show all connecting lines option (on page 187) in the Callouts preferences page. If you hover over a callout, it is highlighted and a tooltip is displayed that contains additional information.
Oxygen XML Editor displays callouts only if View All Changes/Comments or View Only Changes/Comments by is selected in the Track Changes Visualization Modes drop-down menu. Oxygen XML Editor does not display callouts in View Final and View Original modes.

In some cases, the text you are editing can span into the callouts area. For example, this situation can appear for callouts associated with wide images or space-preserved elements (on page 3224) that contain long fragments (such as a DITA `<codeblock>` element or `<programlisting>` in DocBook). To help you view the text under the covered area, Oxygen XML Editor applies transparency to these callouts. When the cursor is located under a callout, the transparency is enhanced, allowing you to both edit the covered content and access the contextual menu of the editing area.
**Adjusting Callout Width**

To display more of the content in all the callouts in the current document, you can adjust the width by dragging the left side of any of the callouts. This will adjust the width for all comments in the current document. When you end the current editing session, the width of all callouts will revert back to the default value, which is the value of the **Initial Width** option (on page 187) in the **Callouts** preferences page.

You can also adjust the maximum number of lines to be shown in the callouts using the **Text Lines Count Limit** option (on page 187). Note that this does not limit the number of lines in the actual comment. It only limits the number of lines shown without opening or editing it.

**Type of Callouts in Oxygen XML Editor**

Oxygen XML Editor uses callouts to display comments and **Tracked Changes** (on page 3224) that you associate with fragments of the document you are editing. You can choose which types of edits will be shown in callouts by configuring the options in the **Callouts** preferences page (on page 186). You can choose to enable the following types of review callouts:

- **Comment Callouts** - As long as the **Comments** option (on page 186) is selected in the **Callouts preferences page** (on page 186), comments are displayed in callouts. A comment callout contains the name of the author who inserts the callout and the comment itself. You can also select the **Show review time** option (on page 187) to include timestamp information in the comment callouts.

  ![Figure 169. Comment Callouts](image)

  There are several types of comments that can be added in **Author** mode:

  - **Author Review Comments** - Comments that you associate with specific content. To insert this type of comment, select the content and use the **Add Comment** action that is available on the toolbar (or in the **Review** submenu of the contextual menu).

  - **Comments Added to Tracked Changes** - Comments that you add to an already existing tracked change insertion or deletion. To insert this type of comment, right-click the change in the main editor or its callout and select **Comment Change**.

  - **Replies to Comments** - You can use this type of comment to create discussion threads. To insert this type of comment, right-click the change in its callout and select **Reply**. A single callout is presented for a root comment or change and its replies. The replies are displayed with an indentation in the callouts and those that are on the same level are sorted depending on the timestamp.
Tip: When adding, editing, or replying to a comment, you can use Enter to insert line breaks and Oxygen XML Editor will take the line breaks into account when presenting the callout. You can also use Ctrl + Enter to accept your changes and close the dialog box.

- **Tracked Change Deletion Callouts** - As long as the Track Changes Deletions option (on page 186) is selected in the Callouts preferences page (on page 186), deletions that are made while the Track Changes feature is enabled are displayed in callouts. A deletion callout contains the type of callout (Deleted) and the name of the author that made the deletion. You can also select the Show deleted content in callout option (on page 186) to display the actual deleted content in the callout. Additionally, you can select the Show review time option (on page 187) to include timestamp information in the deletion callouts.

- **Tracked Change Insertion Callouts** - As long as the Track Changes Insertions option (on page 186) is selected in the Callouts preferences page (on page 186), insertions that are done while the Track Changes feature is enabled are displayed in callouts. An insertion callout contains the type of callout (Inserted) and the name of the author that inserted the content. You can also select the Show inserted content in callout option (on page 186) to display the actual deleted content in the callout. Additionally,
you can select the **Show review time** option (on page 187) to include timestamp information in the deletion callouts.

**Figure 172. Insertion Callouts**

Callout Contextual Menu Actions

Some useful actions are available when the contextual menu is invoked on a callout. The actions depend on the type of callout.

**Insertion or Deletion Callout Actions**

The following actions are available in the contextual menu of an insertion or deletion callout:

- **Reply**
  
  Opens a dialog box that allows you to add a reply to a comment or Tracked Changes (on page 3224). When replying to a comment, the dialog box shows the entire conversation in the comment thread, starting with the first comment added in the particular thread, followed by all the replies. After replies are added to a comment thread, they are displayed with an indentation in the callouts and Review view (on page 664).

- **Mark as Done**
  
  A toggle action that marks or unmarks a comment or comment thread as being done. It is also available for Tracked Changes (on page 3224) that are displayed in a callout. When a comment or change is marked as done, the callout is grayed out and cannot be edited unless the action is toggled to the unmarked state. The action applies to the particular comment and all of its descendents. This is useful for marking comments or changes that have been addressed, leaving only those that still need some attention.

- **Accept Change**
  
  Accepts the tracked change, removes the callout, and moves to the next change. For an *insertion* change, it keeps the inserted text and for a *deletion* change, it removes the content from the document.

- **Reject Change**
  
  Rejects the tracked change, removes the callout, and moves to the next change. For an *insertion* change, it removes the inserted text and for a *deletion* change, it preserves the original content.

- **Comment Change**
Opens a dialog box that allows you to add a comment to an existing Tracked Change (on page 3224). The comment will appear in a callout and a tooltip when hovering over the change. If the action is selected on an existing commented change, the dialog box will allow you to edit the comment.

**Edit Reference**

If the fragment that contains a callout is a reference, use this option to go to the reference and edit the callout.

**Callouts Options**

Select this option to open the Callouts preference page (on page 186) where you can configure various callout options.

**Comment Callout Actions**

The following options are available in the contextual menu of a comment callout:

**Reply**

Opens a dialog box that allows you to add a reply to a comment or Tracked Changes (on page 3224). When replying to a comment, the dialog box shows the entire conversation in the comment thread, starting with the first comment added in the particular thread, followed by all the replies. After replies are added to a comment thread, they are displayed with an indentation in the callouts and Review view (on page 664).

**Mark as Done**

A toggle action that marks or unmarks a comment or comment thread as being done. It is also available for Tracked Changes (on page 3224) that are displayed in a callout. When a comment or change is marked as done, the callout is grayed out and cannot be edited unless the action is toggled to the unmarked state. The action applies to the particular comment and all of its descendents. This is useful for marking comments or changes that have been addressed, leaving only those that still need some attention.

**Show/Edit Comment**

Opens a dialog box that displays the discussion thread and allows the current user to edit comments that do not have replies. If you are not the author who inserted the original comment, the dialog box just displays the comment without the possibility of editing it.

**Remove Comment**

Removes a selected comment. If you remove a comment that contains replies, all of the replies will also be removed.

**Edit Reference**
If the fragment that contains a callout is a reference, use this option to go to the reference and edit the callout.

**Callouts Options**

Select this option to open the Callouts preference page (on page 186) where you can configure various callout options.

**Printing Callouts**

When you print a document from Author mode, all callouts that you or other authors have added to the document are printed. For a preview of the document and its callouts, go to File > Print Preview.

**Review View**

The Review view (on page 664) is also useful for managing the information in callouts. In this view, changes and comments are presented in a compact form, in the order they appear in the document, and they are synchronized with the changes in the callouts. You can also search for specific changes or comments and it includes some filtering options (for example, you can filter it to only show certain types of edits or to only show edits for a particular author).

For more information, see Review View (on page 664).

To watch our video demonstration about the Callouts support, see our video demonstration:

https://www.youtube.com/embed/kCCWyFqBaUM

**Related Information:**

- Managing Tracked Changes (on page 643)
- Managing Comments (on page 653)
- Review View (on page 664)

**Review View**

The Review view is an independent panel, available both for built-in and custom XML document frameworks (on page 3219). It is designed to offer an enhanced way of monitoring all the changes that you make to a document. This means you can view and manage highlights, comments, and tracked changes (on page 3224) using a single view.

The Review view is useful when you are working with documents that contain large number of edits. The edits are presented in a compact form, in the order they appear in the document. Each type of edit is marked with a specific icon. This view and the editing area are synchronized. When you select an edit listed in the Review view, its corresponding fragment of text is highlighted in the editing area and the reverse is also true.

For example, when you place the cursor inside an area of text marked as inserted, its corresponding edit is selected in the list.
You can use this view to quickly navigate through changes and it includes some useful hover actions and contextual menu actions to help you manage changes, comments, and highlights. You can also search for specific changes or comments and it includes some filtering options (for example, you can filter it to only show certain types of edits or to only show edits for a particular author).

**Figure 173. Review View**

### Activating the Review View

To activate the Review view, do one of the following:

- Click the Manage reviews button on the toolbar.
- Right-click anywhere in a document and select Review > Manage reviews.
- Open it from the Window > Show View menu.

### Review View Toolbar Actions and Settings

The upper part of the view contains a filtering area that allows you to search for specific edits. The filter field also includes a search history drop-down list. The toolbar also includes `Previous` and `Next` navigation buttons and a `Settings` menu button.

**Previous**

Use this button to navigate to the previous review item.

For DITA projects, as long as the Link with Editor toolbar button (on page 2900) is enabled in the DITA Maps Manager, if you reach the first review item in the document, clicking this button will open a dialog box asking if you want to open the previous document (from the current DITA map hierarchy) that contains review items. This default behavior can be changed by choosing one of the options in the When navigating items in the Review view and you reach the last/first item section of the DITA preferences page (on page 273).

**Next**

Use this button to navigate to the next review item.
For DITA projects, as long as the Link with Editor toolbar button (on page 2900) is enabled in the DITA Maps Manager, if you reach the last review item in the document, clicking this button will open a dialog box asking if you want to open the next document (from the current DITA map hierarchy) that contains review items. This default behavior can be changed by choosing one of the options in the When navigating items in the Review view and you reach the last/first item section of the DITA preferences page (on page 273).

Settings

The Settings menu includes the following options:

Show highlights

Controls whether or not the Review view displays the highlighting in your document.

Show comments

Controls whether or not the Review view displays the comments in the document you are editing.

Show track changes

Controls whether or not the Review view displays the inserted and deleted content in your document.

Show reviews in read-only content

Controls whether or not the Review view displays review items from content referenced with a @conref or @conkeyref attribute.

Show review time

Displays the time when the edits from the Review view were made.

Sort by date

Expands to offer the following sorting options: Oldest to newest, Newest to oldest, and No sorting.

Configure review options

Opens the Review preferences page (on page 183) where you can configure various options for review information.

Hover Actions in the Review View

You can use this view to easily manage changes, highlights, and comments that have been added by you or other users. The following actions are available when you hover over the changes in the Review view:

Remove

Available for highlights and comments presented in the Review view and it removes the particular highlight or comment from your document and moves to the next change.
Accept
Available for inserted and deleted content presented in the Review view and it accepts the particular change in your document and moves to the next change.

Reject
Available for inserted and deleted content presented in the Review view and it rejects the particular change in your document and moves to the next change.

Contextual Menu Actions in the Review View
Depending on the type of an edit, the following additional actions are available in the contextual menu of the Review view:

Reply
Opens the Reply dialog box where you can add a reply to comment or change. The replies are displayed with an indentation in this view.

Mark as Done
Toggles the comment or change as being done and grays it out. You can mark a whole discussion thread as being done by selecting the action on the first (parent) comment in the thread.

Show Comment
Available for comments added by other users and you can use this option to view it in a Show comment dialog box.

Edit Comment
Available for comments you have added and you can use this action to edit a comment.

Remove Comment
Use this action to remove the selected comment.

Show only Reviews by '<author name>'
Use this action to filter the edits to only show them for a certain author.

Remove All Comments
Use this action to remove all the comments that appear in the edited document.

Change Color
Available for highlights and it opens a palette where you can choose a new color for the highlighted content.

Remove Highlight
Available for highlights and you can use this action to remove the selected highlight.

Remove Highlights with the Same Color
Available for highlights and you can use this action to remove all the highlights with the same color from the entire document.

**Remove All Highlights**

Available for highlights and you can use this action to remove all the highlights in your document.

**✓ Accept Change**

Accepts the selected change and moves to the next change.

**✗ Reject change**

Rejects the selected change and moves to the next change.

**Comment change**

Available for insertions or deletions and you can use this option to add a comment for the particular change.

**✓ Accept all changes**

Accepts all the changes in the current document.

**✗ Reject all changes**

Rejects all the changes in the current document.

For more information about the Review view, watch our video demonstration:

https://www.youtube.com/embed/W22jkblhl60

**Related Information:**
- Managing Tracked Changes *(on page 643)*
- Managing Comments *(on page 653)*
- Managing Highlights *(on page 656)*
- Author Callouts *(on page 658)*

**Profiling and Conditional Text**

Profiling text is a way to mark blocks of text meant to appear in some renditions of the document but not in others. Conditional text differs from one variant of the document to another, while unconditional text appears in all document versions. For example, you can mark a section of a document that is to be included in a manual to be designated for *expert* users and another section for *novice* users, while unmarked sections are included in all renditions.

**Profiling Attributes and Condition Sets**

Oxygen XML Editor allows you to define values for the profiling attributes and they can be easily managed to filter content in the published output. You can switch between profile sets to see how the edited content
looks like before publishing. You can also conditionally profile parts of a document so that certain parts are displayed when certain profiling conditions are set. You can even customize the colors and styling of how the profiling is displayed in Author mode.

You can use profiling and conditional text to help you create documentation for multiple output scenarios, including:

- Multiple outputs for a series of similar products.
- Multiple outputs for various releases of a product.
- Multiple outputs for various audiences.

This feature helps to reduce the effort for updating and translating your content and provides an easy way to customize the output for various audiences.

**Figure 174. Example: Profiling Content**

Oxygen XML Editor includes a preconfigured set of profiling attribute values for some of the most popular document types. These attributes can be redefined to match your specific needs in the *Attributes and Condition Sets* preferences page (on page 188). You can also define your own profiling attributes and condition sets for each document type (framework (on page 3219)) and your profiling configuration can be shared among content authors through the project file.

For information about creating and editing profiling attributes, see Creating and Editing Profiling Attributes (on page 670) (for information about sharing them, see Sharing Profiling Attribute Configurations (on page 672)/Sharing Profiling Attribute Configurations (on page 3137)).

For information about creating and editing condition sets, see Creating and Editing Profiling Condition Sets (on page 675) (for information about sharing them, see Sharing Condition Set Configurations (on page 677)).

**Related Information:**
Customizing Elements that Wrap Profiled Content (on page 2263)
Creating and Editing Profiling Attributes

Oxygen XML Editor includes support for defining your own profiling attributes, or modifying existing ones, for each particular document type ([framework (on page 3219)]). You can then apply the profiling attributes to content in **Author** mode to see how the profiling will affect the output.

Create or Editing Profiling Attributes

To create or edit profiling attributes for a specific document type, follow these steps:

1. If you are creating a new attribute, make sure the attribute is already defined in the document DTD or schema before continuing with the procedure.
2. Open the Preferences dialog box (**Options > Preferences**) ([on page 124]) and go to **Editor > Edit modes > Author > Profiling/Conditional Text > Attributes and Condition Sets**.

   **Information:**
   The **Profiling Attributes** section ([on page 188]) is used to define the attributes and their values for each document type.

3. To add new attributes and values, click the **New** button at the bottom of the **Profiling Attributes** table. To customize existing attributes and their values, select an attribute and click the **Edit** button.

   **Step Result:** In either case, this opens a **Profiling Attribute** configuration dialog box where you can define attributes that exist in your schema.

**Figure 175. Profiling Attribute Dialog Box**
The following options are available in this dialog box:

**Document type**

Select the document type (**framework (on page 3219)**).

**Tip:**

You can use the * or ? wildcards in this combo box. For example, DITA* would match any document type that starts with "DITA". You can also specify multiple document types by using commas to separate them.

**Attribute name**

The name of the profiling attribute.

**Display name**

This optional field is used for descriptive rendering in profiling dialog boxes.

**Attribute Values Table**

This table displays information about the values for the profiling attribute. You can configure them by using the buttons at the bottom of the table (**New, Edit, Delete**).

The columns are as follows:

- **Value** - The attribute value.
- **Label** - You can specify a label for the attribute value that will be rendered as its name in various components in **Author** mode (**Edit Profiling Attributes** dialog box (on page 673), **Condition Set** dialog box (on page 675), and other UI components where the profiling is shown (on page 679)). If the Label is not specified, the Value will be used as its rendered name.
- **Description** - A description for the attribute value that will be displayed in this table.

**Single value**

Select this option if you want the attribute to only accept a single value.

**Multiple values separated by**

Select this option if you want the attribute to accept multiple values, and you can choose the type of delimiter to use. You can choose between space, comma, and semicolon, or you can enter a custom delimiter in the text field. A custom delimiter must be supported by the specified document type. For example, the DITA document type only accepts spaces as delimiters for attribute values.

4. After defining or configuring the attributes and their values according to your needs, click **OK** to confirm your selections and close the **Profiling Attributes** configuration dialog box.

5. Click **Apply** to save the changes.
Adding Profiling Attribute Values Directly in a Document

You can add values directly to the existing profiling attributes in a document using the In-Place Attributes Editor (on page 610) in Author mode, the Attributes view (on page 627), or in the source code in Text mode. However, this just adds them to the document and does not change the conditional text configuration. If you invoke the Edit Profiling Attributes action (from the contextual menu in Author mode) on the new value, the Profiling Values Conflict dialog box will appear and it includes an Add these values to the configuration action that will automatically add the new value to the particular profiling attribute. It also includes an Edit the configuration action that opens the Attributes and Condition Sets preferences page (on page 188) where you can edit the profiling configuration.

Note:
If the Allow contributing extra profiling attribute values option (on page 189) is not selected in the Attributes and Condition Sets preferences page, the Profiling Values Conflict dialog box will never appear, so this automatically adding value not be possible.

Figure 176. Profiling Values Conflict Dialog Box

Sharing Profiling Attribute Configurations

Your profiling configuration can be shared with other users through a project file. If you select Project Options (on page 3222) at the bottom of the Profiling/Conditional Text preferences page, your configuration is stored in the project file and can be shared with others. For instance, if your project file is saved on a version control system (such as SVN, CVS, or Source Safe) or in a shared folder, your team will have the same option configuration that you stored in the project file.

For more information about sharing project files, see Sharing a Project - Team Collaboration (on page 415).
Applying Profiling Attributes

Profiling attributes are applied on element nodes. You can apply profiling attributes on a text fragment (it will automatically be wrapped into a phrase-type element), on a single element, or on multiple elements at the same time. If there is no selection in your document, the profiling attributes are applied on the element at the cursor position.

To apply a profiling attribute to content in **Author** mode, follow these steps:

1. To apply a profiling attribute to content in **Author** mode, highlight the content and select **Edit Profiling Attributes** from the contextual menu. To profile an entire element, position the cursor inside the element, right-click, and select **Edit Profiling Attributes** (you can also right-click the element in the breadcrumb **(on page 602)** or **Outline** **(on page 539)** view).

   **Step Result:** The **Edit Profiling Attributes** dialog box is displayed and shows all the profiling attributes and their values, as defined for the particular document type (**framework**). If you have a large list of profiling attributes, you can use the text filter field to search for attributes or values, and you can expand or collapse attributes by using the **Expand All** / **Collapse All** buttons to the right of the text filter or the arrow button to the left of the profiling attribute name.

   The attributes and values that appear in the dialog box are determined as follows:

   ◦ If you have defined profiling attribute values **(on page 670)** for the DITA document type in the **Attributes and Condition Sets preferences page** **(on page 188)** and you store them at **project-level** **(on page 3222)**, those values are displayed in the dialog box.

   ◦ If you have defined profiling attribute values **(on page 670)** for the DITA document type in the **Attributes and Condition Sets preferences page** **(on page 188)** and you store them at **global-level** **(on page 3219)**, those values are displayed in the dialog box.

   ◦ Otherwise, a generic default set of profiling attributes and values are available.
2. In the **Edit Profiling Attributes** dialog box, select the checkboxes that correspond to the attribute values you want to apply on the document fragment (on page 3218).

3. Click **OK** to finish the profiling configuration.

**Result:** The attribute names and values selected in the **Edit Profiling Attributes** dialog box are set on the elements contained in the profiled fragment. If you only select a fragment of content (rather than the entire element), this fragment is wrapped in phrase-type elements where the profiling attributes are set.

If the **Show Profiling Attributes** option (on page 679) (available in the Profiling / Conditional Text toolbar menu) is selected, a green border is painted around profiled text in the Author mode and all profiling attributes set on the current element are listed at the end of the highlighted block. To edit the
attributes of a profiled fragment, click one of the listed attribute values. A form control pops up and allows you to add or remove attribute values.

**Figure 178. Profiling Attribute Value Form Control Pop Up**

![Form Control Pop Up](image)

**Related Information:**
- Creating and Editing Profiling Attributes *(on page 670)*
- Creating and Editing Profiling Condition Sets *(on page 675)*
- Applying Profiling Condition Sets *(on page 677)*
- Showing and Filtering Profiled Content in Author Mode *(on page 679)*
- Customizing Colors and Styles for Rendering Profiling in Author Mode *(on page 3147)*

### Creating and Editing Profiling Condition Sets

Multiple profiling attributes can be aggregated into a profiling condition set that allows you to apply more complex filters on the document content. A *Profiling Condition Set* is a very powerful and convenient tool that can be used to preview the content that goes into the published output. For example, an installation manual available in both Windows and Linux variants can be profiled to highlight only the Linux procedures for more advanced users.

**Create Profiling Condition Sets**

To create a new profiling condition set, follow these steps:

1. Open the **Preferences** dialog box *(Options > Preferences)* *(on page 124)* and go to *Editor > Edit modes > Author > Profiling/Conditional Text > Attributes and Condition Sets*.

**Information:**

The *Profiling Condition Sets* section *(on page 189)* is used to define condition sets.
2. To add new condition set, click the **New** button at the bottom of the **Profiling Condition Sets** table. To customize existing condition sets, select an existing condition set and click the **Edit** button.

**Step Result:** In either case, this opens a **Condition Set** configuration dialog box where you can define attributes that exist in your schema.

![Condition Set Configuration Dialog Box](image)

The following options are available in this dialog box:

**Name**

The name of the new condition set.

**Document type**

Select the document type ([framework](on page 3219)) that has profiling attributes defined.

**Use DITAVAL file**

For DITA projects, select this option if you want the **Profiling Condition Set** to reference a **DITAVAL file** ([on page 3155]). You can specify the path by using the text field, its history drop-down, the ➔ **Insert Editor Variables** ([on page 323]) button, or the browsing actions in the ➔ **Browse** drop-down list.
Include the content matching the following conditions

You can select this option to define the combination of attribute values for your condition set by selecting the appropriate checkboxes for the values you want to be included in this particular condition set. If you have defined a lot of profiling attributes, you can use the filter text field to search for specific conditions.

Shortcut key

You can click the Choose button to open a dialog box that allows you to define a shortcut key for this particular condition set. You can then use that shortcut key anytime you want to select this condition set to filter content.

3. After defining or configuring the condition sets according to your needs, click OK to confirm your selections and close the Condition Set configuration dialog box.

4. Click Apply to save the condition set.

Sharing Condition Set Configurations

Your condition set configuration can be shared with other users through a project file. If you select Project Options (on page 3222) at the bottom of the Profiling/Conditional Text preferences page, your configuration is stored in the project file and can be shared with others. For instance, if your project file is saved on a version control system (such as SVN, CVS, or Source Safe) or in a shared folder, your team will have the same option configuration that you stored in the project file.

For more information about sharing project files, see Sharing a Project - Team Collaboration (on page 415).

Related Information:

- Applying Profiling Condition Sets (on page 677)
- Creating and Editing Profiling Attributes (on page 670)
- Applying Profiling Attributes (on page 673)
- Showing and Filtering Profiled Content in Author Mode (on page 679)
- Customizing Colors and Styles for Rendering Profiling in Author Mode (on page 3147)

Applying Profiling Condition Sets

All defined Profiling Condition Sets (on page 675) are available as shortcuts in the Formatting / Conditional Text toolbar menu (on page 679). Select a menu entry to apply the condition set. The filtered content is then grayed-out in the Author mode and Outline view (on page 539). An element is filtered-out when one of its attributes is part of the condition set and its value does not match any of the values covered by the condition set.

EXAMPLE:

Suppose that you have the following document:
### Spray painting

**Short Description:** When paint is applied using a spray nozzle, it is referred to as spray painting.

**Context:**

- The garage is a good place to spray paint.

<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
<th>Audience</th>
<th>Other</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Move the car out of the garage to avoid getting paint on it.</td>
<td>novice</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Place newspaper, cardboard, or a drop-cloth on the garage floor.</td>
<td>expert</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Place the object to be painted on the covered area.</td>
<td>expert</td>
<td>prop2</td>
</tr>
<tr>
<td>4</td>
<td>Follow the directions on the paint can to paint the object.</td>
<td>expert</td>
<td>prop1</td>
</tr>
<tr>
<td>5</td>
<td>Let the paint dry thoroughly before you move the object.</td>
<td>novice</td>
<td>prop1</td>
</tr>
</tbody>
</table>

If you apply the following condition set, it means that you want to filter out the content to only include content profiled with the `expert` value for the `audience` attribute and content that has the `prop1` value for the `other` attribute.
This is how the document looks in Author mode after you apply the condition set:

```
Spray painting

Short Description: When paint is applied using a spray nozzle, it is referred to as spray painting.
Context:

The garage is a good place to spray paint.

Step 1
Move the car out of the garage to avoid getting paint on it. Audience [novice]

Step 2
Place newspaper, cardboard, or a drop-cloth on the garage floor. Audience [expert]

Step 3
Place the object to be painted on the covered area. Audience [expert] Other [prop2]

Step 4
Follow the directions on the paint can to paint the object. Audience [expert] Other [prop1]

Step 5
Let the paint dry thoroughly before you move the object. Audience [novice] Other [prop1]
```

Related Information:
Creating and Editing Profiling Condition Sets (on page 675)
Creating and Editing Profiling Attributes (on page 670)
Applying Profiling Attributes (on page 673)
Showing and Filtering Profiled Content in Author Mode (on page 679)
Customizing Colors and Styles for Rendering Profiling in Author Mode (on page 681)

Showing and Filtering Profiled Content in Author Mode

You can visualize the effects of profiled content in Author mode by using the options in the Profiling/Conditional Text drop-down menu that is located on toolbar. This drop-down menu includes the following filtering options:

**Show Profiling Colors and Styles**

Select this option to show colors and styles for profiled content in Author mode. You can configure the colors and styles or specify whether or not this option is selected by default in the Profiling/Conditional Text > Colors and Styles preferences page (on page 190).

**Show Profiling Attributes**

Select this option to display the values of the profiling attributes at the end of profiled content in Author mode. You can specify whether or not this option is selected by default in the Profiling/Conditional Text preferences page (on page 187).
Show Excluded Content

Controls whether the content filtered out by a particular condition set is hidden or grayed-out in Author mode and the Outline (on page 539) view. When this option is selected and a condition set is selected in this drop-down menu (on page 680), the filtered content is grayed-out. If this option is not selected and a condition set is selected in this drop-down menu (on page 680), the filtered content is hidden. You can specify whether or not this option is selected by default in the Profiling/Conditional Text preferences page (on page 188).

Choose Condition Set (Available if more than 15 condition sets are defined)

This option is available if you have more than 15 conditions sets defined. It opens a dialog box that makes it easier to find and select condition sets that are not displayed in this drop-down menu.

List of Defined Condition Sets

Up to 15 defined condition sets are listed and you can toggle each one of them on to filter the content in Author mode to only show content that will appear in the output for that particular condition set. If there are more than 15 defined condition sets, the rest of them can be accessed in the More submenu or by using the Choose Condition Set option (on page 680) to access a dialog box that presents all of them.

Profiling Settings

Opens the Attributes and Condition Sets preferences page (on page 188) where you can add and edit profiling attributes and condition sets.

Figure 180. Example: Profiling Controls in Author Mode

If the Show Profiling Attributes option is selected, a green border is painted around profiled text in the Author mode. Also, all profiling attributes set on the current element are listed at the end of the highlighted block and
in its tooltip message. To edit the attributes of a profiled fragment, click one of the listed attribute values. A form control pops up and allows you to add or remove attribute values.

**Figure 181. Profiling Attribute Value Form Control Pop Up**

![Form control for profiling attribute values](image)

**Related Information:**
- Creating and Editing Profiling Attributes (*on page 670*)
- Applying Profiling Attributes (*on page 673*)
- Creating and Editing Profiling Condition Sets (*on page 675*)
- Applying Profiling Condition Sets (*on page 677*)
- Customizing Colors and Styles for Rendering Profiling in Author Mode (*on page 681*)

**Customizing Colors and Styles for Rendering Profiling in Author Mode**

By applying profiling colors and styles, you can mark profiled content in **Author** mode so that you can instantly spot differences between multiple variants of the output. This allows you to preview the content that will go into the published output. The excluded text is grayed-out or hidden in **Author** mode, allowing you to easily recognize the differences.
Choosing the right style for a specific profiling attribute is a matter of personal taste, but be aware of the following:

- If the same block of text is profiled with two or more profiling attributes, their associated styles combine. Depending on the styling, this might result in an excessively styled content that may prove difficult to read or work with.
- It is recommended that you only profile the differences. There is no need to profile common content, since excessive profiling can visually pollute the document.
- A mnemonic associated with a style will help you instantly spot differences in the types of content.

**Styling Profiling Attribute Values**

To set colors and styles for profiling attribute values, follow these steps:

1. Select the **Show Profiling Colors and Styles** option (on page 679) from the **Profiling / Conditional Text** toolbar drop-down menu.
2. Select **Profiling Settings** (on page 680) from the **Profiling / Conditional Text** toolbar drop-down menu. This is a shortcut to the **Attributes and Condition Sets** preferences page (on page 188).
3. Go to the **Colors and Styles** preferences page (on page 190) to configure the colors and styling for the profiling attributes.
4. Go to the **Attributes** preferences page (on page 191) to configure how you want the profiling attributes to appear in Oxygen XML Editor.

**Result:** The styling is now applied in the **Author** editing mode and the **Outline view** (on page 539). Also, to help you more easily identify the profiling you want to apply in the current context, the styling is applied in the...
Edit Profiling Attributes dialog box (on page 670) and in the inline form control pop-up that allows you to quickly set the profiling attributes.

**Figure 183. Profiling Attribute Value Form Control Pop Up**

Related Information:
- Creating and Editing Profiling Attributes (on page 670)
- Applying Profiling Attributes (on page 673)
- Creating and Editing Profiling Condition Sets (on page 675)
- Applying Profiling Condition Sets (on page 677)
- Showing and Filtering Profiled Content in Author Mode (on page 679)

### Adding Tables in Author Mode

You can use the **Insert Table** action on the toolbar or from the contextual menu to add a table in various frameworks (on page 3219) (DITA, DocBook, TEI, and XHTML). This opens the **Insert Table** dialog box. Each framework has a different set of options that are available in this dialog box for configuring the properties of the tables. In all cases, Oxygen XML Editor includes some general editing actions for configuring tables in Author mode.

This section explains those general actions and the various configuration options and layouts for tables that are inserted in the most commonly used document types.

### Editing Tables in Author Mode

Oxygen XML Editor provides support for editing data in a tabular form. A variety of features and operations are available for editing tables in Author mode and they include the following:

#### Adjusting Column Width

To adjust the width of a column or table, drag the border of the column or table. The changes you make to a table are committed into the source document. You can also manage table width and column width
specifications from the source document, and some types of tables include a `colspecs` section that appears above the table in **Author** mode that allows you to easily configure some column specifications (such as column width). These column width specifications are supported in fixed, dynamic, and proportional dimensions. The built-in DITA, DocBook, and XHTML **frameworks (on page 3219)** support this feature. The layout of the tables for these document types takes into account the table width and the column width specifications particular to them.

**Figure 184. Resizing a Table Column in Author Mode**

```
col[span:1, width:2*]  
col[span:1, width:0.5*]  
<table>
<thead>
<tr>
<th>Person Name</th>
<th>Age</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jane</td>
<td>26</td>
</tr>
<tr>
<td>Bart</td>
<td>24</td>
</tr>
<tr>
<td>Alexander</td>
<td>22</td>
</tr>
</tbody>
</table>

They are all students of the computer science department
```

**Selecting Columns and Rows**

To select a row or a column of a table, place the mouse cursor above the column or in front of the row you want to select, then click. When hovering the mouse cursor in front of rows or above column headers, the cursor changes to `>` for row selection and to `<` for column selection and that specific row or column is highlighted.

You can use the **Ctrl** and **Shift** keys to select multiple rows.

**Selecting Cells**

To select a cell in a table, press and hold the **Ctrl** key and click anywhere inside the cell. You can also use the **Ctrl** and **Shift** keys to select multiple cells or to deselect cells from a selection. Alternatively, you can click the left corner of a cell (right corner if you are editing an **RTL document (on page 752)**) to select it. The cursor changes to `>` when you hover over the corner of the cell.

You can also select multiple rectangular blocks of cells by using your mouse to select a cell and drag it to expand the selection.

**Drag and Drop**

You can use the drag and drop action to edit the content of a table. You can select a column and drag it to another location in the table you are editing. When you drag a column and hover the cursor over a valid drop position, Oxygen XML Editor decorates the target location with bold rectangles. The same drag and drop action is also available for entire rows or columns by hovering the mouse cursor in front of rows or above column headers, then selecting the row or column and dragging them to the desired location.
Copy/Cut and Paste

In Oxygen XML Editor, you can copy/cut entire rows or columns of the table you are editing and paste the copied columns or rows inside the same table or inside other tables. You can also use the copy or cut actions for tables located in other documents. If you paste a row or column into non-table content, Oxygen XML Editor introduces a new table that contains the fragments of the copied row or column content.

For copied columns, the fragments are introduced starting with the header of the column. Also, if the copied column is from a CALS table, Oxygen XML Editor preserves column width information. This information is then used when you paste the column into another CALS table.

For copied cells, when pasting them into another cell without a selection (the cursor is just placed in the new cell), the copied cells are pasted while preserving their initial order and spacing. If pasting them into a selection of cells, first the content of the selected cells is deleted, then the copied cells are pasted with their initial order and spacing preserved and if there are more cells in the selection than in the copied content, the pasting will repeat the copied cells until the end of the selection.

Deleting Content

To delete the content of a cell, select the cell and press the Delete or Backspace key on your keyboard. If you press Delete or Backspace again, the selected table structure will also be removed.

To delete an entire row or column, place the cursor inside the row or column (or select it) and use the Delete Row(s) or Delete Column(s) actions from the toolbar or contextual menu. This will delete both the content and the table structure for the current row or column.

To delete a selection of multiple rows or columns, select them and use the Delete Row(s) or Delete Column(s) actions from the toolbar or contextual menu. This will delete both the content and the table structure for all rows or columns that exist in the current selection.

Navigating Cells

Along with the normal mouse navigation, you can also navigate between cells by using the arrow keys on your keyboard. By default, when using the arrow keys to navigate between table cells, the cursor jumps from one cell to another. However, if the Quick navigation in tables option (on page 180) is not selected in the Cursor Navigation preferences page, using the arrow keys to navigate between table cells will cause the cursor to navigate between XML nodes, rather than jumping from cell to cell.

Related Information:

Adding Tables in DocBook (on page 686)
Adding Tables in DITA Topics (on page 2986)
Adding Tables in XHTML Documents (on page 710)
Adding Tables in DocBook

You can use the **Insert Table** action on the toolbar or from the contextual menu to add a table in a DocBook document.

DocBook supports two types of tables:

- **CALS** table model - This is used for more advanced functionality.
- **HTML** table model - This is used for inserting a formal (captioned) HTML table.

Inserting a CALS Table Model in DocBook

To insert a **CALS** table model in DocBook documents, select the **Insert Table** action on the toolbar or from the contextual menu. The **Insert Table** dialog box appears. Select **CALS** for the table **Model**. This model allows you to configure a few more properties than the **HTML** model.

**Figure 185. Insert Table Dialog Box - CALS Model**

The dialog box allows you to configure the following options when you select the **CALS** table model:

**Title**

If this checkbox is selected, you can specify a title for your table in the adjacent text box.

**Table Size**

Allows you to choose the number of **Rows** and **Columns** for the table.

**Generate table header**

If selected, an extra row will be inserted at the top of the table to be used as the table header.
Generate table footer

If selected, an extra row will be inserted at the bottom of the table to be used as the table footer.

Column widths

Allows you to specify the type of properties for column widths (@colwidth attribute). You can choose one of the following properties for the column width:

- **proportional** - The width is specified in proportional (relative) units of measure. The proportion of the column is specified in a @colwidth attribute with the values listed as the number of shares followed by an asterisk. The value of the shares is totaled and rendered as a percent. For example, colwidth="1* 2* 3*" causes widths of 16.7%, 33.3%, and 66.7%. When entering content into a cell in one column, the width proportions of the other columns are maintained. If you change the width by dragging a column in Author mode, the values of the @colwidth attribute are automatically changed accordingly. By default, when you insert, drag and drop, or copy/paste a column, the value of the @colwidth attribute is 1*.

- **dynamic** - If you choose this option, the columns are created without a specified width (@colwidth attribute). Entering content into a cell changes the rendered width dynamically. If you change the width by dragging a column in Author mode, a dialog box will be displayed that asks you if you want to switch to proportional or fixed column widths.

- **fixed** - The width is specified in fixed units. By default, the pt unit is inserted, but you can change the units in the colspecs (column specifications) section above the table or in Text mode. The following units are allowed: pt (points), cm (centimeters), mm (millimeters), pi (picas), in (inches).

Frame

Allows you to specify a value for the @frame attribute. It is used to specify where a border should appear in the table. There are a variety of allowed values, as specified in the DocBook CALS table specifications.

Row separator

Specifies whether or not to include row separators (@rowsep attribute). The allowed values are: 0 (no separator) and 1 (include separators).

Column separator

Specifies whether or not to include column separators (@colsep attribute). The allowed values are: 0 (no separator) and 1 (include separators).

Alignment

Specifies the alignment of the text within the table (@align attribute). The allowed values are:

- **left** - Aligns the text to a left position.
- **right** - Aligns the text to a right position.
- **center** - Aligns the text to a centered position.
- **justify** - Stretches the line of text so that it has equal width. Note that this value cannot be rendered in **Author** mode, so you will only see it in the output.
- **char** - Aligns text to the leftmost occurrence of the value specified on the `@char` attribute for alignment.

**Note:**
The options in the **Insert Table** dialog box for DocBook documents are persistent, so changes made in one session will carry over to another.

When you click **Insert**, a CALS table is inserted into your document at the current cursor position.

When you insert a CALS table, you see a link for setting the `<colspecs>` (column specifications) of your table. Click the link to open the controls that allow you to adjust various column properties. Although they appear as part of the **Author mode (on page 354)**, the `colspecs` link and its controls will not appear in your output. They are just there to make it easier to adjust how the columns of your table are formatted.

![Figure 186. CALS Table in DocBook](image)

**Inserting an HTML Table Model**

To insert an **HTML** table model in DocBook documents, select the **Insert Table** action on the toolbar or from the contextual menu. The **Insert Table** dialog box appears. Select **HTML** for the table **Model**.
The dialog box allows you to configure the following options when you select the **HTML** table model:

**Title**

If this checkbox is selected, you can specify a title for your table in the adjacent text box.

**Table Size**

Allows you to choose the number of **Rows** and **Columns** for the table.

**Generate table header**

If selected, an extra row will be inserted at the top of the table to be used as the table header.

**Generate table footer**

If selected, an extra row will be inserted at the bottom of the table to be used as the table footer.

**Column widths**

Allows you to specify the type of properties for column widths (**@width** attribute). You can choose one of the following properties for the column width:

- **proportional** - The width is specified in proportional (relative) units of measure. The proportion of the column is specified in a **@width** attribute (in a `<col>` element) with the values listed as the number of shares followed by an asterisk. The value of the shares is totaled and rendered as a percent. For example, `width="1* 2* 3*"` causes widths of 16.7%, 33.3%, and 66.7%. When entering content into a cell in one column, the width proportions of the other columns are maintained. If you change the width by dragging a column in **Author** mode, the values of the **@width** attribute are automatically changed.
accordingly. By default, when you insert, drag and drop, or copy/paste a column, the value of the `@width` attribute is 1\*.

- **dynamic** - If you choose this option, the columns are created without a specified width. Entering content into a cell changes the rendered width dynamically. If you change the width by dragging a column in **Author** mode, a dialog box will be displayed that asks you if you want to switch to proportional or fixed column widths.

- **fixed** - The width is specified in fixed units. By default, the `pt` unit is inserted, but you can change the units in the section above the table or in **Text** mode. In addition to the standard pixel, percentage, and relative values, this attribute also allows the special form "0\*" (zero asterisk), which means that the width of each column in the group should be the minimum width necessary to hold the contents.

**Frame**

Allows you to specify a value for the `@frame` attribute. It is used to specify where a border should appear in the table. There are a variety of allowed values, as specified in the **DocBook HTML table specifications**.

**Alignment**

Specifies the alignment of the text within the table (`@align` attribute). The allowed values are:

- **left** - Aligns the text to a left position.
- **right** - Aligns the text to a right position.
- **center** - Aligns the text to a centered position.
- **justify** - Stretches the line of text so that it has equal width. Note that this value cannot be rendered in **Author** mode, so you will only see it in the output.
- **char** - Aligns text to the leftmost occurrence of the value specified on the `@char` attribute for alignment.

**Note:**

The options in the **Insert Table** dialog box for DocBook documents are persistent, so changes made in one session will carry over to another.

When you click **Insert**, an HTML style of table is inserted into your document at the current cursor position.

When you insert an HTML table, you see a section above the table that allows you to easily configure some properties without opening the **Table Properties** dialog box. Although this section appears as part of the **Author mode** (on page 354), it will not appear in your output. It is just there to make it easier to adjust how the columns of your table are formatted.

**Editing an Existing Table**

You can edit the structure of an existing table using the table buttons on the toolbar (or in the contextual menu) to add or remove cells, rows, or columns, and to set basic table properties. Additional attributes can be
used to fine-tune the formatting of your tables by using the **Attributes view** *(on page 627)* *(Window > Show View > Attributes)*.

You can also use the **Table Properties (Ctrl + T (Command + T on macOS))** *(on page 2997)* action from the toolbar or contextual menu to **modify many of the properties of the table** *(on page 693)*.

Also, remember that underneath the visual representation, both table models are really just XML. If necessary, you can edit the XML directly by switching to **Text mode** *(on page 353)*.

### DocBook Table Layouts

The DocBook framework *(on page 3219)* supports the following two table model layouts:

- **CALS table model** *(on page 691)*
- **HTML table model** *(on page 691)*

#### CALS Table Model Layout

The **CALS** table model allows for more flexibility and table customization than other models. When choosing a **CALS** table model from the **Insert Table** dialog box, you have access to more configurable properties. The layout of a **CALS** table includes a **colspecs** section that allows you to easily configure some properties without opening the **Table Properties** dialog box. For example, you can change the value of column widths (@colwidth attribute) or the text alignment (@align attribute). Although they appear as part of the **Author mode** *(on page 354)*, the **colspecs** link and its controls will not appear in your output. They are just there to make it easier to adjust how the columns of your table are formatted.

![Sample CALS Table with no specified width and proportional column widths](image)

#### HTML Table Model Layout

Choosing an **HTML** table model from the **Insert Table** dialog box in a DocBook document inserts a formal (captioned) HTML table. The layout of an **HTML** table includes a section above the table that allows you to
easily configure some properties without opening the **Table Properties** dialog box. For example, you can change the value of column widths (\texttt{@width} attribute) or the text alignment (\texttt{@align} attribute). Although these properties appear as part of the **Author mode (on page 354)**, they will not appear in your output. They are just there to make it easier to adjust how the columns of your table are formatted.

**Figure 189. HTML Table in DocBook**

![Sample HTML Table](image)

<table>
<thead>
<tr>
<th>Column 1</th>
<th>Column 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data A</td>
<td>Data B</td>
</tr>
<tr>
<td>Data C</td>
<td>Data C</td>
</tr>
</tbody>
</table>

**Pasting Tables in DocBook**

Tables that are pasted into a DocBook file are automatically converted to the **CALS** model. If you want to overwrite this behavior and instruct Oxygen XML Editor to convert them to **HTML** tables, set the \texttt{docbook.html.table} parameter to \texttt{1}. You can find this parameter in the following stylesheet:

- \texttt{[OXYGEN_INSTALL_DIR]/frameworks/docbook/resources/xhtml2db5Driver.xsl} for DocBook 5
- \texttt{[OXYGEN_INSTALL_DIR]/frameworks/docbook/resources/xhtml2db4Driver.xsl} for DocBook 4

**Table Validation in DocBook**

Oxygen XML Editor reports table layout problems that are detected in manual or automatic validations. The types of errors that may be reported for DocBook table layout problems include:

**CALS Tables**

- A row has fewer cells than the number of columns detected from the table \texttt{@cols} attribute.
- A row has more cells than the number of columns detected from the table \texttt{@cols} attribute.
- A cell has a vertical span greater than the available rows count.
- The number of \texttt{<colspecs>} is different than the number of columns detected from the table \texttt{@cols} attribute.
- The number of columns detected from the table \texttt{@cols} attribute is different than the number of columns detected in the table structure.
- The value of the \texttt{@cols}, \texttt{@rowsep}, or \texttt{@colsep} attributes are not numeric.
- The \texttt{@namest}, \texttt{@nameend}, or \texttt{@colname} attributes point to an incorrect column name.

**HTML Tables**
• A row has fewer cells than the number of table columns.
• The value of the @colspan, @rowspan, or @span attributes are not numeric.
• A cell has a vertical span greater than the available rows count.

Editing Table Properties in DocBook

You can edit the structure of an existing table using the table buttons on the toolbar (or from the contextual menu) to add or remove cells, rows, or columns, and to set basic table properties. Additional attributes can be used to fine-tune the formatting of your tables by using the Attributes view (on page 627) (Window > Show View > Attributes).

You can use the Table Properties (Ctrl + T (Command + T on macOS)) action to modify many of the properties of the table. You can also adjust some of the properties in the specification section above the table.

The Table properties dialog box allows you to set specific properties to the table elements. The options that are available depend on the context and location within the table where the action was invoked.

Note:

Some properties allow the following special values, depending on the context and the current properties or values:

• <not set> - Use this value if you want to remove a property.
• <preserve> - If you select multiple elements that have the same property set to different values, you can choose this value to keep the values that are already set. In some cases it can also be used to keep the current non-standard value for a particular property.

Edit Table Properties for a CALS Table Model

For a CALS table model, the Table properties dialog box includes four tabs of options:

• Table tab - The options in this tab apply to the entire table.
• Row tab - The options in this tab apply to the current row or selection of multiple rows. A message at the bottom of the tab tells you how many rows will be affected.
• Column tab - The options in this tab apply to the current column or selection of multiple columns. A message at the bottom of the tab tells you how many columns will be affected.
• Cell tab - The options in this tab apply to the current cell or selection of multiple cells. A message at the bottom of the tab tells you how many cells will be affected.

The options in four tabs include a Preview pane that shows a representation of the modification.
Figure 190. Table Properties Dialog Box with Cell Tab Selected (DocBook CALS Table Model)

The options in the four tabs include the following:

**Horizontal alignment (Available in the Table, Column, and Cell tabs)**

Specifies the horizontal alignment of text within the current table/column/cell or selection of multiple columns/cells (@align attribute). The allowed values are as follows:

- **left** - Aligns the text to a left position.
- **right** - Aligns the text to a right position.
- **center** - Aligns the text to a centered position.
- **justify** - Stretches the line of text so that it has equal width. Note that this value cannot be rendered in Author mode, so you will only see it in the output.
- **char** - Aligns text to the leftmost occurrence of the value specified on the @char attribute for alignment.

**Vertical alignment (Available in the Row and Cell tabs)**

Specifies the vertical alignment of text within the current row/cell or selection of multiple rows/cells (@align attribute). The allowed values are as follows:

- **top** - Aligns the text at the top of the cell.
- **middle** - Aligns the text in a vertically centered position.
- **bottom** - Aligns the text at the bottom of the cell.

**Column separator (Available in the Table, Column, and Cell tabs)**
Specifies whether or not to include column separators (borders/grid lines) in the form of the @colsep attribute. The allowed values are: 0 (no separator) and 1 (include separators).

**Row separator (Available in all four tabs)**

Specifies whether or not to include row separators (borders/grid lines) in the form of the @rowsep attribute. The allowed values are: 0 (no separator) and 1 (include separators).

**Frame**

Allows you to specify a value for the @frame attribute. It is used to specify where a border should appear in the table. There are a variety of allowed values, as specified in the DocBook CALS table specifications.

**Row type (Available in the Row tab only)**

Allows you change the row to a header, body, or footer type of row (within a @thead, @tbody, or @tfoot attribute).

---

**Edit Table Properties for an HTML Table Model**

For an HTML table model, the Table properties dialog box includes four tabs of options (Table, Row, Column, and Cell) and the options include a Preview pane that shows a representation of the modification.

The options in the four tabs include the following:

**Frame (Available only in the Table tab)**

Allows you to specify a value for the @frame attribute. It is used to specify where a border should appear in the table. There are a variety of allowed values, as specified in the DocBook HTML table specifications.

**Row type (Available in the Row tab only)**

Allows you change the row to a header, body, or footer type of row (within a @thead, @tbody, or @tfoot attribute).

**Horizontal alignment (Available in the Row, Column, and Cell tabs)**

Specifies the horizontal alignment for the text in the current row/column/cell or selection of multiple rows/columns/cells (@align attribute). The allowed values are:

- **left** - Aligns the text to a left position.
- **right** - Aligns the text to a right position.
- **center** - Aligns the text to a centered position.
- **justify** - Stretches the line of text so that it has equal width. Note that this value cannot be rendered in Author mode, so you will only see it in the output.
- **char** - Aligns text to the leftmost occurrence of the value specified on the @char attribute for alignment.

**Vertical alignment (Available in the Row, Column, and Cell tabs)**
Specifies the vertical alignment for the text in the current row/column/cell or selection of multiple rows/columns/cells (@valign attribute). The allowed values are:

- **top** - Aligns the text at the top of the cell.
- **middle** - Aligns the text in a vertically centered position.
- **bottom** - Aligns the text at the bottom of the cell.
- **baseline** - Sets the row so that all the table data share the same baseline. This often has the same effect as the `bottom` value. However, if the fonts are different sizes, the `baseline` value often makes the table look better.

Related Information:
Editing Tables in Author Mode (on page 683)

Adding Tables in DITA Topics

You can use the [Insert Table](#) action on the toolbar or from the contextual menu to add a table in a DITA topic. By default, DITA supports four types of tables:

- **DITA Simple table model (on page 2987)** - This is the most commonly used model for basic tables.
- **CALS table model (OASIS Exchange Table Model) (on page 2988)** - This is used for more advanced functionality.
- **DITA Choice table model (on page 2991)** - This is used within a `<step>` element in a DITA Task document to describe a series of optional choices that a user must make before proceeding.
- **DITA Properties table model (on page 2993)** - This is used in DITA Reference documents to describe a property (for example, its type, value, and description).

If you are using a specialized DITA vocabulary, it may contain specialized versions of these table models.

Since DITA is a structured format, you can only insert a table in places in the structure of a topic where tables are allowed. The Oxygen XML Editor toolbar provides support for entering and editing tables. It also helps to indicate where you are allowed to insert a table or its components by disabling the appropriate buttons.

**Inserting a Simple Table Model**

To insert a Simple DITA table, select the [Insert Table](#) action on the toolbar or from the contextual menu (or the Table submenu from the DITA menu). The Insert Table dialog box appears. Select **Simple** for the table Model.
The dialog box allows you to configure the following options when you select the **Simple** table model:

**Title**

If this checkbox is selected, you can specify a title for your table in the adjacent text box.

**Generate table header**

If selected, an extra row will be inserted at the top of the table to be used as the table header.

**Column widths**

Allows you to specify the type of properties for column widths (`@colwidth` attribute). You can choose one of the following properties for the column width:

- **proportional** - The width is specified in proportional (relative) units of measure. The proportion of the column is specified in a `@relcolwidth` attribute with the values listed as the number of shares followed by an asterisk. The value of the shares is totaled and rendered as a percent. For example, `relcolwidth="1* 2* 3*"` causes widths of 16.7%, 33.3%, and 66.7%. When entering content into a cell in one column, the width proportions of the other columns are maintained. If you change the width by dragging a column in **Author** mode, the values of the `@relcolwidth` attribute are automatically changed accordingly. By default, when you insert, drag and drop, or copy/paste a column, the value of the `@relcolwidth` attribute is `1*`.

- **dynamic** - If you choose this option, the columns are created without a specified width. Entering content into a cell changes the rendered width dynamically. If you change the width by dragging a column in **Author** mode, a dialog box will be displayed that asks you if you want to switch to proportional or fixed column widths.
Frame

Allows you to specify a value for the @frame attribute. It is used to specify where a border should appear in the table. The allowed values are as follows:

- **none** - No border will be added.
- **all** - A border will be added to all frames.
- **top** - A border will be added to the top frame.
- **topbot** - A border will be added to the top and bottom frames.
- **bottom** - A border will be added to the bottom frame.
- **sides** - A border will be added to the side frames.
- **-dita-use-conref-target** - Normally, when using a @conref, the values of attributes specified locally are preserved. You can choose this option to override this behavior and pull the value of this particular attribute from the @conref target. For more information, see [https://www.oxygenxml.com/dita/1.3/specs/langRef/attributes/ditauseconreftarget.html](https://www.oxygenxml.com/dita/1.3/specs/langRef/attributes/ditauseconreftarget.html).

**Note:**

The options in the Insert Table dialog box for DITA documents are persistent, so changes made in one session will carry over to another.

When you click Insert, a simple table is inserted into your document at the current cursor position.

**Inserting a CALS Table Model (OASIS Exchange Table)**

To insert an OASIS Exchange Table (CALS), select the Insert Table action on the toolbar or from the contextual menu (or the Table submenu from the DITA menu). The Insert Table dialog box appears. Select CALS for the table Model. This model allows you to configure more properties than the Simple model.
The dialog box allows you to configure the following options when you select the **CALS** table model:

**Title**

If this checkbox is selected, you can specify a title for your table in the adjacent text box.

**Table Size**

Allows you to choose the number of **Rows** and **Columns** for the table.

**Generate table header**

If selected, an extra row will be inserted at the top of the table to be used as the table header.

**Column widths**

Allows you to specify the type of properties for column widths (**@colwidth** attribute). You can choose one of the following properties for the column width:

- **proportional** - The width is specified in proportional (relative) units of measure. The proportion of the column is specified in a **@colwidth** attribute with the values listed as the number of shares followed by an asterisk. The value of the shares is totaled and rendered as a percent. For example, `colwidth="1* 2* 3*"` causes widths of 16.7%, 33.3%, and 66.7%. When entering content into a cell in one column, the width proportions of the other columns are maintained. If you change the width by dragging a column in **Author** mode, the values of the **@colwidth** attribute are automatically changed accordingly. By default, when you insert, drag and drop, or copy/paste a column, the value of the **@colwidth** attribute is `1*`.

- **dynamic** - If you choose this option, the columns are created without a specified width (**@colwidth** attribute). Entering content into a cell changes the rendered width dynamically.
If you change the width by dragging a column in **Author** mode, a dialog box will be displayed that asks you if you want to switch to proportional or fixed column widths.

- **fixed** - The width is specified in fixed units. By default, the `pt` unit is inserted, but you can change the units in the `colspecs` (column specifications) section above the table or in **Text** mode. The following units are allowed: `pt` (points), `cm` (centimeters), `mm` (millimeters), `pi` (picas), `in` (inches).

**Frame**

Allows you to specify a value for the `@frame` attribute. It is used to specify where a border should appear in the table. The allowed values are as follows:

- **none** - No border will be added.
- **all** - A border will be added to all frames.
- **top** - A border will be added to the top frame.
- **topbot** - A border will be added to the top and bottom frames.
- **bottom** - A border will be added to the bottom frame.
- **sides** - A border will be added to the side frames.
- **-dita-use-conref-target** - Normally, when using a `@conref`, the values of attributes specified locally are preserved. You can choose this option to override this behavior and pull the value of this particular attribute from the `@conref` target. For more information, see [https://www.oxygenxml.com/dita/1.3/specs/langRef/attributes/ditauseconreftarget.html](https://www.oxygenxml.com/dita/1.3/specs/langRef/attributes/ditauseconreftarget.html).

**Row separator**

Specifies whether or not to include row separators (`@rowsep` attribute). The allowed values are: 0 (no separator) and 1 (include separators).

**Column separator**

Specifies whether or not to include column separators (`@colsep` attribute). The allowed values are: 0 (no separator) and 1 (include separators).

**Alignment**

Specifies the alignment of the text within the table (`@align` attribute). The allowed values are:

- **left** - Aligns the text to a left position.
- **right** - Aligns the text to a right position.
- **center** - Aligns the text to a centered position.
- **justify** - Stretches the line of text so that it has equal width.

**Note:**

The `justify` value cannot be rendered in **Author** mode, so you will only see it in the output.
• **char** - Aligns text to the leftmost occurrence of the value specified on the `@char` attribute for alignment.

• **-dita-use-conref-target** - Normally, when using a `@conref`, the values of attributes specified locally are preserved. You can choose this option to override this behavior and pull the value of this particular attribute from the `@conref` target. For more information, see [https://www.oxygenxml.com/dita/1.3/specs/langRef/attributes/ditauseconreftarget.html](https://www.oxygenxml.com/dita/1.3/specs/langRef/attributes/ditauseconreftarget.html).

---

**Note:**

The options in the **Insert Table** dialog box for DITA documents are persistent, so changes made in one session will carry over to another.

When you click **Insert**, a CALS table is inserted into your document at the current cursor position.

When you insert a CALS table, you see a link for setting the `colspecs` (column specifications) of your table. Click the link to open the controls that allow you to adjust various column properties. Although they appear as part of the **Author mode** (on page 354), the `colspecs` link and its controls will not appear in your output. They are just there to make it easier to adjust how the columns of your table are formatted.

**Figure 193. CALS Table in DITA**

![Sample CALS Table with Fixed Width](https://www.oxygenxml.com/dita/1.3/specs/langRef/attributes/ditauseconreftarget.html)

---

**Inserting a Choice Table Model**

To insert a **Choice** table within a `<step>` element in a DITA Task document, select the **Insert Table** action on the toolbar or in the **Insert** submenu from the contextual menu (or the **Table** submenu from the DITA menu), or select **choicetable** from the **Content Completion Assistant** (on page 3217). The **Insert Table** dialog box appears. Select **Simple** for the table **Model**.
The dialog box allows you to configure the following options when you insert a *Choice* table model within a DITA Task:

**Table Size**

Allows you to choose the number of *Rows* and *Columns* for the table.

**Generate table header**

If selected, an extra row will be inserted at the top of the table to be used as the table header.

**Column widths**

Allows you to specify the type of properties for column widths ([@colwidth](#) attribute). You can choose one of the following properties for the column width:

- **proportional** - The width is specified in proportional (relative) units of measure. The proportion of the column is specified in a [@relcolwidth](#) attribute with the values listed as the number of shares followed by an asterisk. The value of the shares is totaled and rendered as a percent. For example, `relcolwidth="1* 2* 3*"` causes widths of 16.7%, 33.3%, and 66.7%. When entering content into a cell in one column, the width proportions of the other columns are maintained. If you change the width by dragging a column in *Author* mode, the values of the [@relcolwidth](#) attribute are automatically changed accordingly. By default, when you insert, drag and drop, or copy/paste a column, the value of the [@relcolwidth](#) attribute is `1*`.

- **dynamic** - If you choose this option, the columns are created without a specified width. Entering content into a cell changes the rendered width dynamically. If you change the width by dragging a column in *Author* mode, a dialog box will be displayed that asks you if you want to switch to proportional or fixed column widths.

**Frame**

Allows you to specify a value for the [@frame](#) attribute. It is used to specify where a border should appear in the table. The allowed values are as follows:

- **none** - No border will be added.
- **all** - A border will be added to all frames.
• **top** - A border will be added to the top frame.
• **topbot** - A border will be added to the top and bottom frames.
• **bottom** - A border will be added to the bottom frame.
• **sides** - A border will be added to the side frames.
• **-dita-use-conref-target** - Normally, when using a `@conref`, the values of attributes specified locally are preserved. You can choose this option to override this behavior and pull the value of this particular attribute from the `@conref` target. For more information, see [https://www.oxygenxml.com/dita/1.3/specs/langRef/attributes/ditauseconreftarget.html](https://www.oxygenxml.com/dita/1.3/specs/langRef/attributes/ditauseconreftarget.html).

When you click **Insert**, a *Choice* table is inserted into your DITA Task document at the current cursor position (within a `<step>` element).

### Inserting a Properties Table Model

To insert a `Properties` table within a `<refbody>` element in a DITA Reference document, select the ✉️ **Insert Table** action on the toolbar or in the **Insert** submenu from the contextual menu (or the **Table** submenu from the **DITA** menu), or select **properties(wizard)** from the **Content Completion Assistant** *(on page 3217)*. The **Insert Table** dialog box appears. Select **Properties** for the table **Model**.

![Insert Table Dialog Box - Properties Model](image)

The dialog box allows you to configure the following options when you insert a `Properties` table model within a DITA Reference:

**Table Size**

Allows you to choose the number of **Rows** and **Columns** for the table.

**Generate table header**
If selected, an extra row will be inserted at the top of the table to be used as the table header.

**Frame**

Allows you to specify a value for the `@frame` attribute. It is used to specify where a border should appear in the table. The allowed values are as follows:

- **none** - No border will be added.
- **all** - A border will be added to all frames.
- **top** - A border will be added to the top frame.
- **topbot** - A border will be added to the top and bottom frames.
- **bottom** - A border will be added to the bottom frame.
- **sides** - A border will be added to the side frames.
- **-dita-use-conref-target** - Normally, when using a `@conref`, the values of attributes specified locally are preserved. You can choose this option to override this behavior and pull the value of this particular attribute from the `@conref` target. For more information, see https://www.oxygenxml.com/dita/1.3/specs/langRef/attributes/ditauseconreftarget.html.

When you click **Insert**, a **Properties** table is inserted into your DITA Reference document at the current cursor position (within a `<refbody>` element).

**Editing an Existing Table**

You can edit the structure of an existing table using the table buttons on the toolbar (or in the contextual menu) to add or remove cells, rows, or columns, and to set basic table properties. Additional attributes can be used to fine-tune the formatting of your tables by using the **Attributes view** (on page 627) (Window > Show View > Attributes). See the DITA documentation for a full explanation of these attributes.

You can also use the ![Table Properties](Ctrl + T (Command + T on macOS)) (on page 2997) action from the toolbar or contextual menu (or DITA menu) to modify many of the properties of the table (on page 2997).

Also, remember that underneath the visual representation, both table models are really just XML. If necessary, you can edit the XML directly by switching to **Text mode** (on page 353).

You can use normal copy/paste shortcuts to move content between cells. Oxygen XML Editor includes a **Smart Paste feature** (on page 613) that preserves certain style and structure information when pasting content.

**Tip:**

When copying a multiple selection of table cells and pasting them outside the table, a new table will be created. When pasting into space-preserved elements, the cell content will be pasted as plain text.

**Related Information:**

Editing Tables in Author Mode (on page 683)
DITA Table Layouts

Depending on the context, DITA accepts the following table layouts:

- *CALS table model* (on page 705)
- *Simple table model* (on page 705)
- *Choice table model* (on page 706)
- *Properties table model* (on page 706)

**CALS Table Model Layout**

The **CALS** table model allows for more flexibility and table customization than other models. When choosing a **CALS** table model from the **Insert Table** dialog box, you have access to more configurable properties. The layout of a **CALS** table includes a **colspecs** section that allows you to easily configure some properties without opening the **Table Properties** dialog box. For example, you can change the value of column widths (**@colwidth** attribute) or the text alignment (**@align** attribute). Although they appear as part of the **Author mode** (on page 354), the **colspecs** link and its controls will not appear in your output. They are just there to make it easier to adjust how the columns of your table are formatted.

**Simple Table Model Layout**

When choosing a **Simple** table model from the **Insert Table** dialog box, you only have access to configure a few properties. For example, you can choose the number of rows and columns, specify values for frames, and choose from a few types of properties for the column width. The layout of this type of table is very simple, as the name suggests.
Choice Table Model Layout

A Choice table model is used within a `<step>` element in a DITA Task document to describe a series of optional choices that a user must make before proceeding. The `<choicetable>` element is a useful device for documenting options within a single step of a task. You can insert Choice tables in DITA Task documents either by selecting `choicetable` from the Content Completion Assistant (on page 3217) (within a `<step>` element) or by using the Insert Table action on the toolbar or from the contextual menu). The options and layout of a Choice table is similar to the Simple table model.

Figure 198. DITA Choice Table

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Opt A</td>
<td></td>
</tr>
<tr>
<td>Opt B</td>
<td></td>
</tr>
<tr>
<td>Opt C</td>
<td></td>
</tr>
</tbody>
</table>

Properties Table Model Layout

A Properties table model is used within a `<refbody>` element in a DITA Reference document to describe a property (for example, its type, value, and description). You can insert Properties tables in DITA Reference documents either by selecting `properties(wizard)` from the Content Completion Assistant (on page 3217) (within a `<refbody>` element) or by using the Insert Table action on the toolbar (or from the contextual menu) and selecting Properties for the Model. The layout of a Properties table is very simple. It allows for a maximum of 3 columns (typically for property type, value, and description) and the only options available are for whether or not you want a header row and for specifying frames (borders).

Figure 199. DITA Properties Table

<table>
<thead>
<tr>
<th>Property Type</th>
<th>Property Value</th>
<th>Property Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>B</td>
<td>C</td>
</tr>
</tbody>
</table>

Table Validation in DITA

Oxygen XML Editor reports table layout problems that are detected in manual or automatic validations. When you validate a DITA map (on page 3218) with the Validate and Check for Completeness action, if the Report table layout problems option (on page 2946) is selected in the DITA Map Completeness Check dialog box, table layout problems will be reported in the validation results. The types of errors that may be reported for DITA table layout problems include:

- **CALS Tables**
  - A row has fewer cells than the number of columns detected from the table `@cols` attribute.
  - A row has more cells than the number of columns detected from the table `@cols` attribute.
  - A cell has a vertical span greater than the available rows count.
• The number of `<colspecs>` is different than the number of columns detected from the table @cols attribute.
• The number of columns detected from the table @cols attribute is different than the number of columns detected in the table structure.
• The value of the @cols, @rowsep, or @colsep attributes are not numeric.
• The @namest, @nameend, or @colname attributes point to an incorrect column name.

Simple or Choice Tables

A row has fewer cells than the number of table columns.

Editing Table Properties in DITA

To customize the look of a table in DITA, place the cursor anywhere in a table and invoke the Table Properties (Ctrl + T (Command + T on macOS)) action from the toolbar or the Table submenu of the contextual menu (or DITA menu). This opens the Table properties dialog box.

The Table properties dialog box allows you to set specific properties to the table elements. The options that are available depend on the context and location within the table where the action was invoked.

**Note:**

Some properties allow the following special values, depending on the context and the current properties or values:

- `<not set>` - Use this value if you want to remove a property.
- `<preserve>` - If you select multiple elements that have the same property set to different values, you can choose this value to keep the values that are already set. In some cases it can also be used to keep the current non-standard value for a particular property.

Edit Table Properties for a CALS Table Model

For a CALS table model, the Table properties dialog box includes four tabs of options:

- **Table tab** - The options in this tab apply to the entire table.
- **Row tab** - The options in this tab apply to the current row or selection of multiple rows. A message at the bottom of the tab tells you how many rows will be affected.
- **Column tab** - The options in this tab apply to the current column or selection of multiple columns. A message at the bottom of the tab tells you how many columns will be affected.
- **Cell tab** - The options in this tab apply to the current cell or selection of multiple cells. A message at the bottom of the tab tells you how many cells will be affected.

The options in four tabs include a Preview pane that shows a representation of the modification.
The options in the four tabs include the following:

**Horizontal alignment (Available in the Table, Column, and Cell tabs)**

Specifies the horizontal alignment of text within the current table/column/cell or selection of multiple columns/cells (@align attribute). The allowed values are as follows:

- **left** - Aligns the text to a left position.
- **right** - Aligns the text to a right position.
- **center** - Aligns the text to a centered position.
- **justify** - Stretches the line of text so that it has equal width.

**Note:**

The justify value cannot be rendered in Author mode, so you will only see it in the output.

- **char** - Aligns text to the leftmost occurrence of the value specified on the @char attribute for alignment.
- **-dita-use-conref-target** - Normally, when using a @conref, the values of attributes specified locally are preserved. You can choose this option to override this behavior and pull the value of this particular attribute from the @conref target. For more information, see [https://www.oxygenxml.com/dita/1.3/specs/langRef/attributes/ditauseconreftarget.html](https://www.oxygenxml.com/dita/1.3/specs/langRef/attributes/ditauseconreftarget.html).

**Vertical alignment (Available in the Row and Cell tabs)**
Specifies the vertical alignment of text within the current row/cell or selection of multiple rows/cells (@valign attribute). The allowed values are as follows:

- **top** - Aligns the text at the top of the cell.
- **middle** - Aligns the text in a vertically centered position.
- **bottom** - Aligns the text at the bottom of the cell.
- **-dita-use-conref-target** - Normally, when using a @conref, the values of attributes specified locally are preserved. You can choose this option to override this behavior and pull the value of this particular attribute from the @conref target. For more information, see https://www.oxygenxml.com/dita/1.3/specs/langRef/attributes/ditauseconreftarget.html.

**Column separator (Available in the Table, Column, and Cell tabs)**

Specifies whether or not to include column separators (borders/grid lines) in the form of the @colsep attribute. The allowed values are: 0 (no separator) and 1 (include separators).

**Row separator (Available in all four tabs)**

Specifies whether or not to include row separators (borders/grid lines) in the form of the @rowsep attribute. The allowed values are: 0 (no separator) and 1 (include separators).

**Frame (Available only in the Table tab)**

Allows you to specify a value for the @frame attribute. It is used to specify where a border should appear in the table. The allowed values are as follows:

- **none** - No border will be added.
- **all** - A border will be added to all frames.
- **top** - A border will be added to the top frame.
- **topbot** - A border will be added to the top and bottom frames.
- **bottom** - A border will be added to the bottom frame.
- **sides** - A border will be added to the side frames.
- **-dita-use-conref-target** - Normally, when using a @conref, the values of attributes specified locally are preserved. You can choose this option to override this behavior and pull the value of this particular attribute from the @conref target. For more information, see https://www.oxygenxml.com/dita/1.3/specs/langRef/attributes/ditauseconreftarget.html.

**Edit Table Properties for a Simple, Choice, or Properties Table Model**

For a Simple, Choice, Properties table model, the Table properties dialog box only allows you to edit a few options.

**Table tab**

**Frame**

Allows you to specify a value for the @frame attribute. It is used to specify where a border should appear in the table. The allowed values are as follows:
• **none** - No border will be added.
• **all** - A border will be added to all frames.
• **top** - A border will be added to the top frame.
• **topbot** - A border will be added to the top and bottom frames.
• **bottom** - A border will be added to the bottom frame.
• **sides** - A border will be added to the side frames.
• **-dita-use-conref-target** - Normally, when using a `@conref`, the values of attributes specified locally are preserved. You can choose this option to override this behavior and pull the value of this particular attribute from the `@conref` target. For more information, see https://www.oxygenxml.com/dita/1.3/specs/langRef/attributes/ditauseconreftarget.html.

**Row tab (not available for Properties tables)**

**Row type**

Allows you change the row to a body or header type of row.

**Related Information:**

* Adding Tables in DITA Topics *(on page 2986)*
* Editing Tables in Author Mode *(on page 683)*

**Adding Tables in XHTML Documents**

You can use the ✉️ Insert Table action on the toolbar or from the contextual menu to add a table in an XHTML document. This action opens the Insert Table dialog box.

**Figure 201. Insert Table Dialog Box in XHTML**

The dialog box allows you to configure the following options:

**Caption**
If this checkbox is selected, you can specify a title (caption) for your table in the adjacent text box.

**Table Size**

Allows you to choose the number of Rows and Columns for the table.

**Generate table header**

If selected, an extra row will be inserted at the top of the table to be used as the table header.

**Generate table footer**

If selected, an extra row will be inserted at the bottom of the table to be used as the table footer.

**Column widths**

Allows you to specify the type of properties for column widths (@width attribute). You can choose one of the following properties for the column width:

- **proportional** - The width is specified in proportional (relative) units of measure. The proportion of the column is specified in a @width attribute (in a <col> element) with the values listed as the number of shares followed by an asterisk. The value of the shares is totaled and rendered as a percent. For example, width="1* 2* 3*" causes widths of 16.7%, 33.3%, and 66.7%. When entering content into a cell in one column, the width proportions of the other columns are maintained. If you change the width by dragging a column in **Author** mode, the values of the @width attribute are automatically changed accordingly. By default, when you insert, drag and drop, or copy/paste a column, the value of the @width attribute is 1*.

- **dynamic** - If you choose this option, the columns are created without a specified width. Entering content into a cell changes the rendered width dynamically. If you change the width by dragging a column in **Author** mode, a dialog box will be displayed that asks you if you want to switch to proportional or fixed column widths.

- **fixed** - The width is specified in fixed units. By default, the pt unit is inserted, but you can change the units in the section above the table or in **Text** mode. In addition to the standard pixel, percentage, and relative values, this attribute also allows the special form “0*” (zero asterisk), which means that the width of each column in the group should be the minimum width necessary to hold the contents.

**Frame**

Allows you to specify a value for the @frame attribute. It is used to specify where a border should appear in the table. There are a variety of allowed values, as specified in HTML specifications.

**Alignment**

Specifies the alignment of the text within the table (@align attribute). The allowed values are:

- **left** - Aligns the text to a left position.
- **right** - Aligns the text to a right position.
- **center** - Aligns the text to a centered position.
• **justify** - Stretches the line of text so that it has equal width. Note that this value cannot be rendered in **Author** mode, so you will only see it in the output.
• **char** - Aligns text to the leftmost occurrence of the value specified on the @char attribute for alignment.

**Note:**
The options in the **Insert Table** dialog box for XHTML documents are persistent, so changes made in one session will carry over to another.

When you click **Insert**, an HTML style of table is inserted into your XHTML document at the current cursor position.

When you insert an HTML table, you see a link for setting the `<colspecs>` (column specifications) of your table. Click the link to open the controls that allow you to adjust various column properties. Although they appear as part of the **Author mode (on page 354)**, the colspecs link and its controls will not appear in your output. They are just there to make it easier to adjust how the columns of your table are formatted.

**Editing an Existing Table**

You can edit the structure of an existing table using the table buttons on the toolbar (or in the contextual menu) to add or remove cells, rows, or columns, and to set basic table properties. Additional attributes can be used to fine-tune the formatting of your tables by using the **Attributes view (on page 627)** (**Window > Show View > Attributes**). Also, remember that underneath the visual representation, the table is really just XML. If necessary, you can edit the XML directly by switching to **Text mode (on page 353)**.

**XHTML Table Layout**

When you insert a table in an XHTML document, an HTML type of table is added. The layout of an XHTML table includes a colspecs section that allows you to easily configure some properties. For example, you can change the value of column widths (@width attribute) or the text alignment (@align attribute). Although they appear as part of the **Author mode (on page 354)**, the colspecs link and its controls will not appear in your output. They are just there to make it easier to adjust how the columns of your table are formatted.

**Figure 202. Table Layout in XHTML Documents**
Table Validation in XHTML

Oxygen XML Editor reports table layout problems that are detected in manual or automatic validations. The types of errors that may be reported for XHTML table layout problems include:

**HTML Tables**

- A row has fewer cells than the number of table columns.
- The value of the `@colspan`, `@rowspan`, or `@span` attributes are not numeric.
- A cell has a vertical span greater than the available rows count.

Adding Tables in TEI Documents

You can use the ✉️ **Insert Table** action on the toolbar or from the contextual menu to add a table in a TEI document. This action opens the **Insert Table** dialog box.

**Figure 203. Insert Table Dialog Box in TEI**

The dialog box allows you to configure the following options:

**Head**

If this checkbox is selected, you can specify a title for your table in the adjacent text box.

**Table Size**

Allows you to choose the number of **Rows** and **Columns** for the table.

**Generate table header**

If selected, an extra row will be inserted at the top of the table to be used as the table header.

**Note:**

The options in the **Insert Table** dialog box for TEI documents are persistent, so changes made in one session will carry over to another.

When you click **Insert**, a simple table is inserted into your TEI document at the current cursor position.
Editing an Existing Table

You can edit the structure of an existing table using the table buttons on the toolbar (or in the contextual menu) to add or remove cells, rows, or columns. Additional attributes can be used to fine-tune the formatting of your tables by using the Attributes view (on page 627) (Window > Show View > Attributes). Also, remember that underneath the visual representation, the table is really just XML. If necessary, you can edit the XML directly by switching to Text mode (on page 353).

Adding Tables in JATS Documents

You can use the Insert Table action on the toolbar or from the contextual menu to add a table in a JATS document. This action inserts a simple HTML-type table into your document at the current cursor position. Once inserted, you can edit the structure of the table using the table buttons on the toolbar (or in the contextual menu). For example you can add or remove cells, rows, and columns, or split or join cells. Additional attributes can be used to fine-tune the formatting of your tables by using the Attributes view (on page 627) (Window > Show View > Attributes).

Also, remember that underneath the visual representation, the table is really just XML. If necessary, you can edit the XML directly by switching to Text mode (on page 353).

Sorting Content in Tables and List Items

Oxygen XML Editor offers support for sorting the content of tables and list items of ordered and unordered lists.

Sorting a Table

To sort rows in a table, select the entire table (or specific rows) and use the Sort action from the main toolbar or the contextual menu. This opens the Sort dialog box.

Figure 204. Sort Dialog Box
This dialog box sets the range that is sorted and the sorting criteria. The range is automatically selected depending on whether you sort an entire table or only a selection of its rows.

**Note:**
When you invoke the sorting operation over an entire table, the **Selected rows** option is disabled.

The **Criteria** section specifies the sorting criteria (a maximum of three sorting criteria are available), defined by the following:

- A name, which is collected from the column heading.
- The type of the information that is sorted. You can choose between the following:
  - **Text** - Alphanumeric characters.
  - **Numeric** - Regular integer or floating point numbers are accepted.
  - **Date** - Default date and time formats from the local OS are accepted (such as `short`, `medium`, `long`, `full`, `xs:date`, and `xs:dateTime`).
- The sorting direction (either **ascending** or **descending**).

The sort criteria is automatically set to the column where the cursor is located at the time when the sorting operation is invoked.

After you finish configuring the options in the **Sort** dialog box, click **OK** to complete the sorting operation.

If you want to revert to the initial order of your content, press **Ctrl + Z (Command + Z on macOS)** on your keyboard.

**Note:**
The sorting support takes the value of the `@xml:lang` attribute into account and sorts the content in a natural order.

### Sorting a Selection of Rows

To sort a selection of rows in a table, select the rows that you want to sort and either right-click the selection and choose **Sort**, or click **Sort** on the main toolbar. This opens the **Sort** dialog box.
This dialog box sets the range that is sorted and the sorting criteria. The range is automatically selected depending on whether you sort an entire table or only a selection of its rows.

The Sort dialog box also allows you to apply the sorting operation to the entire table, using the All rows option.

The Criteria section specifies the sorting criteria (a maximum of three sorting criteria are available), defined by the following:

- A name, which is collected from the column heading.
- The type of the information that is sorted. You can choose between the following:
  - Text - Alphanumeric characters.
  - Numeric - Regular integer or floating point numbers are accepted.
  - Date - Default date and time formats from the local OS are accepted (such as short, medium, long, full, xs:date, and xs:dateTime).
- The sorting direction (either ascending or descending).

The sort criteria is automatically set to the column where the cursor is located at the time when the sorting operation is invoked.

After you finish configuring the options in the Sort dialog box, click OK to complete the sorting operation. If you want to revert to the initial order of your content, press Ctrl + Z (Command + Z on macOS) on your keyboard.

**Note:**

The sorting support takes the value of the @xml:lang attribute into account and sorts the content in a natural order.
Sort Using Multiple Criteria

You can also sort an entire table or a selection of its rows based on multiple sorting criteria. To do so, select the rest of boxes in the Criteria section of the Sort dialog box, configure the applicable items, and click OK to complete the sorting operation.

Figure 206. Sorting Based on Multiple Criteria

![Sort dialog box]

Sorting a Table that Contains Merged Cells

If a table contains cells that span over multiple rows, you can not perform the sorting operation over the entire table. Still, the sorting mechanism works over a selection of rows that do not contain rowspans.

Note:
For this type of table, the Sort dialog box keeps the All rows option disabled even if you perform the sorting operation over a selection of rows.

Sorting List Items

A sorting operation can be performed on various types of lists and list items. The types of lists that can be sorted in Oxygen XML Editor depend on the framework (document type), but examples of the types that can be sorted include:

- Ordered list (DITA, DocBook, XHTML, TEI)
- Unordered list (DITA, DocBook, XHTML, TEI)
- Definition list (DITA)
- Variable list (DocBook)
- Parameter list (DITA)
- Simple list (DITA)
- Required conditions (DITA Machinery Task)
- Supplies list (DITA Machinery Task)
- Spare parts list (DITA Machinery Task)
- Safety conditions (DITA Machinery Task)

The sorting mechanism works on an entire list or on a selection of list items. To sort items in a list, select the items or list and use the **Sort** action from the main toolbar or the contextual menu. This opens the **Sort** dialog box.

**Figure 207. Sorting List Items**

![Sort dialog box](image)

This dialog box sets the range that is sorted and the sorting criteria. The range is automatically selected depending on whether you sort an entire list or only a selection of its items.

**Note:**
When you invoke the sorting operation over an entire list, the **Selected rows** option is disabled.

The **Criteria** section specifies the sorting criteria, defined by the following:

- The name of the type of item being sorted.
- The type of the information that is sorted. You can choose between the following:
  - **Text** - Alphanumeric characters.
  - **Numeric** - Regular integer or floating point numbers are accepted.
  - **Date** - Default date and time formats from the local OS are accepted (such as `short`, `medium`, `long`, `full`, `xs:date`, and `xs:dateTime`).
- The sorting direction (either **ascending** or **descending**).

After you finish configuring the options in the **Sort** dialog box, click **OK** to complete the sorting operation. If you want to revert to the initial order of your content, press **Ctrl + Z (Command + Z on macOS)** on your keyboard.

**Note:**
The sorting support takes the value of the `@xml:lang` attribute into account and sorts the content in a natural order.
Inserting Images

To insert an image in a document while editing in Author mode, use one of the following methods:

- Click the [Insert Image] action from the toolbar. This opens a dialog box that allows you to choose the image file you want to insert and configure some properties. Oxygen XML Editor tries to reference the image with a path that is relative to that of the document you are currently editing. For example, if you want to add the `file:/C:/project/xml/dir/img1.jpg` image into the `file:/C:/project/xml/doc1.xml` document, Oxygen XML Editor inserts a reference to `dir/img1.jpg`. This is useful when multiple users work on a common project and they have it stored in multiple locations.

  ![Note](image-note.png)

  The [Insert Image] action is available for the following document types: DITA, DocBook, TEI, XHTML, JATS.

- Drag an image from other application and drop it in the Author editing mode. If it is an image file, it is inserted as a reference to the image file. For example, in a DITA topic the path of the image file is inserted as the value of the `@href` attribute in an `<image>` element:

  ```xml
  <image href="../images/image_file.png"/>
  ```

  **Tip:**

  To replace an image, just drag and drop a new image over the existing one. Oxygen XML Editor will automatically update the reference to the new image.

- Copy an image file from another document or another application (such as a system file browser or web browser) and paste it into your document. Oxygen XML Editor will insert it as a reference to the image file the same as the drag/drop method.

- Select an image (or part of an image) from another application (such as an image editor), copy it, and paste it into your document. Oxygen XML Editor will prompt you to save it. After saving the image, a reference to that file path is inserted at the paste position.

**Related Information:**

- Image Map Editor ([on page 725](#))
- Image Rendering in Author Mode ([on page 719](#))
- Adding Video, Audio, and Embedded HTML Resources in DITA Topics ([on page 2977](#))

**Image Rendering in Author Mode**

The Author mode and the output transformation process might render the images referenced in an XML document differently, since they use different rendering engines.
### Table 5. Supported Image Formats

<table>
<thead>
<tr>
<th>Image Type</th>
<th>Support</th>
<th>Additional Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>GIF</td>
<td>built-in</td>
<td>Animations not yet supported.</td>
</tr>
<tr>
<td>JPG, JPEG</td>
<td>built-in</td>
<td>JPEG images with CMYK color profiles (<a href="#">on page 2875</a>) are properly rendered only if color profile is inside the image.</td>
</tr>
<tr>
<td>PNG</td>
<td>built-in</td>
<td></td>
</tr>
<tr>
<td>SVG, SVGZ, WMF</td>
<td>built-in</td>
<td>Rendered using the open-source Apache Batik library that supports SVG 1.1.</td>
</tr>
<tr>
<td>BMP</td>
<td>built-in</td>
<td></td>
</tr>
<tr>
<td>TIFF</td>
<td>built-in</td>
<td>Rendered using a part of the Java JAI Image library.</td>
</tr>
<tr>
<td>EPS</td>
<td>built-in</td>
<td>Renders the preview TIFF image inside the EPS.</td>
</tr>
<tr>
<td>AI</td>
<td>built-in</td>
<td>Renders the preview image inside the Adobe Illustrator file.</td>
</tr>
<tr>
<td>PDF</td>
<td>built-in</td>
<td>Rendered by using the bundled Apache PDF Box library.</td>
</tr>
</tbody>
</table>

When an image cannot be rendered, Oxygen XML Editor Author mode displays a warning message that contains the reason why this is happening. Possible causes include the following:

- The image is too large. Select the **Show very large images** option ([on page 177](#)).
- The image format is not supported by default. It is recommended to install the [Java Advanced Imaging (JAI) Image I/O Tools plug-in](#) (on page 722).

**Tip:**

If you are using a custom XML format and you want images to be displayed in Author mode, you could use a custom CSS to define the rendering.

For example, if your XML has something like this:

```
<image href="blue.png"/>
```

You can add a selector in your custom CSS like this:

```css
image[href]{
    content: attr(href, url);
}
```
Scaling Images

Image dimension and scaling attributes are taken into account when an image is rendered. The following rules apply:

- If you specify only the width attribute of an image, the height of the image is proportionally applied.
- If you specify only the height attribute of an image, the width of the image is proportionally applied.
- If you specify width and height attributes of an image, both of them control the rendered image.
- If you want to scale both the width and height of an image proportionally, use the `<scale>` attribute.

Note:
As a Java application, Oxygen XML Editor uses the Java Advanced Imaging API that provides a pluggable support for new image types. If you have an ImageIO library that supports additional image formats, just copy this library to the `[OXYGEN_INSTALL_DIR]/lib` directory.

Rendering CGM Images

Oxygen XML Editor offers a few add-ons that provide support for CGM 1.0 images. To allow the rendering of CGM images in Author mode, follow this procedure:

1. Go to Help > Install new add-ons to open an add-on selection dialog box.
2. Enter or paste `https://www.oxygenxml.com/InstData/Addons/default/updateSite.xml` in the Show add-ons from field or select it from the drop-down menu.
3. Select the Oxygen CGM support add-on and click Next.
4. Select the I accept all terms of the end user license agreement option and click Finish.
5. Restart the application.

Result: You should be able to see CGM images in Author mode.

Note:
If you want to render CGM images in your PDF output, make sure you add support for this image format in the transformation scenario (on page 1996).

Rendering PDF Images

Oxygen XML Editor provides built-in support for rendering PDF images in Author mode and PDF output.

Rendering PSD Images

Oxygen XML Editor provides support for rendering PSD (Adobe Photoshop) images.

To allow the rendering of PSD images in Author mode, follow this procedure:
1. Download the following JAR (on page 3220) files:
   - http://search.maven.org/remotecontent?filepath=com/twelvemonkeys/common/common-image/3.1.0/common-image-3.1.0.jar
   - http://search.maven.org/remotecontent?filepath=com/twelvemonkeys/imageio/imageio-metadata/3.1.0/imageio-metadata-3.1.0.jar

2. Copy the downloaded JAR libraries to the \{OXYGEN_INSTALL_DIR\}\lib directory.

3. Restart the application.

Rendering EPS and AI Images

Most EPS and AI image files include a preview picture of the content. Oxygen XML Editor tries to render this preview picture. The following scenarios are possible:

- The EPS or AI image does not include the preview picture. Oxygen XML Editor cannot render the image.
- The EPS image includes a TIFF preview picture.

   Note:
   Some newer versions of the TIFF picture preview are rendered in gray-scale.

- The AI image contains a JPEG preview picture. Oxygen XML Editor renders the image correctly.

Rendering Special Images with Java Advanced Imaging (JAI) Plugin

Certain special image types can be rendered in Oxygen XML Editor by using a Java Advanced Imaging (JAI) Image I/O Tools plugin.

How to Install JAI Image I/O Tools Plugin

To install this plugin, follow this procedure:

1. Start Oxygen XML Editor and open the Help > About dialog box. Go to the System properties tab and look for the java.runtime.name and java.home properties. Keep their values for later use.

2. Download the JAI Image I/O kit corresponding to your operating system and Java distribution (found in the java.runtime.name property). A list of archived JAI distributions can be found at: http://www.oracle.com/technetwork/java/javasebusiness/downloads/java-archive-downloads-java-client-419417.html.
Note:
The JAI API is not the same thing as JAI Image I/O. Make sure you have installed the latter.

3. Run the installer. When the installation wizard displays the Choose Destination Location page, fill-in the Destination Folder field with the value of the java.home property. Continue with the installation procedure and follow the on-screen instructions.

macOS Workaround

There is no native implementation of the JAI Image I/O Tools plugin for macOS 10.5 and later. However, it has a Java implementation fallback that also works on macOS. Some of the image formats are not fully supported in this fallback mode, but at least the TIFF image format is known to be supported.

Use the following procedure for this macOS workaround:

2. In the [OXYGEN_INSTALL_DIR]/lib directory, create a directory named endorsed ([OXYGEN_INSTALL_DIR]/lib/endorsed).
3. Unpack the .tar.gz. Copy the clibwrapper_jiio.jar and jai_imageio.jar files from its lib directory and paste them in the [OXYGEN_INSTALL_DIR]/lib/endorsed directory.
4. Restart the application and the JAI Image I/O support will be up and running.

Rendering EMF Images

By default, Oxygen XML Editor does not render EMF images. However, a plugin is available that provides support for rendering EMF images in the Author visual editing mode. You make this work, you need to install the plugin and some additional libraries.

How to Install Support for Rendering EMF Images

To install support for rendering EMF images in Author mode, follow these steps:

1. Go to https://github.com/oxygenxml/oxygenxml.emf.support and download the project as a ZIP file.
2. Extract the contents of the ZIP file to the [OXYGEN_INSTALL_DIR]/plugins directory.
4. Unpack the ZIP file and move all the .jar libraries from its lib folder to the [OXYGEN_INSTALL_DIR]/plugins/lib directory.
Retina/HiDPI Images in Author Mode

Oxygen XML Editor provides support for Retina and HiDPI images through simple naming conventions. The higher resolution images are stored in the same images folder as the normal resolution images and they are identified by a scaling factor that is included in the name of the image files. For instance, images with a Retina scaling factor of 2 will include \texttt{@2x} in the name (for example, myImage@2x.png). Oxygen XML Editor displays the larger set of icons starting with 150\% (1.5x) scaling.

You can reference an image to style an element in a CSS by using the \texttt{url} function in the \texttt{content} property, as in the following example:

```css
listItem:before{
    content: url('..//img/myImage.png');
}
```

This would place the image that is loaded from the myImage.png file just before the \texttt{<listItem>} element. However, if you are using a Retina display (on a Mac), the icon looks a bit blurry as it automatically gets scaled, or if you are using an HiDPI display (on a Windows-based PC), the icon remains at the original size, thus it will look very small. To solve this rendering problem, you need to be able to reference both a normal DPI image and a high DPI image. However, referencing both of them from the CSS is not practical, as there is no standard way of doing this.

Starting with version 17, Oxygen XML Editor interprets the argument of the \texttt{url} function as key rather than a fixed URL. Therefore, when running on a system with a Retina or HiDPI display, Oxygen XML Editor will first try to find the image file that corresponds to the retina scaling factor. For instance, using the previous example, Oxygen XML Editor would first try to find myImage@2x.png. If this file is not found, it defaults back to the normal resolution image file (myImage.png).

Oxygen XML Editor also supports dark color themes. This means that the background of the editor area can be of a dark color and the foreground a lighter color. On a dark background, you may find it useful to invert the colors of images. Again, this can be done with simple naming conventions. If an image designed for a dark background is not found, the normal image is used.

**Retina/HiDPI Naming Convention**

Refer to the following table for examples of the Retina/HiDPI image naming convention that is used in Oxygen XML Editor:

<table>
<thead>
<tr>
<th>Color Theme</th>
<th>Referred Image File</th>
<th>Double Density Image File</th>
<th>Triple Density Image File</th>
</tr>
</thead>
<tbody>
<tr>
<td>normal</td>
<td>../img/myImage.png</td>
<td>../img/myImage@2x.png</td>
<td>../img/myImage@3x.png</td>
</tr>
<tr>
<td>dark</td>
<td>../img/myImage_dark.png</td>
<td>../img/myImage_dark@2x.png</td>
<td>../img/myImage_dark@3x.png</td>
</tr>
</tbody>
</table>
Image Map Editor

Oxygen XML Editor includes an Image Map Editor that allows you to create hyperlinks in specific areas of an image that will link to various destinations. For example, an image that is a map of the seven continents may have a specific hyperlink for each continent that links to a resource that has information about the particular continent. The main purpose of an image map is to provide an easy way of linking various parts of an image without having to divide the image into separate image files.

The support for image maps in Oxygen XML Editor is available for images in DITA, DocBook, TEI, and XHTML document types (frameworks (on page 3219)). To create an image map on an existing image and open the Image Map Editor, right-click the image and select Image Map Editor.

Image Maps in DITA

Oxygen XML Editor includes support for image maps in DITA documents through the use of the `<imagemap>` element. This feature provides an easy way to create hyperlinks in various areas within an image without having to divide the image into separate image files. The visual Author editing mode includes an Image Map Editor that helps you to easily create and configure image maps.
Image Map Editor Interface in DITA

The interface of the Image Map Editor consists of the following sections and actions:

**Toolbar**

- **New Rectangle**
  
  Use this button to draw a rectangular shape over an area in the image. You can drag any of the four points to adjust the size and shape of the rectangle.

- **New Circle**
  
  Use this button to draw a circle over an area in the image. You can drag any of the four points to adjust the size of the circle.

- **New Polygon**
  
  Use this button to draw a polygon shape over an area in the image. This action opens a dialog box that allows you to select the number of points for the polygon. You can drag any of the points to adjust the size and shape of the polygon.

- **New Free Form Shape**
  
  Use this button to draw a free form shape over an area in the image. After selecting this button, left-click anywhere in the image to place the first point of your shape.
Then move the cursor to the location of the next desired point and left-click to place the next point, and so on. To complete the shape (area), click the first point again and a line will automatically be added from the last point that was added, or simply double-click the last point to automatically add the line from the last point back to the first.

**Duplicate**

Use this button to create a duplicate of the currently selected shape.

**Delete**

Use this button to delete the currently selected shape.

**Undo**

Use this button to undo the last action.

**Redo**

Use this button to redo the last action that was undone.

**Show/Hide Numbers**

Use this button to toggle between showing or hiding the numbers for the shapes.

**Bring Shape to Front**

Use this button to bring the currently selected shape forward to the top layer.

**Bring Shape Forward**

Use this button to bring the currently selected shape forward one layer.

**Send Shape Backward**

Use this button to send the currently selected shape back one layer.

**Send Shape to Back**

Use this button to send the currently selected shape back to the bottom layer.

**Color Chooser**

Use this drop-down menu to select a color scheme for the lines and numbers of the shapes.

**Zoom Slider**

Use this slider to zoom the image in or out in the main image pane.

**Image Pane**

This main Image Pane is where you work with shapes to add hyperlinks to multiple areas within an image. The editing mechanisms that are supported in the Image Pane include the following:
Mouse Controls and Keyboard Shortcuts

• Use the mouse to select and move shapes around in the image pane. It is easy to see which shape is selected in this image pane because the border of the selected shape changes from a solid line to a dotted one.
• You can also drag any of the points of a selected shape to adjust its size and shape.
• You can hold down the Ctrl key to select multiple shapes and then move them simultaneously.
• You can also move shapes by using the arrow keys on your keyboard. In addition, you can hold down Shift while using the arrow keys to move the shape further or Alt to move it 1 pixel at a time.
• To zoom in or out, you can use the NumPad + or NumPad - keys respectively. Use Ctrl + NumPad 0 to reset the zoom level to its default value.
• You can use Ctrl + Z to undo an action or Ctrl + Y to redo the last action that was undone.

Contextual Menu Actions Available in the Image Pane

You can right-click the shapes, points, or anywhere in the Image Pane to invoke the contextual menu where the following actions are available:

+ Add Point
   Adds a point to Polygon or Free Form shapes.

× Remove Point
   Removes the current point from Polygon or Free Form shapes.

☑ Duplicate
   Create a duplicate of the currently selected shape.

× Delete
   Delete the currently selected shape.

☐ New Rectangle
   Creates a rectangular shape over an area in the image. You can drag any of the four points to adjust the size and shape of the rectangle.

☐ New Circle
   Creates a circle over an area in the image. You can drag any of the four points to adjust the size of the circle.

☐ New Polygon
Creates a polygon shape over an area in the image. This action opens a dialog box that allows you to select the number of points for the polygon. You can drag any of the points to adjust the size and shape of the polygon.

Undo

Use this action to undo the last action.

Redo

Use this action to redo the last action that was undone.

Shape Table

The table at the right of the Image Pane is a sequential list of all the areas (shapes) that have been added in the image. It shows their number, type, and description (if one has been added). If you select one of the entries in the table, the corresponding shape will be selected in the Image Pane.

Properties

Type

Displays information about the selected coordinate.

Target

Allows you to choose the target resource that you want the selected area (shape) to be linked to. Select a target by using the Link drop-down menu to the right of the text field. You can choose between the following types of links: Cross Reference, File Reference, or Web Link. All three types will open a dialog box that allows you to define the target resource. This linking process is similar to the normal process of inserting links in DITA (on page 3073) by using the identical Link drop-down menu from the main toolbar.

When you click OK to finalize your changes in the Image Map Editor, an element will be inserted with either an @href attribute or a @keyref attribute. Additional attributes may also be inserted and their values depend on the target and the type of link. For details about the three types of links and their dialog boxes, see Inserting a Link in Oxygen XML Editor (on page 3073).

Description

You can enter an optional description for the selected area (shape) that will be displayed in the Image Map Details section (on page 730) in Author mode and as a tooltip message when the end-user hovers over the hyperlink in the output.

How to Create an Image Map in DITA

To create an image map on an existing image in a DITA document, follow these steps:
1. Right-click the image and select **Image Map Editor**.

   **Step Result:** This action will apply an image map to the current image and open the **Image Map Editor** dialog box.

2. Add hyperlinks to the image by selecting one of the shape buttons ( □ New Rectangle, ○ New Circle, or △ New Polygon).

3. Move the shape to the desired area in the image and drag any of the points on the shape to adjust its size or form. You can use the other buttons on the toolbar (on page 726) to adjust its layer and color, or to perform other editing actions.

   **Tip:**
   You can right-click any of the points, shapes, or anywhere in the Image Pane to access various helpful contextual menu actions (on page 728). For example, the easiest way to remove a point is to right-click the point and select ✗ Remove Point.

4. With the shape selected, use one of the linking options (on page 729) in the ☞ Link drop-down menu to select a target resource (or enter its path in the Target (on page 729) text field).

5. (Optional) Enter a Description (on page 729) for the selected area (shape).

6. If you want to add more hyperlinks to the image, select a shape button again and repeat the appropriate steps.

7. When you are finished creating hyperlinks, click **OK** to process your changes.

   **Result:** The image map is applied on the image and the appropriate elements and attributes are automatically added. In Author mode, the image map is now rendered over the image. If the image includes an `<alt>` element, its value will be displayed under the image. The following two buttons will also now be available under the image in Author mode:

   - **Image Map Editor** - Click this button to open the Image Map Editor.
   - **Image Map Details** - Click this button to expand a section that displays the details of the image map and allows you to change the shape and coordinates of the hyperlinked areas. Keep in mind that if you change the shape in this section, you also need to add or remove coordinates to match the requirements of the new shape.

**Figure 210. Image Map Details**

![Image Map Details](image)
How to Edit an Existing Image Map in DITA

To edit an existing image map, use any of the following methods:

- Simply double-click the image.
- Right-click the image and select Image Map Editor.
- Click the Image Map Editor button below the image.

All three methods open the Image Map Editor where you can make changes to the image map using the various features described above. You can also make changes to the XML structure of the image map in the Text editing mode.

You can also click the Image Map Details button below the image to expand a section that displays the details of the image map and allows you to change the shape and coordinates of the hyperlinked areas. Keep in mind that if you change the shape in this section, you also need to add or remove coordinates to match the requirements of the new shape.

Overlapping Areas

If shapes overlap one another in the Image Map Editor, the one on the top layer takes precedence. The number shown inside each shape represents its layer (if the numbers are not displayed, click the Show/Hide Numbers button on the Image Map Editor toolbar (on page 726)). To change the layer order for a shape, use the layer buttons on the Image Map Editor toolbar (on page 726) (፣ফ, , ,).

If you insert a shape and all of its coordinates are completely inside another shape, the Image Map Editor will display a warning to let you know that the shape is entirely covered by a bigger shape. Keep in mind that if a shape is completely inside another shape, its hyperlink will only be accessible if its layer is on top of the bigger shape.

Related Information:

DITA `imagemap` Element Specifications

Working with Images in DITA Topics (on page 2973)

Image Maps in DocBook

Oxygen XML Editor includes support for image maps in DocBook documents through the use of the `<areaspec>` element. This feature provides an easy way to create hyperlinks in various parts of an image without having to divide the image into separate image files. The visual Author editing mode includes an Image Map Editor that helps you to easily create and configure image maps.
Image Map Editor Interface in DocBook

The interface of the **Image Map Editor** consists of the following sections and actions:

**Toolbar**

- **New Rectangle**
  
  Use this button to draw a rectangular shape over an area in the image. You can drag any of the four points to adjust the size and shape of the rectangle.

- **Duplicate**
  
  Use this button to create a duplicate of the currently selected shape.

- **Delete**
  
  Use this button to delete the currently selected shape.

- **Undo**
  
  Use this button to undo the last action.

- **Redo**
  
  Use this button to redo the last action that was undone.

- **Show/Hide Numbers**
Use this button to toggle between showing or hiding the numbers for the shapes.

**Bring Shape to Front**
Use this button to bring the currently selected shape forward to the top layer.

**Bring Shape Forward**
Use this button to bring the currently selected shape forward one layer.

**Send Shape Backward**
Use this button to send the currently selected shape back one layer.

**Send Shape to Back**
Use this button to send the currently selected shape back to the bottom layer.

**Color Chooser**
Use this drop-down menu to select a color scheme for the lines and numbers of the shapes.

**Zoom Slider**
Use this slider to zoom the image in or out in the main image pane.

**Image Pane**
This main Image Pane is where you work with shapes to add hyperlinks to multiple areas within an image. The editing mechanisms that are supported in the Image Pane include the following:

**Mouse Controls and Keyboard Shortcuts**

- Use the mouse to select and move shapes around in the image pane. It is easy to see which shape is selected in this image pane because the border of the selected shape changes from a solid line to a dotted one.
- You can also drag any of the points of a selected shape to adjust its size and shape.
- You can hold down the **Ctrl** key to select multiple shapes and then move them simultaneously.
- You can also move shapes by using the arrow keys on your keyboard. In addition, you can hold down **Shift** while using the arrow keys to move the shape further or **Alt** to move it 1 pixel at a time.
- To zoom in or out, you can use the **NumPad +** or **NumPad -** keys respectively. Use **Ctrl + NumPad 0** to reset the zoom level to its default value.
- You can use **Ctrl + Z** to undo an action or **Ctrl + Y** to redo the last action that was undone.

**Contextual Menu Actions Available in the Image Pane**
You can right-click the shapes, or anywhere in the Image Pane to invoke the contextual menu where the following actions are available:

- **Duplicate**
  Create a duplicate of the currently selected shape.

- **Delete**
  Delete the currently selected shape.

- **New Rectangle**
  Creates a rectangular shape over an area in the image. You can drag any of the four points to adjust the size and shape of the rectangle.

- **Undo**
  Use this action to undo the last action.

- **Redo**
  Use this action to redo the last action that was undone.

**Shape Table**

The table at the right of the *Image Pane* is a sequential list of all the areas (shapes) that have been added in the image. It shows their number, type, and ID. If you select one of the entries in the table, the corresponding shape will be selected in the *Image Pane*.

**Properties**

- **Type**
  Displays information about the selected coordinate.

- **ID**
  The identifier for the selected area. This will become the value of the `@xml:id` attribute for the particular `<area>` element.

- **Target**
  Allows you to choose the target resource that you want the selected area to be linked to. You can enter the path to the target in the text field but the easiest way to select a target is to use the `Link` drop-down menu to the right of the text field. You can choose between the following types of links: *Cross Reference* or *Web Link*. Both types open a dialog box that allows you to select the target resource and it is inserted as the value of an `@xlink:href` attribute.

- **Description**
  You can enter an optional description for the selected area that will be displayed in the *Image Map Details* section (on page 735) in *Author* mode and as a tooltip message when the end-user hovers over the hyperlink in the output.
How to Create an Image Map in DocBook

To create an image map on an existing image in a DocBook document, follow these steps:

1. Right-click the image and select **Image Map Editor**.

   **Step Result:** This action will apply an *image map* to the current image and open the **Image Map Editor** dialog box.

2. Add hyperlinks to the image by selecting the □ **New Rectangle** button.
3. Move the shape to the desired area in the image and drag any of the points on the shape to adjust its size or form. You can use the other buttons on the toolbar (on page 732) to adjust its layer and color, or to perform other editing actions.

   **Tip:** You can right-click any of the shapes or anywhere in the Image Pane to access various helpful contextual menu actions (on page 733).

4. With the shape selected, enter an **ID** (on page 734) and use one of the linking options (on page 734) in the ☰ - **Link** drop-down menu to select a target resource (or enter its path in the **Target** (on page 734) text field).
5. (Optional) Enter a **Description** (on page 734) for the selected area (shape).
6. If you want to add more hyperlinks to the image, select □ **New Rectangle** button again and repeat the appropriate steps.
7. When you are finished creating hyperlinks, click **OK** to process your changes.

   **Result:** The *image map* is applied on the image and the appropriate elements and attributes are automatically added. In **Author** mode, the image map is now rendered over the image. If the image includes an `<alt>` element, its value will be displayed above the image. The following two buttons will also now be available at the top of the image in **Author** mode:

   - **Image Map Editor** - Click this button to open the **Image Map Editor**.
   - **Image Map Details** - Click this button to expand a section that displays the details of the image map.

How to Edit an Existing Image Map in DocBook

To edit an existing image map, use any of the following methods:

- Simply double-click the image.
- Right-click the image and select **Image Map Editor**.
- Click the **Image Map Editor** button below the image.

All three methods open the **Image Map Editor** where you can make changes to the image map using the various features described above. You can also make changes to the XML structure of the image map in the **Text** editing mode.
You can also click the **Image Map Details** button above the image to expand a section that displays the details of the image map and allows you to change the coordinates and IDs of the hyperlinked areas.

**Note:**
If you want to link a set of related `<area>` elements, you can use `<areaset>` elements. To add `<areaset>` elements, and `<area>` elements to the `areaset`, switch to **Text** mode and insert them manually.

**Overlapping Areas**

If shapes overlap one another in the **Image Map Editor**, the one on the top layer takes precedence. The number shown inside each shape represents its layer (if the numbers are not displayed, click the ![Show/Hide Numbers](on page 732)). To change the layer order for a shape, use the layer buttons on the **Image Map Editor** toolbar (on page 732) ![](, ![](, ![](, ![]().

If you insert a shape and all of its coordinates are completely inside another shape, the **Image Map Editor** will display a warning to let you know that the shape is entirely covered by a bigger shape. Keep in mind that if a shape is completely inside another shape, its hyperlink will only be accessible if its layer is on top of the bigger shape.

**Related Information:**
- [DocBook 'areaset' Element Specifications](on page 736)

**Image Maps in TEI**

Oxygen XML Editor includes support for **image maps** in TEI documents through the use of the `<facsimile>` element. In TEI documents, this feature provides an easy way to create areas (using `<zone>` elements) in an image where the end-user can hover or click to retrieve more information about that particular area of the image. The visual **Author** editing mode includes an **Image Map Editor** that helps you to easily create the areas in the image.
Image Map Editor Interface in TEI

The interface of the Image Map Editor consists of the following sections and actions:

**Toolbar**

- **New Rectangle**
  
  Use this button to draw a rectangular shape over an area in the image. You can drag any of the four points to adjust the size and shape of the rectangle.

- **New Polygon**
  
  Use this button to draw a polygon shape over an area in the image. This action opens a dialog box that allows you to select the number of points for the polygon. You can drag any of the points to adjust the size and shape of the polygon.

- **New Free Form Shape**
  
  Use this button to draw a free form shape over an area in the image. After selecting this button, left-click anywhere in the image to place the first point of your shape. Then move the cursor to the location of the next desired point and left-click to place the next point, and so on. To complete the shape (area), click the first point again and a line will automatically be added from the last point that was added, or
simply double-click the last point to automatically add the line from the last point back to the first.

Duplicate
Use this button to create a duplicate of the currently selected shape.

Delete
Use this button to delete the currently selected shape.

Undo
Use this button to undo the last action.

Redo
Use this button to redo the last action that was undone.

Show/Hide Numbers
Use this button to toggle between showing or hiding the numbers for the shapes.

Bring Shape to Front
Use this button to bring the currently selected shape forward to the top layer.

Bring Shape Forward
Use this button to bring the currently selected shape forward one layer.

Send Shape Backward
Use this button to send the currently selected shape back one layer.

Send Shape to Back
Use this button to send the currently selected shape back to the bottom layer.

Color Chooser
Use this drop-down menu to select a color scheme for the lines and numbers of the shapes.

Zoom Slider
Use this slider to zoom the image in or out in the main image pane.

Image Pane
This main Image Pane is where you work with shapes to add hyperlinks to multiple areas within an image. The editing mechanisms that are supported in the Image Pane include the following:

Mouse Controls and Keyboard Shortcuts
• Use the mouse to select and move shapes around in the image pane. It is easy to see which shape is selected in this image pane because the border of the selected shape changes from a solid line to a dotted one.
• You can also drag any of the points of a selected shape to adjust its size and shape.
• You can hold down the Ctrl key to select multiple shapes and then move them simultaneously.
• You can also move shapes by using the arrow keys on your keyboard. In addition, you can hold down Shift while using the arrow keys to move the shape further or Alt to move it 1 pixel at a time.
• To zoom in or out, you can use the NumPad + or NumPad - keys respectively. Use Ctrl + NumPad 0 to reset the zoom level to its default value.
• You can use Ctrl + Z to undo an action or Ctrl + Y to redo the last action that was undone.

### Contextual Menu Actions

You can right-click the shapes, points, or anywhere in the Image Pane to invoke the contextual menu where the following actions are available:

- **Add Point**
  Adds a point to Polygon or Free Form shapes.

- **Remove Point**
  Removes the current point from Polygon or Free Form shapes.

- **Duplicate**
  Create a duplicate of the currently selected shape.

- **Delete**
  Delete the currently selected shape.

- **New Rectangle**
  Creates a rectangular shape over an area in the image. You can drag any of the four points to adjust the size and shape of the rectangle.

- **New Polygon**
  Creates a polygon shape over an area in the image. This action opens a dialog box that allows you to select the number of points for the polygon. You can drag any of the points to adjust the size and shape of the polygon.

- **Undo**
Use this action to undo the last action.

Redo

Use this action to redo the last action that was undone.

Shape Table

The table at the right of the *Image Pane* is a sequential list of all the areas (shapes) that have been added in the image. It shows their number, type, and ID. If you select one of the entries in the table, the corresponding shape will be selected in the *Image Pane*.

Properties

**Type**

Displays information about the selected coordinate.

**ID**

The identifier for the selected area. This will become the value of the `@xml:id` attribute for the particular `<zone>` element. When you insert a new zone, a unique ID is automatically generated and displayed in this field. However, you can change this value if you want to.

How to Create an Image Map in TEI

To create an image map on an existing image in a TEI document, follow these steps:

1. The image `<graphic>` must be inside a `<facsimile>` element to support the *Image Map Editor* feature.

2. Right-click the image and select *Image Map Editor*.

   **Step Result:** This action will apply an *image map* to the current image and open the *Image Map Editor* dialog box.

3. Add areas (zones) in the image by selecting one of the shape buttons (□ *New Rectangle* or □ *New Polygon*).

4. Move the shape to the desired area in the image and drag any of the points on the shape to adjust its size or form. You can use the other buttons on the toolbar (on page 737) to adjust its layer and color, or to perform other editing actions.

   **Tip:**

   You can right-click any of the points, shapes, or anywhere in the Image Pane to access various helpful contextual menu actions (on page 739). For example, the easiest way to remove a point is to right-click the point and select ✗ *Remove Point*.

5. With the shape selected, enter an **ID** (on page 740).
6. If you want to add more areas (zones) to the image, select a shape button again and repeat the appropriate steps.

7. When you are finished, click OK to process your changes.

**Result:** The *image map* is applied on the image and the appropriate elements and attributes are automatically added. In *Author* mode, the image map is now rendered over the image and the following two buttons will now be available at the bottom of the image:

- **Image Map Editor** - Click this button to open the *Image Map Editor*.
- **Image Map Details** - Click this button to expand a section that displays the details of the image map.

**How to Edit an Existing Image Map in TEI**

To edit an existing image map, use any of the following methods:

- Simply double-click the image.
- Right-click the image and select **Image Map Editor**.
- Click the **Image Map Editor** button below the image.

All three methods open the *Image Map Editor* where you can make changes to the image map using the various features described above. You can also make changes to the XML structure of the image map in the *Text* editing mode.

You can also click the **Image Map Details** button below the image to expand a section that displays the details of the image map and allows you to change the coordinates and IDs of the hyperlinked areas.

**Restriction:**
Currently, if `<zone>` elements contain additional content (such as text or comments) and you edit the image map, the *Image Map Editor* does not preserve the additional content. Therefore, if you do need to insert additional content inside the `<zone>` elements, you should do so after the image map has been created and finalized. Subsequent changes to the image map should then be done in *Text* mode.

**Overlapping Areas**

If shapes overlap one another in the *Image Map Editor*, the one on the top layer takes precedence. The number shown inside each shape represents its layer (if the numbers are not displayed, click the **Show/Hide Numbers** button on the *Image Map Editor* toolbar *(on page 737)*). To change the layer order for a shape, use the layer buttons on the *Image Map Editor* toolbar *(on page 737)* (`เสมอ`, `ขึ้น`, `ลง`, `ลิข`).

If you insert a shape and all of its coordinates are completely inside another shape, the *Image Map Editor* will display a warning to let you know that the shape is entirely covered by a bigger shape. Keep in mind that if a shape is completely inside another shape, its hyperlink will only be accessible if its layer is on top of the bigger shape.
Warning:
PDF output is limited to rectangular shaped image map objects. Therefore, if your image contains circles or polygons, those objects will be redrawn as rectangles in the PDF output. Keep in mind that this might affect overlaps in the output.

Image Maps in XHTML

Oxygen XML Editor includes support for **image maps** in XHTML documents. This feature provides an easy way to create hyperlinks in various parts of an image without having to divide the image into separate image files. In HTML, an image (in the form of an `<img>` element) may be associated with an image map (in the form of a `<map>` element) by specifying a `@usemap` attribute on the `<img>` element. The visual **Author** editing mode includes an **Image Map Editor** that helps you to easily create and configure image maps.

**Figure 213. Image Map Editor in XHTML**

![Image Map Editor Interface in XHTML](image)

**Image Map Editor Interface in XHTML**

The interface of the **Image Map Editor** consists of the following sections and actions:

**Toolbar**

- **New Rectangle**

  Use this button to draw a rectangular shape over an area in the image. You can drag any of the four points to adjust the size and shape of the rectangle.
**New Circle**

Use this button to draw a circle over an area in the image. You can drag any of the four points to adjust the size of the circle.

**New Polygon**

Use this button to draw a polygon shape over an area in the image. This action opens a dialog box that allows you to select the number of points for the polygon. You can drag any of the points to adjust the size and shape of the polygon.

**New Free Form Shape**

Use this button to draw a free form shape over an area in the image. After selecting this button, left-click anywhere in the image to place the first point of your shape. Then move the cursor to the location of the next desired point and left-click to place the next point, and so on. To complete the shape (area), click the first point again and a line will automatically be added from the last point that was added, or simply double-click the last point to automatically add the line from the last point back to the first.

**Duplicate**

Use this button to create a duplicate of the currently selected shape.

**Delete**

Use this button to delete the currently selected shape.

**Undo**

Use this button to undo the last action.

**Redo**

Use this button to redo the last action that was undone.

**Show/Hide Numbers**

Use this button to toggle between showing or hiding the numbers for the shapes.

**Bring Shape to Front**

Use this button to bring the currently selected shape forward to the top layer.

**Bring Shape Forward**

Use this button to bring the currently selected shape forward one layer.

**Send Shape Backward**

Use this button to send the currently selected shape back one layer.

**Send Shape to Back**
Use this button to send the currently selected shape back to the bottom layer.

- **Color Chooser**

Use this drop-down menu to select a color scheme for the lines and numbers of the shapes.

- **Zoom Slider**

Use this slider to zoom the image in or out in the main image pane.

**Image Pane**

This main Image Pane is where you work with shapes to add hyperlinks to multiple areas within an image. The editing mechanisms that are supported in the Image Pane include the following:

**Mouse Controls and Keyboard Shortcuts**

- Use the mouse to select and move shapes around in the image pane. It is easy to see which shape is selected in this image pane because the border of the selected shape changes from a solid line to a dotted one.
- You can also drag any of the points of a selected shape to adjust its size and shape.
- You can hold down the **Ctrl** key to select multiple shapes and then move them simultaneously.
- You can also move shapes by using the arrow keys on your keyboard. In addition, you can hold down **Shift** while using the arrow keys to move the shape further or **Alt** to move it 1 pixel at a time.
- To zoom in or out, you can use the **NumPad +** or **NumPad -** keys respectively. Use **Ctrl + NumPad 0** to reset the zoom level to its default value.
- You can use **Ctrl + Z** to undo an action or **Ctrl + Y** to redo the last action that was undone.

**Contextual Menu Actions Available in the Image Pane**

You can right-click the shapes, points, or anywhere in the Image Pane to invoke the contextual menu where the following actions are available:

- **Add Point**
  
  Adds a point to *Polygon* or *Free Form* shapes.

- **Remove Point**

  Removes the current point from *Polygon* or *Free Form* shapes.

- **Duplicate**

  Create a duplicate of the currently selected shape.
Delete
Delete the currently selected shape.

New Rectangle
Creates a rectangular shape over an area in the image. You can drag any of the four points to adjust the size and shape of the rectangle.

New Circle
Creates a circle over an area in the image. You can drag any of the four points to adjust the size of the circle.

New Polygon
Creates a polygon shape over an area in the image. This action opens a dialog box that allows you to select the number of points for the polygon. You can drag any of the points to adjust the size and shape of the polygon.

Undo
Use this action to undo the last action.

Redo
Use this action to redo the last action that was undone.

Shape Table

The table at the right of the Image Pane is a sequential list of all the areas (shapes) that have been added in the image. It shows their number, type, and description (value of the Alternative property). If you select one of the entries in the table, the corresponding shape will be selected in the Image Pane.

Properties

Type
Displays information about the selected coordinate.

Href
Specifies the hyperlink target for the selected area. This will become the value of the @href attribute for the particular <area> element. The possible values are:

- An Absolute URL - A URL of another website (for example, http://www.example.com/index.htm).
- A Relative URL - A link to a file within your website (for example, index.htm).
- An Element - A link to the ID of an element within the page (for example, #top).
Other Protocols - A specified path using other protocols (such as https://, ftp://, mailto:, file:).

A Script - A link to a script (for example, javascript:alert('Hello');)

Alternate
The description for the selected area. The value is inserted in an alt attribute in the particular <area> element. This is a required attribute to present a text alternative for browsers that do not display images.

Target
Specifies where to open the linked resource. The allowed values are:

- _blank - Opens the linked resource in a new window or tab.
- _self - Opens the linked resource in the same frame as it was clicked.
- _parent - Opens the linked resource in the full body of the window.
- framename - Opens the linked resource in the named frame.

How to Create an Image Map in XHTML
To create an image map on an existing image in an XHTML document, follow these steps:

1. Right-click the image and select Image Map Editor.

   **Step Result:** This action will apply an image map to the current image and open the Image Map Editor dialog box.

2. Add hyperlinks to the image by selecting one of the shape buttons (New Rectangle, New Circle, or New Polygon).
3. Move the shape to the desired area in the image and drag any of the points on the shape to adjust its size or form. You can use the other buttons on the toolbar (on page 742) to adjust its layer and color, or to perform other editing actions.

   **Tip:**
   You can right-click any of the points, shapes, or anywhere in the Image Pane to access various helpful contextual menu actions (on page 744). For example, the easiest way to remove a point is to right-click the point and select Remove Point.

4. With the shape selected, specify the hyperlink target in the Href field (on page 745) and enter a description for the selected area in the Alternate field (on page 746).
5. (Optional) Specify where the hyperlink resource will be opened in the Target field (on page 746).
6. If you want to add more hyperlinks to the image, select a shape button again and repeat the appropriate steps.
7. When you are finished creating hyperlinks, click OK to process your changes.
Result: The *image map* is applied on the image and the appropriate elements and attributes are automatically added. In *Author* mode, the image map is now rendered over the image and its properties are displayed in a section below the image.

**How to Edit an Existing Image Map in XHTML**

To edit an existing image map, use any of the following methods:

- Simply double-click the image.
- Right-click the image and select *Image Map Editor*.
- Click the *Image Map Editor* button below the image.

All three methods open the *Image Map Editor* where you can make changes to the image map using the various features described above. You can also make changes to the XML structure of the image map in the Text editing mode.

In *Author* mode, the details of the image map are also displayed below the image and you can edit the description, *href*, *shape*, and coordinates of the hyperlinked areas. Keep in mind that if you change the *shape* in this section, you also need to add or remove coordinates to match the requirements of the new shape.

**Overlapping Areas**

If shapes overlap one another in the *Image Map Editor*, the one on the top layer takes precedence. The number shown inside each shape represents its layer (if the numbers are not displayed, click the *Show/Hide Numbers* button on the *Image Map Editor* toolbar *(on page 742)*). To change the layer order for a shape, use the layer buttons on the *Image Map Editor* toolbar *(on page 742)* (†, ††, †††, ††††).

If you insert a shape and all of its coordinates are completely inside another shape, the *Image Map Editor* will display a warning to let you know that the shape is entirely covered by a bigger shape. Keep in mind that if a shape is completely inside another shape, its hyperlink will only be accessible if its layer is on top of the bigger shape.

![Warning: PDF output is limited to rectangular shaped image map objects. Therefore, if your image contains circles or polygons, those objects will be redrawn as rectangles in the PDF output. Keep in mind that this might affect overlaps in the output.]

**Related Information:**

- HTML Image Map Specifications

**Adding Video, Audio, and Embedded HTML Resources**

You can insert references to media resources (such as videos, audio clips, or embedded HTML frames) in your DITA, DocBook, or XHTML topics. The media resources can be played directly in *Author* mode and in all
HTML5-based outputs. There is a toolbar button that allows you to insert and configure a reference to the media resource. You can also drag media files from your system explorer or the Project view (on page 402) and drop them into your documents (or copy and paste them).

### Table 6. Supported Media Types

<table>
<thead>
<tr>
<th>Media</th>
<th>Description</th>
<th>Type</th>
<th>Supported Size Properties</th>
</tr>
</thead>
<tbody>
<tr>
<td>mp3</td>
<td>Moving Picture Experts Group Layer-3 Audio</td>
<td>audio</td>
<td>Width</td>
</tr>
<tr>
<td>wav</td>
<td>Windows Wave</td>
<td>audio</td>
<td>Width</td>
</tr>
<tr>
<td>pcm</td>
<td>Pulse Code Modulation</td>
<td>audio</td>
<td>Width</td>
</tr>
<tr>
<td>m4a</td>
<td>Moving Picture Experts Group Layer-4 Audio</td>
<td>audio</td>
<td>Width</td>
</tr>
<tr>
<td>aif</td>
<td>Audio Interchange Format</td>
<td>audio</td>
<td>Width</td>
</tr>
<tr>
<td>mp4</td>
<td>Moving Picture Experts Group Layer-4 Video</td>
<td>video</td>
<td>Width &amp; Height</td>
</tr>
<tr>
<td>m4v</td>
<td>iTunes Video File</td>
<td>video</td>
<td>Width &amp; Height</td>
</tr>
<tr>
<td>avi</td>
<td>Audio Video Interleaved</td>
<td>video</td>
<td>Width &amp; Height</td>
</tr>
<tr>
<td>embedded video (such as YouTube or Vimeo)</td>
<td>Embedded Iframe Code</td>
<td>iframe</td>
<td>Width &amp; Height</td>
</tr>
</tbody>
</table>

### Adding a Media Resource

To insert a media resource in a document, use the following procedure:

1. Place the cursor at the location where you want the media resource.
2. Select the Insert Media Resource action from the toolbar. A Chose Media dialog box appears.

   ![](Note.png)

   **Note:**

   You can also drag media files from your system explorer or the Project view (on page 402) and drop them into your documents (or copy and paste them).

3. Select the URL for the media resource and click Ok.

   **Result in Author Mode:** A reference to the specified media object is inserted and rendered in Author mode so that it can be played directly from there.
Attention:

- On Ubuntu 17.10, if you receive an error when trying to play videos in Author mode, you need to install the libavformat57 library.

Result in Output: In the publishing stage, the media object is converted to an HTML5 element so that it can be rendered properly and played in all HTML5-based outputs.

Embedding HTML Content in DITA Topics

The DITA Open Toolkit that comes bundled with Oxygen XML Editor includes a pre-installed plugin (on page 3222) that allows you to embed well-formed HTML content directly in a DITA topic.

For example, suppose you wanted to embed a YouTube video directly in a DITA topic.

The DITA topic would look like this:

```xml
....
<foreign outputclass="html-embed">
<![CDATA[
    <iframe width="420" height="315"
    src="https://www.youtube.com/embed/qepRkQxhTX"
    frameborder="0" allowfullscreen="true">
    </iframe>
]]>
</foreign>
....
```

The converted HTML output would look like this:

```html
...........
<iframe width="420" height="315" src="https://www.youtube.com/embed/qepRkQxhTX"
frameborder="0" allowfullscreen="true">
</iframe>
...........
```

The plugin is also available on the oxygenxml GitHub projects page.

Related Information:
How to Add Video and Audio Objects in DITA WebHelp Output (on page 1674)

Editing MathML Notations

Oxygen XML Editor includes a built-in editor for MathML notations. To start the MathML editor, double-click a MathML notation (for embedded notations, you can also select the Edit Equation action from its contextual menu). In the MathML editor, you can edit the mathematical symbols of a MathML notation. To open a MathML
file from your current project, select **Open with > MathML editor** from the contextual menu in the **Project view** (on page 402).

![Default MathML Editor](image)

The font size and font family that is used for the equations is based upon the context where the MathML equation appears. To configure the minimum font size of the equation, open the **Preferences** dialog box (**Options > Preferences**) (on page 124) and go to **Editor > Edit modes > Author > MathML**.

**Tip:**
For editing **MathML** equations, you can also try free choices, such as **Apache OpenOffice Math** (on page 751) or **LibreOffice Math** (on page 751).
Configuring the MathFlow Editor

The MathFlow Components product can replace the default MathML editor with a specialized MathML editor. You have to purchase a MathFlow Component from Wiris and configure it in Oxygen XML Editor with the following procedure:

1. Install MathFlow by using the Universal installer (for versions prior to 2.1, use the MathFlow SDK).
2. Set the path to the MathFlow install folder in the MathML preferences page (on page 191).
3. Set the path to the MathFlow license file in the MathML preferences page (on page 191).

Figure 215. Default MathFlow Editor

Configuring an External MathML Editor

You can configure Oxygen XML Editor to use a third-party MathML editor (e.g. the free Libre Office equation editor) by following these steps:

1. Install the third-party application (for example, Libre Office).
2. Open the MathML preferences page (on page 191) and in the External application > Command line field, set the command line used to open the external application. For example, on Windows, for starting the Libre Office equation editor, the command line would need to look like this: "C:\Program Files\LibreOffice 5\program\smath.exe" "$cf". You can use the $cf editor variable in the command line to refer to a temporary file automatically created by Oxygen XML Editor that will contain the edited MathML content.
3. Insert a new equation or double-click an existing equation. The external application starts and it should display the equation inside it. Once you save the equation and close the external application, the equation rendered in the Author visual editing mode will refresh its contents based on your changes. When editing and saving the equation in the started external application, do not alter the path to the saved file in any way as the file is specifically saved in a location from where the Oxygen XML Editor application will load it automatically.
Special Character Support in Author Mode

Oxygen XML Editor offers support for bidirectional text, such as Arabic or Hebrew languages that require right-to-left scripts, certain Asian languages (such as Devanagari, Bengali, Gurmukhi, Gujarati, Oriya, Tamil, Telugu, Kannada, Malayalam, Sinhala, Thai, Khmer), or other special characters (such as combining characters). To achieve this, Oxygen XML Editor uses the Unicode Bidirectional Algorithm, as specified by the Unicode Consortium. The text arrangement is similar to what you get in a modern HTML browser. The final text layout is rendered according to the directional CSS properties matching the XML elements and the Unicode directional formatting codes.

By default, when navigating bidirectional text with the arrow keys in Author mode, pressing the right arrow key moves the cursor in the writing direction and the left arrow moves it in the opposite direction. However, if the Arrow keys move the cursor in the writing direction option (on page 180) in the Cursor Navigation preferences page is not selected, pressing the right arrow will simply move the cursor to the right (and the left arrow moves it to the left), regardless of the text direction.

Tip:
If you experience performance issues when editing documents that contain bidirectional text, you could try one of the following solutions:

- The Eclipse plugin distribution of Oxygen XML Editor is faster than the standalone version when working with bidirectional text.
- You could try changing the font. For example, you could try using the David font in Hebrew content. If it is not already installed in your operating system, this font is available at: https://www.microsoft.com/typography/fonts/family.aspx?FID=234. To change the font in Oxygen XML Editor, open the Preferences dialog box (Options > Preferences) (on page 124), go to Appearance > Fonts, and change the font using the Author default font option in the Fonts preferences page (on page 133).

For more information about the bidirectional text support in the Author mode, watch our video demonstration:

https://www.youtube.com/embed/IC0ahH1IS7s

Related Information:
- Special Character Support in Text Mode (on page 564)
- Special Character Support in Grid Mode (on page 586)
- Inserting Special Characters with the Character Map (on page 465)

Controlling the Text Direction Using XML Markup

Oxygen XML Editor supports the following CSS properties that control the direction of text:
Table 7. CSS Properties Controlling Text Direction

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>direction</td>
<td>Specifies the writing direction of the text. The possible values are ltr (the text direction is left to right), rtl (the text direction is right to left), and inherit (the direction property is inherited from the parent element).</td>
</tr>
<tr>
<td>unicodeBidi</td>
<td>Used along with the direction property to create levels of embedded text with different text directions in the same document. The possible values of this property are bidi-override (creates an additional level of embedding and forces all strong characters to the direction specified in the direction), embed (creates an additional level of embedding), normal (does not use an additional level of embedding), and inherit (the value of the unicodeBidi property is inherited from parent element).</td>
</tr>
</tbody>
</table>

For instance, to declare an element as being Right to Left, you could use a stylesheet like this:

**XML File:**

```xml
<article>
  <myRTLpara>RIGHT TO LEFT TEXT</myRTLPara>
</article>
```

**Associated CSS File:**

```css
myRTLpara{
  direction:rtl;
  unicode-bidi:embed;
}
```

Oxygen XML Editor recognizes the `dir` attribute on any XML document. The supported values are:

<table>
<thead>
<tr>
<th>Values</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ltr</td>
<td>The text from the current element is Left to Right, embedded.</td>
</tr>
<tr>
<td>rtl</td>
<td>The text from the current element is Right to Left, embedded.</td>
</tr>
<tr>
<td>lro</td>
<td>The text from the current element is Left to Right, embedded.</td>
</tr>
<tr>
<td>rlo</td>
<td>The text from the current element is Right to Left, embedded.</td>
</tr>
</tbody>
</table>

The following XML document types make use of the `dir` attribute with the above values:

- DITA
- DocBook
- TEI
- XHTML
Note:
When the *inline element* tags are visible, the text in the line is arranged according to the BIDI algorithm after replacing the tags symbols with Object Replacement Characters. This makes it possible to get a different text arrangement when viewing a document in the *No Tags* mode versus viewing it in the *Full Tags* mode.

Controlling the Text Direction Using the Unicode Direction Formatting Codes

These Unicode Direction Formatting Codes can be embedded in the edited text, specifying a text direction and embedding. However, it is not recommended to use them in XML as they are zero width characters, making it hard to debug the text arrangement.

**Table 8. Directional Formatting Codes**

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>U+202A</td>
<td>LEFT-TO-RIGHT EMBEDDING</td>
<td>Treats the following text as embedded left-to-right.</td>
</tr>
<tr>
<td>U+202B</td>
<td>RIGHT-TO-LEFT EMBEDDING</td>
<td>Treats the following text as embedded right to left.</td>
</tr>
<tr>
<td>U+202D</td>
<td>LEFT-TO-RIGHT OVERRIDE</td>
<td>Forces the following characters to be treated as strong left-to-right characters.</td>
</tr>
<tr>
<td>U+202E</td>
<td>RIGHT-TO-LEFT OVERRIDE</td>
<td>Forces the following characters to be treated as strong right-to-left characters.</td>
</tr>
<tr>
<td>U+202C</td>
<td>POP DIRECTIONAL FORMATTING CODE</td>
<td>Restores the bidirectional state to what it was before the last LRE, RLE, RLO, or LRO.</td>
</tr>
<tr>
<td>U+200E</td>
<td>LEFT-TO-RIGHT MARK</td>
<td>Left-to-right strong zero-width character.</td>
</tr>
<tr>
<td>U+200F</td>
<td>RIGHT-TO-LEFT MARK</td>
<td>Right-to-left strong zero-width character.</td>
</tr>
</tbody>
</table>

To insert Unicode Direction Formatting Codes, use the *Character Map* dialog box. To easily find such a code, you can either enter directly the hexadecimal value, or use the *Details* tab to enter the codes name.

Oxygen XML Editor offers the support for bi-directional text in all the side views (*Outline view*, *Attributes view* and so on) and text fields.

Refreshing the Content

On occasion you may need to reload the content of the document from the disk or reapply the CSS. This can be performed by using the *Reload* action available on the toolbar.
To refresh the content of the referenced resources you can use the Refresh references action that is available in the menu for the current framework (for example, the DITA or DocBook menu). However, this action will not refresh the expanded external entities. For that, you will need to use the Reload action.

Generating IDs for Elements in Author Mode

Oxygen XML Editor allows you to manually assign or edit values of id attributes in Author mode by using the Attributes View (on page 627) or an in-place attribute editor (on page 610). Oxygen XML Editor also includes mechanisms to generate ID values for elements, either on-request or automatically, in DITA, DocBook, or TEI documents.

Generate IDs On-Request

You can generate ID values for specific elements on-request. To do so, select the element that will have an ID generated (or place the cursor inside the element) and select the Generate IDs action from the contextual menu. This action generates a unique ID for the current element. If you invoke the action on a block of selected content, the action will generate IDs for all top-level elements and elements that are listed in the ID Options dialog box (on page 755). To open this dialog box, open an XML document that belongs to that specific document type and then select ID Options from the DITA, DocBook, or TEI main menu (depending on your document type).

Note:
The Generate IDs action does not overwrite existing ID values. It only affects elements that do not already have an id attribute.

Automatically Generate IDs

Oxygen XML Editor includes an option to automatically add unique ID values to certain elements when they are created in Author mode. The Auto generate IDs for elements option can be found in the ID Options dialog box (on page 755) that is displayed when you select the ID Options action from the framework (on page 3219)-specific menu (DITA, DocBook, or TEI). If this Auto generate IDs for elements option is selected, Oxygen XML Editor automatically generates unique ID values for elements that are listed in this dialog box. You can use this dialog box to customize the format of the ID values and choose which elements will have their ID values automatically generated (for example, you can customize the list of elements to include those that you most often need to identify).

ID Options Dialog Box

To configure options for generating IDs, select ID Options from the DITA, DocBook, or TEI menu (depending on your document type).
The **ID Options** dialog box allows you to configure the following options with regard to generating ID values:

**ID Pattern**

The pattern for the ID values that will be generated. This text field can be customized using constant strings or any of the Oxygen XML Editor Editor Variables (on page 323).

**Element name or class value to generate ID for**

The elements that will have ID values generated, specified using class attribute values. To customize the list, use the **Add**, **Edit**, or **Remove** buttons.

**Auto generate IDs for elements**

If selected, Oxygen XML Editor will automatically generate unique IDs for the elements listed in this dialog box when they are created in **Author** mode.

**Remove IDs when copying content in the same document (DITA or TEI)**

When copying and pasting content in the same DITA or TEI document, this option allows you to control whether or not pasted elements that are listed in this dialog box should retain their existing IDs. To retain the element IDs, deselect this option.

**Note:**

This option does not have an effect on content that is cut and pasted.

**Remove IDs when copying content (DocBook)**
This option allows you to control whether or not pasted elements that are listed in this dialog box should retain their existing IDs in DocBook documents. If this option is not selected, IDs are always retained when you copy or cut content and paste it in the same document or other documents. If this option is selected, IDs are never retained for copied content, but if you cut the content, they are preserved for the first paste action (and not retained for any subsequent paste actions).

**Duplicating Elements with Existing IDs**

If you duplicate elements with existing IDs (for example, through copy/paste or drag/drop actions), all IDs are removed at the resolution of the operation. However, you can use the options in the ID Options dialog box to change this behavior. The options in this dialog box affect duplicated elements with existing IDs in the following ways:

- Only the elements listed in this dialog box are affected by these options. Therefore, if you want to use these options to preserve IDs or generate new ones, you must first add the elements to be duplicated to the list in this dialog box.
- If the Auto generate IDs for elements option (on page 756) is selected and you duplicate elements with existing IDs, Oxygen XML Editor assigns new, unique ID values to the duplicates.
- If the Auto generate IDs for elements option (on page 756) is not selected and you duplicate elements with existing IDs, the ID values are removed from the duplicates.
- For DITA and TEI, if the Remove IDs when copying content in the same document option (on page 756) is selected, the ID values are removed from elements that are duplicated in the same document. If it is not selected, the ID values are preserved when elements are duplicated in the same document. Selecting this option has no effect if the Auto generate IDs for elements option is selected and this option has no effect on elements that are duplicated in other documents.
- For DocBook, if the Remove IDs when copying content option (on page 756) is selected, the ID values are removed from any element that is duplicated. If it is not selected, the ID values are preserved when elements are duplicated. Selecting this option has no effect if the Auto generate IDs for elements option is selected.

**Controlling the Default ID Generation Options**

It is possible to configure the default ID generation options for DITA, DocBook, and TEI document types. In the frameworks folder for each of those document types, there is an XML configuration file called idGenerationDefaultOptions.xml that contains the default settings for generating IDs in each particular type of document. To configure the default settings, you can edit this file and save it back to the same directory.

The configuration file can be found in the resources folder within the particular framework (on page 3219). For example, the configuration file for the DITA framework is located in: 

{OXYGEN_INSTALL_DIR}/frameworks/dita/resources/idGenerationDefaultOptions.xml.

**Sharing Default ID Generation Options**
If you want to share your configured default ID generation settings with other members of your team, follow these steps:

1. Configure the `idGenerationDefaultOptions.xml` file for your framework according to your needs.
2. Bundle a modified version of the entire framework folder (for example, 
   `[OXYGEN_INSTALL_DIR]/frameworks/dita/`). To do this:
   a. Open the Preferences dialog box (Options > Preferences) (on page 124) and go to Document Type Association.
   b. Select your document type and click the Extend button.
   c. In the Document type configuration dialog box (on page 140) that is now displayed, select External for the Storage option. By default, this will save the extension in a new folder in the frameworks folder (for example, 
      `[OXYGEN_INSTALL_DIR]/frameworks/dita-extension (1)`), but you can also use the Browse button to specify a specific name and folder.
   d. In this new extension folder, create a new folder called resources and add your modified idGenerationDefaultOptions.xml file to this new resources folder.
   e. Go back to the Document Type Association preferences page, select the extended framework, and click Edit.
   f. Go to the Classpath tab (on page 144), add a reference to your new resources folder, and move this reference up (using the Move Up button) so that it is the first one that appears in the list.
   g. Click OK and exit out of the preferences page.
3. Distribute your newly extended folder to other team members by using one of the methods described in Sharing a Framework (on page 2288).

**Using Form Controls in Author Mode**

Form controls make it easier to capture, organize, and edit content. They are especially helpful for less technical users because form controls provide a way to interact with the content of a document in a graphical manner without intimidating the user with the XML structure.

Oxygen XML Editor includes a variety of built-in form controls (on page 2373) that can be defined in CSS stylesheets that are used to render Author mode (on page 2373). You can also implement custom form controls (on page 2404) for more specific needs. The types of built-in form controls that are available include:

- **Audio (on page 2374)** - A media object that plays audio clips.
- **Browser (on page 2375)** - A media object that renders HTML frames or interact with SVG documents
- **Button (on page 2379)** - A graphical user interface object that performs a specific action.
- **Button Group (on page 2381)** - A graphical user interface group of buttons (such as radio buttons) that perform specific actions.
- **Checkbox (on page 2385)** - A graphical user interface box that you can click to select or deselect a value.
- **Combo Box (on page 2387)** - A graphical user interface object that can be a drop-down menu or a combination of a drop-down menu and a single-line text field.
• **Date Picker (on page 2390)** - A form control object that allows you to select a date in a specified format.
• **HTML Content (on page 2392)** - A graphical user interface box that is used for rendering HTML content.
• **Pop-up (on page 2393)** - A contextual menu that provides quick access to various actions.
• **Text Area (on page 2396)** - A box that allows you to enter multiple lines of text.
• **Text Field (on page 2399)** - A graphical user interface box that allows you to enter a single line of text.
• **URL Chooser (on page 2402)** - A dialog box that allows you to select the location of local or remote resources.
• **Video (on page 2403)** - A media object that plays videos.

The following image is an example of several form controls rendered in **Author** mode. The first (**Direct manager**) is a combo box with both a drop-down menu and an editable text field. This is followed by a simple text field (**Homepage**), and the [+] and [-] icons also represent button form controls that are assigned specific actions to add or delete records from the document.

![Figure 217. Example of Form Controls in Author Mode](image)

You can use your imagination to envision the multitude of ways that you can use form controls to make the editing experience for content authors easier and more efficient. As a working example, a bundled **samples** project (located in the **samples** folder inside the Oxygen XML Editor installation directory) contains a file called **personal.xml** that contains form controls. You can use this file, along with its corresponding **personal.css** file to experiment with an example of how form controls can be implemented in **Author** mode.

Related Information:
Form Controls (on page 2373)

**Contextual Menu Actions in Author Mode**

Oxygen XML Editor includes powerful support for editing XML documents through actions included in the contextual menu. When editing XML documents in **Author** mode, the contextual menu includes **general**
actions that are available for all of the recognized document types and framework (on page 3219)-specific actions that are configured for each document type.

General Contextual Menu Actions in Author Mode

The general actions that are available in the contextual menu (some of them are also available in the submenus of the Document menu) for all built-in document types include the following:

Add File to Review Task

This action can be used to add the current document to a task in the Content Fusion Tasks Manager view. Oxygen Content Fusion is a flexible, intuitive collaboration platform designed to adapt to any type of documentation review workflow. This functionality is available through a pre-installed connector add-on (on page 2527). To fully take advantage of all of the benefits and features of Content Fusion, your organization will need an Oxygen Content Fusion Enterprise Server. For more information, see the Oxygen Content Fusion website.

Quick Fix (Alt + 1 (Command + Alt + 1 on macOS))

Available when the contextual menu is invoked on an error where Oxygen XML Editor can provide a Quick Fix (on page 813).

Open Image

Available when the contextual menu is invoked on an image. This action allows you to open an image in the Oxygen XML Editor Image Viewer (on page 469) or in a default system application associated with the current image type.

Show in Explorer (Show in Finder on macOS)

Available when the contextual menu is invoked on an image. This action allows you to open the parent directory of an image in the system file explorer, and it selects the image file.

Save image as

Available when the contextual menu is invoked on a local, external, or embedded image. This action opens a dialog box where you can choose a location in the local file system where the image will be saved. It also includes an option (Update the reference to the image) that can be toggled to choose whether or not to update the path in existing references to the image.

Track Changes Actions

Available when the Track Changes feature (on page 3224) is enabled and the contextual menu is invoked on a change. The following options are available:

Accept Change(s)

Accepts the Tracked Change (on page 3224) located at the cursor position or all of the changes in a selection. If you select a part of a deletion or insertion change, only the selected content is accepted. If you select multiple changes, all of them are accepted. For an insertion change, it keeps the inserted text and for a deletion change, it removes the content from the document.
Reject Change(s)

Rejects the Tracked Change (on page 3224) located at the cursor position or all of the changes in a selection. If you select a part of a deletion or insertion change, only the selected content is rejected. If you select multiple changes, all of them are rejected. For an insertion change, it removes the inserted text and for a deletion change, it preserves the original content.

Comment Change

Opens a dialog box that allows you to add a comment to an existing Tracked Change (on page 3224). The comment will appear in a callout and a tooltip when hovering over the change. If the action is selected on an existing commented change, the dialog box will allow you to edit the comment.

Author Callout Actions

Available when the contextual menu is invoked on a callout (on page 3217). If the corresponding options in the Show review callouts section (on page 186) are selected in the Callouts preferences page (on page 186), the callouts are displayed in Author mode for comments, tracked insertions, or tracked deletions.

Insertion or Deletion Callout Actions

The following actions are available in the contextual menu when invoked on an insertion or deletion callout box:

Reply

Opens a dialog box that allows you to add a reply to a comment or Tracked Changes (on page 3224). When replying to a comment, the dialog box shows the entire conversation in the comment thread, starting with the first comment added in the particular thread, followed by all the replies. After replies are added to a comment thread, they are displayed with an indentation in the callouts and Review view (on page 664).

Mark as Done

A toggle action that marks or unmarks a comment or comment thread as being done. It is also available for Tracked Changes (on page 3224) that are displayed in a callout. When a comment or change is marked as done, the callout is grayed out and cannot be edited unless the action is toggled to the unmarked state. The action applies to the particular comment and all of its descendents. This is useful for marking comments or changes that have been addressed, leaving only those that still need some attention.

Accept Change(s)
Accepts the *Tracked Change* located at the cursor position or all of the changes in a selection. If you select a part of a *deletion* or *insertion* change, only the selected content is accepted. If you select multiple changes, all of them are accepted. For an *insertion* change, it keeps the inserted text and for a *deletion* change, it removes the content from the document.

**Reject Change(s)**

Rejects the *Tracked Change* located at the cursor position or all of the changes in a selection. If you select a part of a *deletion* or *insertion* change, only the selected content is rejected. If you select multiple changes, all of them are rejected. For an *insertion* change, it removes the inserted text and for a *deletion* change, it preserves the original content.

**Comment Change**

Opens a dialog box that allows you to add a comment to an existing *Tracked Change*. The comment will appear in a callout and a tooltip when hovering over the change. If the action is selected on an existing commented change, the dialog box will allow you to edit the comment.

**Edit Reference**

If the fragment that contains a callout is a reference, use this option to go to the reference and edit the callout.

**Callouts Options**

Select this option to open the *Callouts preference page* where you can configure various callout options.

**Comment Callout Actions**

The following actions are available in the contextual menu when invoked on a comment callout box:

**Reply**

Opens a dialog box that allows you to add a reply to a comment or *Tracked Changes*. When replying to a comment, the dialog box shows the entire conversation in the comment thread, starting with the first comment added in the particular thread, followed by all the replies. After replies are added to a comment thread, they are displayed with an indentation in the callouts and *Review view*.

**Mark as Done**
A toggle action that marks or unmarks a comment or comment thread as being done. It is also available for Tracked Changes (on page 3224) that are displayed in a callout. When a comment or change is marked as done, the callout is grayed out and cannot be edited unless the action is toggled to the unmarked state. The action applies to the particular comment and all of its descendents. This is useful for marking comments or changes that have been addressed, leaving only those that still need some attention.

Show/Edit Comment

Opens a dialog box that displays the discussion thread and allows the current user to edit comments that do not have replies. If you are not the author who inserted the original comment, the dialog box just displays the comment without the possibility of editing it.

Remove Comment

Removes a selected comment. If you remove a comment that contains replies, all of the replies will also be removed.

Callouts Options

Select this option to open the Callouts preference page (on page 186) where you can configure various callout options.

Edit Attributes

Displays an in-place attributes editor (on page 630) that allows you to manage the attributes of an element.

Edit Profiling Attributes

Allows you to change the profiling attributes (on page 668) defined on all selected elements.

Insert submenu

This submenu includes insert actions that are specific to each framework (on page 3219), along with the following general action:

Insert Entity

Allows you to insert a predefined entity or character entity. Surrogate character entities (range #x10000 to #x10FFFF) are also accepted. Character entities can be entered in one of the following forms:

- `#<decimal value>` - e.g. #65
- `&#<decimal value>;` - e.g. &#65
- `#x<hexadecimal value>` - e.g. #x41
- `&#x<hexadecimal value>;` - e.g. &#x41

Cut (Ctrl + X (Command + X on macOS))
Removes the currently selected content from the document and places it in the clipboard.

Copy (Ctrl + C (Command + C on macOS))

Places a copy of the currently selected content in the clipboard.

Paste (Ctrl + V (Command + V on macOS))

Inserts the current clipboard content into the document at the cursor position.

Paste special submenu

This submenu includes special paste actions that are specific to each framework (on page 3219), as well as the following general paste actions:

Paste As XML

Pastes clipboard content that is considered to be XML, preserving its XML structure.

Paste As Text

Pastes clipboard content, ignoring any structure or styling markup.

Select submenu

This submenu allows you to select the following:

Element

Selects the entire element at the current cursor position.

Content

Selects the entire content of the element at the current cursor position, excluding the start and end tag. Performing this action repeatedly will result in the selection of the content of the ancestor of the currently selected element content.

Parent

Selects the entire parent element at the current cursor position.

Text submenu

This submenu contains the following actions:

To Lower Case

Converts the selected content to lower case characters.

To Upper Case

Converts the selected content to upper case characters.

Capitalize Sentences

Converts to upper case the first character of every selected sentence.

Capitalize Words
Converts to upper case the first character of every selected word. 0034

Count Words

Counts the number of words and characters (no spaces) in the entire document or in the selection for regular content and read-only content.

**Note:**
The content marked as deleted with [change tracking (on page 3224)](#) is ignored when counting words.

Convert Hexadecimal Sequence to Character (Ctrl + Shift + X (Command + Shift + X on macOS))

Converts a sequence of hexadecimal characters to the corresponding Unicode character (on page 463). The action can be invoked if there is a selection containing a valid hexadecimal sequence or if the cursor is placed at the right side of a valid hexadecimal sequence. A valid hexadecimal sequence can be composed of 2 to 4 hexadecimal characters and may or may not be preceded by the 0x or 0X prefix. Examples of valid sequences and the characters they will be converted to:

- 0x0045 will be converted to E
- 0X0125 to ĥ
- 265 to ū
- 2190 to ←

**Note:**
For more information about finding the hexadecimal value of a character, see [Finding the Decimal, Hexadecimal, or Character Entity Equivalent (on page 466)](#).

Refactoring submenu

Contains a series of actions designed to alter the XML structure of the document:

- **Toggle Comment**
  
  Encloses the currently selected text in an XML comment, or removes the comment if it is commented.

- **Move Up (Alt + UpArrow)**
  
  Moves the current node or selected nodes in front of the previous node.

- **Move Down (Alt + DownArrow)**
  
  Moves the current node or selected nodes after the subsequent node.

- **Split Element (Alt + Shift + D (Ctrl + Alt + D on macOS))**
Splits the content of the closest element that contains the position of the cursor. Thus, if the cursor is positioned at the beginning or at the end of the element, the newly created sibling will be empty.

Join Elements

Joins two adjacent block elements (on page 3216) that have the same name. The action is available only when the cursor position is between the two adjacent block elements. Also, joining two block elements can be done by pressing the Delete or Backspace keys and the cursor is positioned between the boundaries of these two elements.

Surround with Tags (Ctrl + E (Command + E on macOS))

Allows you to choose a tag to enclose a selected portion of content. If there is no selection, the start and end tags are inserted at the cursor position.

- If the Position cursor between tags option (on page 212) is selected in the Content Completion preferences page, the cursor is placed between the start and end tag.
- If the Position cursor between tags option (on page 212) is not selected in the Content Completion preferences page, the cursor is placed at the end of the start tag, in an insert-attribute position.

Surround with ‘[tag]’ (Ctrl + ForwardSlash (Command + ForwardSlash on macOS))

Surround the selected content with the last tag used.

Rename Element

The element from the cursor position, and any elements with the same name, can be renamed according with the options from the Rename dialog box.

Delete Element Tags

Deletes the tags of the closest element that contains the position of the cursor. This operation is also executed if the start or end tags of an element are deleted by pressing the Delete or Backspace keys.

Remove All Markup

Removes all the XML markup inside the selected block of content and keeps only the text content.

Remove Text

Removes the text content of the selected block of content and keeps the markup intact with empty elements.

DITA-related Refactoring Actions
A variety of built-in XML refactoring operations that pertain to DITA documents with some of the information preconfigured based upon the current context.

**Change Topic ID to File Name**

Use this operation to change the ID of a topic to be the same as its file name.

**Convert CALS Tables to Simple Tables**

Use this operation to convert DITA CALS tables to simple tables. If you invoke this operation from a nested table (a table inside a table), only the nested table will be affected. If it is invoked on a parent table that contains nested tables, all of the contained tables will be converted.

**Convert conrefs to conkeyrefs**

Use this operation to convert @conref attributes to @conkeyref attributes.

**Convert Simple Tables to CALS Tables**

Use this operation to convert DITA simple tables to CALS tables. If you invoke this operation from a nested table (a table inside a table), only the nested table will be affected. If it is invoked on a parent table that contains nested tables, all of the contained tables will be converted.

**Convert to Concept**

Use this operation to convert a DITA topic (of any type) to a DITA Concept topic type (for example, Topic to Concept).

**Convert to General Task**

Use this operation to convert a DITA topic (of any type) to a DITA General Task topic type (for example, Task to General Task). A DITA General Task is a less restrictive alternative to the Strict Task information type.

**Convert to Reference**

Use this operation to convert a DITA topic (of any type) to a DITA Reference topic type (for example, Topic to Reference).

**Convert to Task**

Use this operation to convert a DITA topic (of any type) to a DITA Task topic type (for example, Topic to Task).

**Convert to Topic**

Use this operation to convert a DITA topic (of any type) to a DITA Topic (for example, Task to Topic).
Convert to Troubleshooting

Use this operation to convert a DITA topic (of any type) to a DITA Troubleshooting topic type (for example, Topic to Troubleshooting).

Rename Key

Available when invoked on a key, and can be used to quickly rename a key. It also updates all references to it. Note that it does not work on DITA 1.3 key scopes.

Generate IDs

Use this operation to automatically generate unique IDs for elements.

Attributes Refactoring Actions

Contains built-in XML refactoring operations that pertain to attributes with some of the information preconfigured based upon the current context.

Add/Change attribute

Allows you to change the value of an attribute or insert a new one.

Convert attribute to element

Allows you to change an attribute into an element.

Delete attribute

Allows you to remove one or more attributes.

Rename attribute

Allows you to rename an attribute.

Replace in attribute value

Allows you to search for a text fragment inside an attribute value and change the fragment to a new value.

Comments Refactoring Actions

Contains built-in XML refactoring operations that pertain to comments with some of the information preconfigured based upon the current context.

Delete comments

Allows you to delete comments found inside one or more elements.

Elements Refactoring Actions

Contains built-in XML refactoring operations that pertain to elements with some of the information preconfigured based upon the current context.

Delete element

Allows you to delete elements.
Delete element content

Allows you to delete the content of elements.

Insert element

Allows you to insert new elements.

Rename element

Allows you to rename elements.

Unwrap element

Allows you to remove the surrounding tags of elements, while keeping the content unchanged.

Wrap element

Allows you to surround elements with element tags.

Wrap element content

Allows you to surround the content of elements with element tags.

Fragments Refactoring Actions

Contains built-in XML refactoring operations that pertain to XML fragments with some of the information preconfigured based upon the current context.

Insert XML fragment

Allows you to insert an XML fragment.

Replace element content with XML fragment

Allows you to replace the content of elements with an XML fragment.

Replace element with XML fragment

Allows you to replace elements with an XML fragment.

JATSKit Refactoring Actions

Available for JATS documents. Contains built-in XML refactoring operations that pertain to JATS documents with some of the information preconfigured based upon the current context.

Add BITS DOCTYPE - NLM/NCBI Book Interchange 2.0

Adds an NLM 'BITS' 2.0 DOCTYPE declaration.

Add Blue DOCTYPE - NISO JATS Publishing 1.1

Adds a JATS 'Blue' 1.1 DOCTYPE declaration.

Normalize IDs

Assigned IDs are normalized and IDs are assigned to some elements that are missing them.
**Processing Instructions Refactoring Actions**

Contains built-in XML refactoring operations that pertain to processing instructions.

**Accept all tracked changes, remove all Oxygen-specific comments and highlights**

Use this operation to accept all application-specific tracked changes (from elements and attributes) or remove all application-specific comments or highlights.

**Delete processing instructions**

Deletes the detected processing instructions.

**Review submenu**

This submenu includes the following actions:

- **Track Changes**
  Enables or disables the Track Changes (on page 3224) support for the current document.

- **Accept Change(s) and Move to Next**
  Accepts the Tracked Change (on page 3224) located at the cursor position or all of the changes in a selection and then moves to the next change. If you select a part of a deletion or insertion change, only the selected content is accepted.

- **Accept All Changes**
  Accepts all Tracked Changes (on page 3224) in the current document.

- **Reject Change(s) and Move to Next**
  Rejects the Tracked Change (on page 3224) located at the cursor position or all of the changes in a selection and then moves to the next change. If you select a part of a deletion or insertion change, only the selected content is rejected.

- **Reject All Changes**
  Rejects all Tracked Changes (on page 3224) in the current document.

- **Comment Change**
  Opens a dialog box that allows you to add a comment to an existing Tracked Change (on page 3224). The comment will appear in a callout and a tooltip when hovering over the change. If the action is selected on an existing commented change, the dialog box will allow you to edit the comment.

- **Highlight**
  Enables the highlighting tool that allows you to mark text in your document.
Colors

Allows you to select the color for highlighting text.

Stop highlighting

Use this action to deactivate the highlighting tool.

Remove highlight(s)

Use this action to remove highlighting from the document.

Add Comment

Inserts a comment at the cursor position. The comment appears in a callout box and a tooltip (when hovering over the change).

Show/Edit Comment

Opens a dialog box that displays the discussion thread and allows the current user to edit comments that do not have replies. If you are not the author who inserted the original comment, the dialog box just displays the comment without the possibility of editing it.

Remove Comment

Removes a selected comment. If you remove a comment that contains replies, all of the replies will also be removed.

Manage Reviews

Opens the Review view (on page 664).

Manage IDs submenu

This submenu is available for XML documents that have an associated DTD, XML Schema, or Relax NG schema. It includes the following actions:

Rename

Renames the ID and all its occurrences. Selecting this action opens the Rename XML ID dialog box. This dialog box lets you insert the new ID value and choose the scope of the rename operation.

Search References

Searches for the references of the ID. By default, the scope of this action is the current project. If you configure a scope using the Select the scope for the Search and Refactor operations (on page 831) dialog box, this scope will be used instead.

Search References in

Searches for the references of the ID. Selecting this action opens the Select the scope for the Search and Refactor operations (on page 831).
Search Occurrences in file

Searches for the occurrences of the ID in the current document.

Folding submenu

This submenu includes the following actions:

- **Toggle Fold**
  Toggles the state of the current fold.

- **Collapse Other Folds (Ctrl + NumPad/ (Command + NumPad/ on macOS))**
  Folds all the elements except the current element.

- **Collapse Child Folds (Ctrl + NumPad. (Command + NumPad. on macOS))**
  Folds the elements indented with one level inside the current element.

- **Expand Child Folds**
  Unfolds all child elements of the currently selected element.

- **Expand All (Ctrl + NumPad* (Command + NumPad* on macOS))**
  Unfolds all elements in the current document.

Inspect Styles

Opens the CSS Inspector view (on page 640) that allows you to examine the CSS rules that match the currently selected element.

Options

Opens the Author mode preferences page (on page 176).

Document Type-Specific Contextual Menu Actions in Author Mode

Other document type-specific actions are available in the contextual menu of Author mode for the following document types (click the links to see the default actions that are available for each specific document type):

- [DocBook4 Author Actions](on page 1293)
- [DocBook5 Author Actions](on page 1314)
- [DITA Author Actions](on page 3001)
- [DITA Map Author Actions](on page 2947)
- [XHTML Author Actions](on page 1375)
- [TEI ODD Author Actions](on page 1402)
- [TEI P5 Author Actions](on page 1389)
- [JATS Author Actions](on page 1415)

Validating XML Documents

The W3C XML specification states that a program should not continue to process an XML document if it finds a validation error. The reason is that XML software should be easy to write and all XML documents should
be compatible. With HTML, for example, it is possible to create documents with lots of errors (for instance, when you forget an end tag). One of the main reasons that various HTML browsers have performance and compatibility problems is that they have different methods of figuring out how to render a document when an HTML error is encountered. Using XML helps to eliminate such problems.

Even when creating XML documents, errors are easily introduced. When working with large projects or a large number of files, the probability that errors will occur is even greater. Preventing and solving errors in your projects can be time consuming and frustrating. Fortunately, Oxygen XML Editor provides validation functions that allow you to easily identify errors and their location.

Related Information:
Modular Contextual XML Editing Using 'Main Files' Support (on page 829)

Checking XML Well-Formedness

A *Well-formed XML* document is a document that conforms to the XML syntax rules. A *Namespace Well-Formed XML* document is a document that is *Well-formed XML* and is also *Namespace-wellformed* and *Namespace-valid*.

**Well-Formedness Rules**

The XML Syntax rules for *Well-formed XML* include:

- All XML elements must have a closing tag.
- XML tags are case-sensitive.
- All XML elements must be properly nested.
- All XML documents must have a root element.
- Attribute values must always be quoted.
- With XML, whitespace is preserved.

The *Namespace-wellformed* rules include:

- All element and attribute names contain either zero or one colon.
- No entity names, processing instruction targets, or notation names contain any colons.

The *Namespace-valid* rules include:

- The *xml* prefix is by definition bound to the namespace name: `http://www.w3.org/XML/1998/namespace`. It MAY be declared, but MUST NOT be undeclared or bound to any other namespace name. Other prefixes MUST NOT be bound to this namespace name.
- The *xmlns* prefix is used only to declare namespace bindings and is by definition bound to the namespace name: `http://www.w3.org/2000/xmlns/`. It MUST NOT be declared or undeclared. Other prefixes MUST NOT be bound to this namespace name.
• All other prefixes beginning with the three-letter sequence \( x, m, l \), in any case combination, are reserved. This means that users SHOULD NOT use them except as defined by later specifications and processors MUST NOT treat them as fatal errors.

• The namespace prefix (unless it is \( xml \) or \( xmlns \)) MUST have been declared in a namespace declaration attribute in either the start tag of the element where the prefix is used or in an ancestor element (for example, an element in whose content the prefixed markup occurs). Furthermore, the attribute value in the innermost such declaration MUST NOT be an empty string.

Check for Well-Formedness

To check if a document is Namespace Well-Formed XML and Namespace-valid:

• Select the \( \text{Check Well-Formedness (Ctrl + Shift + W (Command + Shift + W on macOS))} \) action from the Validation drop-down menu on the toolbar (or the Document > Validate menu).

• A selection of files can be checked for well-formedness by selecting the \( \text{Check Well-Formedness} \) action from the Validate submenu when invoking the contextual menu in the Project view (on page 402).

Result: If any errors are found, the result is displayed in the message panel at the bottom of the editor. Each error is displayed as one record in the result list and is accompanied by an error message. Clicking the record will open the document containing the error and highlight its approximate location.

Example: A non Well-formed XML Document

\[
<\text{root}><\text{tag}></\text{root}>
\]

When Check Well-Formedness is performed the following error is raised:

The element type "tag" must be terminated by the matching end-tag "</tag>"

To resolve the error, click the record in the result list and it will locate and highlight the approximate position of the error. In this case, identify the tag that is missing an end tag and insert \( </\text{tag}> \).

Example: A non Namespace-wellformed Document

\[
<\text{prefix:elem}></\text{prefix:elem}>
\]

When Check document form is performed the following error is raised:

The prefix "prefix" for element "prefix:elem" is not bound.

Example: A non Namespace-valid Document

\[
<x:y></x:y>
\]

When Check document form is performed the following error is raised:

The prefix "x" for element "x:y" is not bound.
Validating XML Documents Against a Schema

A **Valid** XML document is a **Well-Formed** XML document that also conforms to the rules of a schema that defines the legal elements of an XML document. The schema type can be: XML Schema, Relax NG (full or compact syntax), Schematron, Document Type Definition (DTD), or Namespace-based Validation Dispatching Language (NVDL).

The purpose of the schema is to define the legal building blocks of an XML document. It defines the document structure with a list of legal elements.

This section contains topics that explain the automatic and manual validation possibilities in Oxygen XML Editor, how validation errors are presented, and information about built-in and custom validation scenarios.

For information about how to associate a schema for the purposes of validation (and content completion), see the [Associating a Schema to XML Documents](on page 816) section.

Automatic Validation

By default, Oxygen XML Editor automatically checks for validation errors as you are editing a document. The **Enable automatic validation** option *(on page 226)* in the **Document Checking** preferences page *(on page 226)* controls whether or not all validation errors and warnings will automatically be highlighted in the editor panel.

The automatic validation starts parsing the document and marking the errors after a **configurable delay** *(on page 226)* from the last key typed. Errors are highlighted with underline markers in the main editor panel and small rectangles on the right side ruler of the editor panel. Hovering over a validation error presents a tooltip message with more details about the error.

If the error message is long and it is not displayed completely in the error line at the bottom of the editing area, double-clicking the error icon at the left of the error line or on the error line displays an information dialog box with the full error message. You can use the arrow buttons in this dialog box to navigate through the errors issued by the Automatic Validation feature.

Related Information:
- [Manual Validation Actions](on page 775)
- [Presenting Validation Errors in Text Mode](on page 777)
- [Presenting Validation Errors in Author Mode](on page 780)

Manual Validation Actions

You can choose to validate documents at any time by using the manual validation actions that are available in Oxygen XML Editor.
Tip:
Status information generated by certain operations (such as validation) are fed into the Information view. This could be helpful for troubleshooting problems encountered during such operations. To open this view, select Window > Show View > Information.

Manual Validation Actions

To manually validate the currently edited document, use one of the following actions:

- **Validate (Ctrl + Shift + V (Command + Shift + V on macOS))**
  Available from the Validation drop-down menu on the toolbar, the Document > Validate menu, or from the Validate submenu when invoking the contextual menu in the Project view (on page 402).
  An error list is presented in the message panel at the bottom of the editor. Markup of the current document is checked to conform with the specified DTD, XML Schema, or Relax NG schema rules. This action also re-parses the XML Catalogs (on page 3225) and resets the schema used for content completion.

- **Validate (cached)**
  Available from the Validation drop-down menu on the toolbar or the Document > Validate menu.
  This action caches the schema, allowing it to be reused for the next validation. Markup of the current document is checked to conform with the specified DTD, XML Schema, or Relax NG schema rules.

  **Note:**
  Automatic validation also caches the associated schema.

- **Validate with**
  Available from the Validation drop-down menu on the toolbar, (or Document > Validate menu).
  This action opens a dialog box that allows you to specify a schema for validating the current document (on page 819).
  You can use this action to validate the current document using a schema of your choice (XML Schema, DTD, Relax NG, NVDL, Schematron schema), other than the associated one. An error list is presented in the message panel at the bottom of the editor. Markup of current document is checked to conform with the specified schema rules.
Note:
The Validate with action does not work for files loaded through a custom protocol plugin (on page 2427) developed independently and added to Oxygen XML Editor after installation.

Validate with Schema

Available from the Validate submenu when invoking contextual menu in the Project view (on page 402).

This action opens a dialog box that allows you to specify a schema for validating all selected files (on page 820).

Other Validation Options

To quickly open the schema used for validating the current document, select the Open Associated Schema action from the toolbar (or Document > Schema menu).

The Validation options button, available in the Document > Validate menu, allows you to quickly access to the validation options (on page 237) for the built-in validator in the Oxygen XML Editor preferences page.

Tip:
If a large number of validation errors are detected and the validation process takes too long, you can limit the maximum number of reported errors in the Document Checking preferences page (on page 226).

Related information
Automatic Validation (on page 775)
Presenting Validation Errors in Text Mode (on page 777)
Presenting Validation Errors in Author Mode (on page 780)

Presenting Validation Errors in Text Mode

By default, Oxygen XML Editor automatically validates documents (on page 775) while editing in the Text mode, and actions are also available to manually validate documents (on page 775) on-request.
Validation Marker Locations

In Text mode, validation issues are marked in the following locations:

- In the main editing pane, with the issue underlined in a color according to the type of issue.
- In the right-side vertical stripe, with a marker that is colored according to the type of issue.
- For attributes with detected issues, in the Attributes view (on page 542), with the attribute and its value colored according to the type of issue.

Validation Marker Colors

The colors for each type of issue are as follows:

- **Validation Errors** [Red] - By default, the underline in the editing pane, the marker in the right vertical stripe, and the foreground color of the attribute in the Attributes view are colored in red.
- **Validation Warnings** [Yellow] - By default, the underline in the editing pane, the marker in the right vertical stripe, and the foreground color of the attribute in the Attributes view are colored in yellow.
- **Validation Info** [Blue] - By default, the underline in the editing pane, the marker in the right vertical stripe, and the foreground color of the attribute in the Attributes view are colored in blue.

You can configure the color for each type in the Document Checking preferences page (on page 226).

Validation Markers in the Right-Side Stripe

Also, the stripe on the right side of the editor panel is designed to display the issues found during the validation process and to help you locate them in the document. The stripe contains the following:

**Upper Part of the Stripe**

A success indicator square will turn green if the validation is successful or only info messages are found, red if validation errors are found, or yellow if only validation warnings are found. More
details about the issues are displayed in a tooltip when you hover over indicator square. If there are numerous problems, only the first three are presented in the tooltip.

Middle Part of the Stripe

Errors are presented with red markers, warnings with yellow markers, and info message with blue markers. If you want to limit the number of markers that are displayed, open the Preferences dialog box (Options > Preferences) (on page 124), go to Editor > Document checking, and specify the desired limit in the Maximum number of validation highlights option (on page 226).

Clicking a marker will highlight the corresponding text area in the editor. The validation message is also displayed both in a tooltip (when hovering over the marker) and in the message area on the bottom of the editor panel (clicking the Document checking options button opens the Document Checking preferences page (on page 226).

Bottom Part of the Stripe

Two navigation arrows ( ) can be used to jump to the next or previous issue. The same actions can be triggered from Document > Automatic validation > Next Error/Highlight (Ctrl + Period (Command + Period on macOS)) and Document > Automatic validation > Previous Error/Highlight (Ctrl + Comma (Command + Comma on macOS)). Also, the Remove All button can be used to clear all the validation markers.

Hovering Over Validation Issues

Hovering over a validation issue presents a tooltip message with more details about the problem and possible quick fixes (on page 813) (if available for that issue). Also, when hovering over an issue, pressing F2 will change the focus to the tooltip where you can use Tab and Shift + Tab to navigate between quick fixes and Space to trigger them.

Details About Validation Issues

- Information about the issue is also displayed in the message area on the bottom of the editor panel (clicking the Document checking options button opens the Document Checking preferences page (on page 226) where you can configure some validation options (such as the colors used to present the validation issues). Some validation messages have an icon ( ) and clicking it opens a dialog box with additional information and a link to specifications.
- When a validation is processed, information about the validation scenario is displayed in the stripe at the very bottom of the application. It includes the name of the validation scenario and its status. If you hover over the information, a tooltip is presented with more information. You can also click the button to open the Information view (on page 512), where even more details are displayed.
• If you want to see all the validation messages grouped in the Results view (on page 548), use the Validate action from the toolbar or Document > Validate menu. To see more information about a validation message, right-click the item in the Results view and select Show message. Some validation messages have an icon (⯄) in the Info column and clicking it opens a dialog box with additional information and a link to specifications.

Related Information:
Validating XML Documents Against a Schema (on page 775)
Presenting Schematron Validation Issues (on page 1206)

Presenting Validation Errors in Author Mode

By default, Oxygen XML Editor automatically validates documents (on page 775) while editing in the Author mode, and actions are also available to manually validate documents (on page 775) on-request.

Figure 219. Presenting Validation Errors in Author Mode

Validation Marker Locations

In Author mode, validation issues are marked in the following locations:
In the main editing pane, with the issue underlined in a color according to the type of issue.

In the right-side vertical stripe, with a marker that is colored according to the type of issue.

For attributes with detected issues, in the Attributes view (on page 627), with the attribute and its value colored according to the type of issue.

Validation Marker Colors

The colors for each type of issue are as follows:

- **Validation Errors** [Red] - By default, the underline in the editing pane, the marker in the right vertical stripe, and the foreground color of the attribute in the Attributes view are colored in red.

- **Validation Warnings** [Yellow] - By default, the underline in the editing pane, the marker in the right vertical stripe, and the foreground color of the attribute in the Attributes view are colored in yellow.

- **Validation Info** [Blue] - By default, the underline in the editing pane, the marker in the right vertical stripe, and the foreground color of the attribute in the Attributes view are colored in blue.

Validation Markers in the Right-Side Stripe

Also, the stripe on the right side of the editor panel is designed to display the issues found during the validation process and to help you locate them in the document. The stripe contains the following:

**Upper Part of the Stripe**

A success indicator square will turn green if the validation is successful or only info messages are found, red if validation errors are found, or yellow if only validation warnings are found. More details about the issues are displayed in a tooltip when you hover over indicator square. If there are numerous problems, only the first three are presented in the tooltip.

**Middle Part of the Stripe**

Errors are presented with red markers, warnings with yellow markers, and info message with blue markers. If you want to limit the number of markers that are displayed, open the Preferences dialog box (Options > Preferences) (on page 124), go to Editor > Document checking, and specify the desired limit in the Maximum number of validation highlights option (on page 226).

Clicking a marker will highlight the corresponding text area in the editor. The validation message is also displayed both in a tooltip (when hovering over the marker) and in the message area on the bottom of the editor panel (clicking the Document checking options button opens the Document Checking preferences page (on page 226).

**Bottom Part of the Stripe**

Two navigation arrows (↵) can be used to jump to the next or previous issue. The same actions can be triggered from Document > Automatic validation > Next Error/Highlight (Ctrl + Period (Command + Period on macOS)) and Document > Automatic validation > Previous Error/Highlight (Ctrl + Comma (Command + Comma on macOS)). Also, the Remove All button can be used to clear all the validation markers.
You can configure the color for each type in the Document Checking preferences page (on page 226).

**Hovering Over Validation Issues**

Hovering over a validation issue presents a tooltip message with more details about the problem and possible quick fixes (on page 813) (if available for that issue). Also, when hovering over an issue, pressing F2 will change the focus to the tooltip where you can use Tab and Shift + Tab to navigate between quick fixes and Space to trigger them.

**Details About Validation Issues**

- Information about the issue is also displayed in the message area on the bottom of the editor panel (clicking the Document checking options button opens the Document Checking preferences page (on page 226) where you can configure some validation options (such as the colors used to present the validation issues). Some validation messages have an icon and clicking it opens a dialog box with additional information and a link to specifications.
- When a validation is processed, information about the validation scenario is displayed in the stripe at the very bottom of the application. It includes the name of the validation scenario and the status. If you hover over the information, a tooltip is presented with more information. You can also click the button to open the Information view (on page 512), where even more details are displayed.

![Example of a validation message]

- If you want to see all the validation messages grouped in the Results view (on page 548), use the Validate action from the toolbar or Document > Validate menu. To see more information about a validation message, right-click the item in the Results view and select Show message. Some validation messages have an icon in the Info column and clicking it opens a dialog box with additional information and a link to specifications.

**Related Information:**

- Validating XML Documents Against a Schema (on page 775)
- Presenting Schematron Validation Issues (on page 1206)
Customizing Assert Error Messages

To customize the error messages that the Xerces or Saxon validation engines display for the `<assert>` and `<assertion>` elements, set the `@message` attribute on these elements.

- For Xerces, the `@message` attribute has to belong to the `http://xerces.apache.org` namespace.
- For Saxon, the `@message` attribute has to belong to the `http://saxon.sourceforge.net/` namespace.

The value of the `@message` attribute is the error message displayed if the assertion fails.

Custom Validators

If you need to validate the edited document with a validation engine that is different from the built-in engine, you can configure external validators in the Custom Validation Engines preferences page (on page 227). After a custom validation engine is properly configured (on page 227), it can be applied on the current document by selecting it from the list of custom validation engines in the Validation toolbar drop-down menu. The document is validated against the schema declared in the document.

Some validators are configured by default but there are third-party processors that do not support the output message format (on page 784) of Oxygen XML Editor for linked messages:

- **Saxon-EE** - Included in Oxygen XML Editor. It is associated to XML Editor and XSD Editor. It is able to validate XML Schema schemas and XML documents against XML Schema schemas. The validation is done according to the W3C XML Schema 1.0 or 1.1. This can be configured in Preferences (on page 238).
- **MSXML 4.0 (Legacy)** - Included in Oxygen XML Editor (Windows edition only). It is associated to XML Editor, XSD Editor and XSL Editor. It is able to validate the edited document against XML Schema, internal DTD (included in the XML document), external DTD or a custom schema type.
- **MSXML.NET (Legacy)** - Included in Oxygen XML Editor (Windows edition only). It is associated to XML Editor, XSD Editor and XSL Editor. It is able to validate the edited document against XML Schema, internal DTD (included in the XML document), external DTD or a custom schema type.
- **LIBXML** - Not included in Oxygen XML Editor and, depending on your operating system, the libraries need to be downloaded and installed separately from [http://xmlsoft.org/downloads.html](http://xmlsoft.org/downloads.html). Afterward, the `PATH` environment variable needs to be updated to contain the parent folder of the `xmllint` executable. Alternatively, you can go to Options > Preferences > Editor > Custom Validation Engines, edit the LIBXML validation engine and set a custom path to the `xmllint` executable.

The LIBXML validator is associated with the XML Editor. It is able to validate the edited document against XML Schema, Relax NG schema full syntax, internal DTD (included in the XML document) or a custom schema type. Support for XML Catalogs (on page 3225) (the `--catalogs` parameter) and XInclude processing (`--xinclude`) are enabled by default in the preconfigured LIBXML validator. The `--postvalid` parameter is also set by default and it allows LIBXML to validate correctly the main document even if the XInclude fragments contain IDREFS to ID’s located in other fragments.
For validation against an external DTD specified by URI in the XML document, add the `--dtdvalid` parameter manually to the DTD validation command line. `$ds` represents the detected DTD declaration in the XML document.

**CAUTION:**
File paths containing spaces are not handled correctly in the LiBXXML processor. For example, the built-in *XML Catalog (on page 3225)* files of the built-in document types (DocBook, TEI, DITA, etc.) are not handled by LIBXML if Oxygen XML Editor is installed in the default location on Windows *(C:\Program Files)* because the built-in *XML catalog* files are stored in the *frameworks* subfolder of the installation folder and in this case, the file path contains at least one space character.

---

**Attention:**
On macOS, if the full path to the LIBXML executable file is not specified in the **Executable path** text field, some errors may occur during validation against a W3C XML Schema, such as:

```
Unimplemented block at ... xmlschema.c
```

To avoid these errors, specify the full path to the LIBXML executable file.

• **XSV (Legacy)** - Not included in Oxygen XML Editor. Windows and Linux distributions of XSV can be downloaded from [http://www.cogsci.ed.ac.uk/~ht/xsv-status.html](http://www.cogsci.ed.ac.uk/~ht/xsv-status.html). The executable path is already configured in Oxygen XML Editor *(on page 227)* for the `{OXYGEN_INSTALL_DIR}/xsv` installation folder. If it is installed in a different folder, the predefined executable path must be corrected in Preferences. *(on page 227)* It is associated to XML Editor and XSD Editor. It is able to validate the edited document against XML Schema or a custom schema type.

• **SQC [Schema Quality Checker from IBM] (Legacy)** - Not included in Oxygen XML Editor. It can be downloaded from here (it comes as a .zip file, at the time of this writing SQC2.2.1.zip is about 3 megabytes). The executable path and working directory are already configured for the SQC installation directory `{OXYGEN_INSTALL_DIR}/sqc`. If it is installed in a different folder, the predefined executable path and working directory must be corrected in the Preferences page. *(on page 227)* It is associated to XSD Editor.

A custom validator cannot be applied on files loaded through an Oxygen XML Editor custom protocol plugin *(on page 2427)* developed independently and added to Oxygen XML Editor after installation.

**Linked Output Messages of an External Engine**

Validation engines display messages in an output view at the bottom of the Oxygen XML Editor window. If such an output message (*warning*, *error*, *fatal error*, etc) spans between three to six lines of text and has the format specified below, then the message is linked to a location in the validated document. Clicking the message in the output view highlights the location of the message in an editor panel containing the file
referenced in the message. This behavior is similar to the linked messages generated by the default built-in validator.

Linked messages have the following format:

- **Type**: [F|E|W] (the string Type: followed by a letter for the type of the message: fatal error, error, warning). This property is optional in a linked message.
- **SystemID**: A system ID of a file (the string SystemID: followed by the system ID of the file that will be opened for highlighting when the message is clicked in the output message - usually the validated file, the schema file or an included file).
- **Line**: A line number (the string Line: followed by the number of the line that will be highlighted).
- **Column**: A column number (the string Column: followed by the number of the column where the highlight will start on the highlighted line). This property is optional in a linked message.
- **EndLine**: A line number (the string EndLine: followed by the number of the line where the highlight ends). This property is optional in a linked message.
- **EndColumn**: A column number (the string EndColumn: followed by the number of the column where the highlight ends on the end line). This property is optional in a linked message.
- **AdditionalInfoURL**: The URL string pointing to a remote location where additional information about the error can be found - this line is optional in a linked message.
- **Description**: Message content (the string Description: followed by the content of the message that will be displayed in the output view).

**Example:**

Example of how a custom validation engine can report an error using the format specified above:

```
Type: E
SystemID: file:///c:/path/to/validatedFile.xml
Line: 10
Column: 20
EndLine: 10
EndColumn: 35
AdditionalInfoURL: http://www.host.com/path/to/errors.html#errorID
Description: custom validator message
```
Using Saxon Integrated Extension Functions

Saxon, the transformation and validation engine used by Oxygen XML Editor, can be customized by adding custom functions (called Integrated Extension Functions) that can be called from XPath.

To define such a function, follow these steps:

1. Create a file with a Java class that extends `net.sf.saxon.lib.ExtensionFunctionDefinition`. Here is an example:

   ```java
   private static class ShiftLeft extends ExtensionFunctionDefinition {
      @Override
      public StructuredQName getFunctionQName() {
         return new StructuredQName("eg", "http://example.com/saxon-extension", "shift-left");
      }

      @Override
      public SequenceType[] getArgumentTypes() {
         return new SequenceType[]{SequenceType.SINGLE_INTEGER, SequenceType.SINGLE_INTEGER};
      }

      @Override
      public SequenceType getResultType(SequenceType[] suppliedArgumentTypes) {
         return SequenceType.SINGLE_INTEGER;
      }

      @Override
      public ExtensionFunctionCall makeCallExpression() {
         return new ExtensionFunctionCall() {
            public SequenceIterator call(SequenceIterator[] arguments, XPathContext context) throws XPathException {
               long v0 = ((IntegerValue)arguments[0].next()).longValue();
               long v1 = ((IntegerValue)arguments[1].next()).longValue();
               long result = v0<<v1;
               return Value.asIterator(Int64Value.makeIntegerValue(result));
            }
         };
      }
   }
   ```

2. Compile the class and add it to a JAR file.
3. Add a file called `net.sf.saxon.lib.ExtensionFunctionDefinition` that contains the fully qualified name of the Java class in the `META-INF/services/` folder of the JAR file.

**Note:**
To add more function definitions in the same JAR file, you need to add their fully qualified names on different lines.

To enable Oxygen XML Editor to pick up your custom function definition, the JAR file should be added to the classpath of the transformer. Here are some possibilities:

- If you develop a framework, you just need to link the JAR file in the **Classpath** tab (on page 144).
- In a validation scenario (on page 788), you can use the **Extensions** button to open a dialog box where you can add libraries.
- In a transformation scenario, you can use the **Extensions** button in the **XSLT** tab (on page 1466) to open a dialog box where you can add libraries.
- You can also create a plugin that contributes such a JAR file in the classpath (on page 2411).

**Validation Scenarios**

A complex XML document is split in smaller interrelated modules. These modules do not make much sense individually and cannot be validated in isolation due to interdependencies with other modules. Oxygen XML Editor validates the main module of the document when an imported module is checked for errors.

A typical example is the chunking of a DocBook XSL stylesheet that has `chunk.xsl` as the main module and `param.xsl`, `chunk-common.xsl`, and `chunk-code.xsl` as imported modules. `param.xsl` only defines XSLT parameters. The module `chunk-common.xsl` defines an XSLT template with the name `chunk`. `chunk-code.xsl` calls this template. The parameters defined in `param.xsl` are used in the other modules without being redefined.

Validating `chunk-code.xsl` as an individual XSLT stylesheet generates misleading errors regarding parameters and templates that are used but undefined. These errors are only caused by ignoring the context in which this module is used in real XSLT transformations and validations. To validate such a module, define a validation scenario to set the main module of the stylesheet and the validation engine used to find the errors. Usually this engine applies the transformation during the validation process to detect the errors that the transformation generates.

You can validate a stylesheet with several engines to make sure that you can use it in various environments and have the same results. For example, an XSLT stylesheet may be applied with Saxon 6.5, Xalan, and MSXML 4.0 engines in different production systems.

Other examples of documents that can benefit from a validation scenario include:
A complex XQuery file with a main module that imports modules developed independently but validated in the context of the main module of the query. In an XQuery validation scenario, the default validator of Oxygen XML Editor (Saxon 9) or any connection to a database that supports validation (eXist XML Database, MarkLogic version 5 or newer) can be set as a validation engine.

An XML document where the main file (on page 3220) includes smaller fragment files using XML entity references.

**Note:**
If a main file is associated with the current file, the validation scenarios defined in the main file, along with any Schematron schema defined in the default scenarios for that particular framework, are used for the validation. These take precedence over other types of validation units defined in the default scenarios for the particular framework. For more information on main files, see Contextual Project Operations Using 'Main Files' Support (on page 418) or Modular Contextual XML Editing Using 'Main Files' Support (on page 829).

**Tip:**
Status information generated by certain operations (such as validation) are fed into the Information view. This could be helpful for troubleshooting problems encountered during such operations. To open this view, select Window > Show View > Information.

Related information
- Validating XML Documents Against a Schema (on page 775)
- Presenting Validation Errors in Author Mode (on page 780)
- Presenting Validation Errors in Text Mode (on page 777)

**Creating a New Validation Scenario**

To create a validation scenario, follow these steps:

1. Select the Configure Validation Scenario(s) from the Validation toolbar drop-down menu, or from the Document > Validate menu (or the Validate submenu when invoking the contextual menu on a file in the Project view (on page 402)).

The Configure Validation Scenario(s) dialog box is displayed. It contains built-in and user-defined scenarios. The built-in scenarios are organized in categories depending on the type of file they apply to and you can identify them by a yellow key icon that marks them as read-only. The user-defined scenarios are organized under a single category. The default scenarios for the particular framework (on page 3219) are rendered in bold.

**Note:**
If a main file is associated with the current file, the validation scenarios defined in the main file, along with any Schematron schema defined in the default scenarios for that particular
framework, are used for the validation. These take precedence over other types of validation units defined in the default scenarios for the particular framework. For more information on main files, see Contextual Project Operations Using 'Main Files' Support (on page 418) or Modular Contextual XML Editing Using 'Main Files' Support (on page 829).

Figure 220. Configure Validation Scenario Dialog Box

The top section of the dialog box contains a filter that allows you to search through the scenarios list and the Settings button allows you to configure the following options:

**Show all scenarios**
Select this option to display all the available scenarios, regardless of the document they are associated with.

**Show only the scenarios available for the editor**
Select this option to only display the scenarios that Oxygen XML Editor can apply for the current document type.

**Show associated scenarios**
Select this option to only display the scenarios associated with the document you are editing.

**Import scenarios**
This option opens the Import scenarios dialog box that allows you to select the scenarios file that contains the scenarios you want to import. If one of the scenarios you import is identical to an existing scenario, Oxygen XML Editor ignores it. If a conflict appears (an imported scenario has the same name as an existing one), you can choose between two options:
Keep or replace the existing scenario.
- Keep both scenarios.

Note:
When you keep both scenarios, Oxygen XML Editor adds imported to the name of the imported scenario.

Export selected scenarios

Use this option to export selected scenarios individually. Oxygen XML Editor creates a scenarios file that contains the exported scenarios. This is useful if you want to share scenarios with others or export them to another computer.

2. To add a scenario, click the New button.
   A validation scenario configuration dialog box is displayed and it lists all the validation units for the scenario.

   Figure 221. Validation Scenario Configuration Dialog Box

This scenario configuration dialog box allows you to configure the following information and options:

Name
The name of the validation scenario.

Storage
You can choose between storing the scenario in the Project Options (on page 3222) or Global Options (on page 3219).

URL of the file to validate
The URL of the main module that includes the current module. It is also the entry module of the validation process when the current one is validated. To edit the URL, double-click its cell and specify the URL of the main module by doing one of the following:
Enter the URL in the text field or select it from the drop-down list.

Use the **Browse** drop-down button to browse for a local, remote, or archived file.

Use the **Insert Editor Variable** button to insert an editor variable (on page 323) or a custom editor variable (on page 332).

**Figure 222. Insert an Editor Variable**

```
$(Desktop) - My Desktop
$(start-dir) - Start directory of custom validator
$(standard-params) - List of standard params for command line
$(fn) - The current file name without extension
$(currentFileURL) - The path of the currently edited file (URL)
$(fdir) - The path of current file directory (URL)
$(framework) - Oxygen frameworks directory (URL)
$(pdir) - Project directory (URL)
$(oxygenHome) - Oxygen installation directory (URL)
$(home) - The path to user home directory (URL)
$(pn) - Project name
$(env(VAR_NAME)) - Value of environment variable VAR_NAME
$(system(var.name)) - Value of system variable var.name
```

**File type**

The type of the document that is validated in the current validation unit. Oxygen XML Editor automatically selects the file type depending on the value of the **URL of the file to validate** field.

**Validation engine**

You can select one of the engines available in Oxygen XML Editor for validation of the particular document type:

- **Default engine** - The default engine is used to run the validation for the current document type, as specified in the preferences page for that type of document (for example, XSLT preferences page (on page 245), XQuery preferences page (on page 253), XML Schema preferences page (on page 238)).

- **DITA Validation** engine - Performs DITA-specific checks in the context of the specifications (it is similar to the process when using the Validate and Check for Completeness action (on page 2941) in the DITA Maps Manager, but for a local file rather than an entire DITA map (on page 3218)).

- **DITA Map Validation and Completeness Check** engine - Performs a validation process that checks the DITA map document and all referenced topics and maps (it is similar to the process when using the Validate and Check for Completeness action (on page 2941) in the DITA Maps Manager).
• **DITA-OT Project Validation and Completeness Check** engine - Performs a validation process that checks each context from the provided DITA-OT project file (it is similar to the process when using the Validate and Check for Completeness action (on page 2941) in the DITA Maps Manager).

• **Table Layout Validation** engine - Looks for table layout problems (for more information, see the Report table layout problems option (on page 2946)).

**Automatic validation**

If this option is selected, the validation operation defined by this row is also applied by the automatic validation feature (on page 775). If the Automatic validation feature is disabled in the Document Checking preferences page (on page 226), then this option is ignored, as the preference setting has a higher priority.

**Schema**

This option becomes active when you set the File type to XML Document and allows you to specify the schema used for the validation unit.

**Settings**

Depending on the selected validation engine, clicking the Settings button either opens the Specify Schema dialog box or the Configure validation engine dialog box.

• **Specify Schema Dialog Box**

This dialog box allows you to specify a custom schema to be used for the validation process.

**Figure 223. Specify Schema Dialog Box**

The Specify Schema dialog box contains the following options:

**Use detected schema**

Uses the schema detected for the particular document (on page 817).

**Use custom schema**
Allows you to specify the schema using the following options:

- **URL** - Allows you to specify or select a URL for the schema. It also keeps a history of the last used schemas. The URL must point to the schema file that can be loaded from the local disk or from a remote server through HTTP(S), FTP(S) or a custom protocol (on page 2444). You can specify the URL by using the text field, the history drop-down, the **Insert Editor Variables** (on page 323) button, or the browsing actions in the **Browse** drop-down list.

- **Schema type** - Select a possible schema type from this combo box that is populated based on the extension of the schema file that was entered in the **URL** field. The possible schema types are: XML Schema, DTD, Relax NG, Relax NG Compact, Schematron, or NVDL.

- **Embedded Schematron rules** - If you have selected XML Schema or Relax NG schemas with embedded Schematron rules and you want to use those embedded rules, select this option.

- **Extensions** - Opens a dialog box that allows you to specify Java extension JARs (on page 3220) to be used during the validation.

- **Public ID** - Allows you to specify a public ID if you have selected a DTD.

- **Schematron phase** - If you select a Schematron schema, this drop-down list allows you to select a Schematron phase that you want to use for validation. The listed phases are defined in the Schematron document.

**Configure Validation Engine Dialog Box**

This dialog box allows you to configure options for checking the DITA map document and all referenced topics and maps (similar to the process done when using the **Validate and Check for Completeness** action (on page 2941) in the DITA Maps Manager).

**Note:**

The options presented in the **Configure validation engine** dialog box depends on type of validation engine. For example, when configuring the **DITA-OT Project Validation and Completeness Check** validation engine, the dialog box has slightly fewer options (omitting those that are not applicable).
The **Configure Validation Engine** dialog box contains the following options:

**Batch validate referenced DITA resources**

This option specifies the level of validation that applies to referenced DITA files:

- If the checkbox is left unchecked (default setting), the DITA files will be validated using the rules defined in the DTD or XML Schema declared in the document.
- If the checkbox is selected, the DITA files will be validated using rules defined in their associated validation scenario (on page 787).
Check the existence of non-DITA references resources

Extends the validation of referenced resources to non-DITA files.

**Include remote resources**

Select this option if you want to check that remote referenced binary resources (such as images, movie clips, ZIP archives) exist at the specified location.

**Use DITAVAL filters**

The content of the map is filtered by applying a profiling condition set before validation. You can choose between the following options:

- **From the current condition set** - The map is filtered using the condition set currently applied in the DITA Maps Manager view (on page 2897). Clicking the Details icon opens a topic in the Oxygen XML Editor User Guide that explains how to create a profiling condition set.

- **From all available condition sets** - For each available condition set, the map content is filtered using that set before validation.

- **From the associated transformation scenario** - The filtering condition set is specified explicitly as a DITAVAL file in the current transformation scenario associated with the DITA map.

- **Other DITAVAL files** - For each DITAVAL file from this list, the map content is filtered using the DITAVAL file before validation. Use the Add or Remove buttons to configure the list. The Add button opens a dialog box that allows you to select a local or remote path to a DITAVAL file. You can specify the path by using the text field, its history drop-down, the Insert Editor Variables (on page 323) button, or the browsing actions in the Browse drop-down list.

**Report references to resources outside of the DITA map folder**

If selected, it will report any references to DITA resources that are located outside the main DITA map (on page 3223) folder.

**Report links to topics not referenced in DITA maps**

Checks that all the topics referenced by other topics are also linked in the DITA map.
Report multiple references to the same topic

If selected, it will report warnings when a topic is referenced multiple times in the DITA map, unless a unique @copy-to attribute is used on the <topicref> element for any topic that is referenced multiple times.

For example, it will **not** report a warning if there is a topic referenced twice, but the second <topicref> has a @copy-to attribute set:

```xml
<topicref href="topic.dita"/>
......
<topicref href="topic.dita" copy-to="topic2.dita"/>
```

On the other hand, it **will** report a warning if there is a topic referenced twice and none of the reference-type elements has a @copy-to attribute set or both of them have the @copy-to attribute set to the same value:

```xml
<topicref href="topic.dita" copy-to="topic2.dita"/>
......
<topicref href="topic.dita" copy-to="topic2.dita"/>
```

Check for duplicate topic IDs within the DITA map context

Checks for multiple topics with the same ID in the context of the entire map.

Report duplicate key definitions

Checks the DITA map for multiple key references with the same key defined for them. This is helpful because if you have two different resources with the same value for the @keys attribute, all references will point to the first one encountered and the other will be ignored.

**Note:**

This option takes key scopes (on page 3058) into account.

For example, if you have something like this:

```xml
<topicref href="t2.dita" keys="k2"/>
<topicgroup keyscope="ks">
  <topicref href="t2.dita" keys="k2"/>
</topicgroup>
```

it will not report the "k2" key as a duplicate because it is defined in a key scope (on page 3058) on the second occurrence.

Report unreferenced key definitions
Checks the entire DITA map and reports any key definitions that are not referenced anywhere. Note that if the Use DITAVAL filters option is selected, this check will search for unreferenced key definitions based upon your selected filter.

Report unreferenced reusable elements

Checks the entire DITA map and reports any detected reusable elements that are not referenced anywhere. It looks for elements that have an ID specified in the following types of topic references:

- Any \texttt{<topicref>} that contains a \texttt{@processing-role} attribute set to \texttt{resource-only}.
- Any other referenced topic that contains elements that are reused elsewhere through a \texttt{@conref} or \texttt{@conkeyref}.

Report table layout problems

Looks for table layout problems. The types of errors that may be reported include:

- If a row has fewer cells than the number of columns detected.
- For a CALS table, if a cell has a vertical span greater than the available rows count.
- For a CALS table, if the number of \texttt{<colspecs>} is different than the number of columns detected from the table \texttt{@cols} attribute.
- For a CALS table, if the number of columns detected from the table \texttt{@cols} attribute is different than the number of columns detected in the table structure.
- For a CALS table, if the value of the \texttt{@cols}, \texttt{@rowsep}, or \texttt{@colsep} attributes are not numeric.
- For a CALS table, if the \texttt{@namest}, \texttt{@nameend}, or \texttt{@colname} attributes point to an incorrect column name.

Identify possible conflicts in profile attribute values

When the profiling attributes of a topic contain values that are not found in parent topic profiling attributes, the content of the topic is overshadowed when generating profiled output. This option reports these possible conflicts.

Report attributes and values that conflict with profiling preferences

Looks for profiling attributes and values that are not defined in the Profiling / Conditional Text preferences page (on page 187) (you can click the \texttt{Profiling Preferences} button to open this preferences page). It also checks if profiling attributes defined as \texttt{single-value} have multiple values set in the searched topics.

Additional Schematron checks
Allows you to select a Schematron file that Oxygen XML Editor will use for the validation of DITA resources. You can specify the path by using the text field, its history drop-down, the Insert Editor Variables (on page 323) button, or the browsing actions in the Browse drop-down list.

**Move Up**
Moves the selected validation unit up one spot in the list.

**Move Down**
Moves the selected validation unit down one spot in the list.

**Add**
Adds a new validation unit to the list.

**Remove**
Removes an existing validation unit from the list.

3. Configure any of the existing validation units according to the information above, and you can use the buttons at the bottom of the table to add, remove, or move validation units. Note that if you add a Schematron validation unit, you can also select the validation phase.

4. Click OK.
The newly created validation scenario will now be included in the list of scenarios in the Configure Validation Scenario(s) dialog box. You can select the scenario in this dialog box to associate it with the current document and click the Apply associated button to run the validation scenario.

### Editing a Validation Scenario

To edit an existing validation scenario, follow these steps:

1. Select the Configure Validation Scenario(s) from the Validation toolbar drop-down menu, or from the Document > Validate menu (or the Validate submenu when invoking the contextual menu on a file in the Project view (on page 402)).
The Configure Validation Scenario(s) dialog box is displayed. It contains built-in and user-defined scenarios. The built-in scenarios are organized in categories depending on the type of file they apply to and you can identify them by a yellow key icon that marks them as read-only. The user-defined scenarios are organized under a single category. The default scenarios for the particular framework (on page 3219) are rendered in bold.

Note:

If a main file is associated with the current file, the validation scenarios defined in the main file, along with any Schematron schema defined in the default scenarios for that particular framework, are used for the validation. These take precedence over other types of validation units defined in the default scenarios for the particular framework. For more information on
The top section of the dialog box contains a filter that allows you to search through the scenarios list and the Settings button allows you to configure the following options:

**Show all scenarios**

Select this option to display all the available scenarios, regardless of the document they are associated with.

**Show only the scenarios available for the editor**

Select this option to only display the scenarios that Oxygen XML Editor can apply for the current document type.

**Show associated scenarios**

Select this option to only display the scenarios associated with the document you are editing.

**Import scenarios**

This option opens the Import scenarios dialog box that allows you to select the scenarios file that contains the scenarios you want to import. If one of the scenarios you import is identical to an existing scenario, Oxygen XML Editor ignores it. If a conflict appears (an imported scenario has the same name as an existing one), you can choose between two options:
• Keep or replace the existing scenario.
• Keep both scenarios.

Note:
When you keep both scenarios, Oxygen XML Editor adds imported to the name of the imported scenario.

Export selected scenarios

Use this option to export selected scenarios individually. Oxygen XML Editor creates a scenarios file that contains the exported scenarios. This is useful if you want to share scenarios with others or export them to another computer.

2. Select the scenario and click the Edit button. If you try to edit one of the read-only built-in scenarios, you will receive a warning message that Oxygen XML Editor needs to creates customizable duplicate (you can also use the Duplicate button).

The Edit scenario dialog box is displayed and it lists all the validation units for the scenario.

Figure 226. Edit Validation Scenario

This scenario configuration dialog box allows you to configure the following information and options:

Name
The name of the validation scenario.

Storage
You can choose between storing the scenario in the Project Options (on page 3222) or Global Options (on page 3219).

URL of the file to validate
The URL of the main module that includes the current module. It is also the entry module of the validation process when the current one is validated. To edit the URL, double-click its cell and specify the URL of the main module by doing one of the following:
Enter the URL in the text field or select it from the drop-down list.

- Use the ![Browse](#) drop-down button to browse for a local, remote, or archived file.

- Use the ![Insert Editor Variable](#) button to insert an editor variable (on page 323) or a custom editor variable (on page 332).

**Figure 227. Insert an Editor Variable**

| `{(Desktop)}` - My Desktop
| `{(start-dir)}` - Start directory of custom validator
| `{(standard-params)}` - List of standard parameters for command line
| `{(fn)}` - The current file name without extension
| `{(currentFileURL)}` - The path of the currently edited file (URL)
| `{(fd)}` - The path of current file directory (URL)
| `{(frameworks)}` - Oxygen frameworks directory (URL)
| `{(pd)}` - Project directory (URL)
| `{(oxygen-home)}` - Oxygen installation directory (URL)
| `{(home)}` - The path to user home directory (URL)
| `{(pn)}` - Project name
| `{(env(VAR_NAME))}` - Value of environment variable VAR_NAME
| `{(sys(var.name))}` - Value of system variable var.name

**File type**

The type of the document that is validated in the current validation unit. Oxygen XML Editor automatically selects the file type depending on the value of the **URL of the file to validate** field.

**Validation engine**

You can select one of the engines available in Oxygen XML Editor for validation of the particular document type:

- **Default engine** - The default engine is used to run the validation for the current document type, as specified in the preferences page for that type of document (for example, XSLT preferences page (on page 245), XQuery preferences page (on page 253), XML Schema preferences page (on page 238)).

- **DITA Validation** engine - Performs DITA-specific checks in the context of the specifications (it is similar to the process when using the ![Validate and Check for Completeness](#) action (on page 2941) in the DITA Maps Manager, but for a local file rather than an entire DITA map (on page 3218)).

- **DITA Map Validation and Completeness Check** engine - Performs a validation process that checks the DITA map document and all referenced topics and maps (it is similar to the process when using the ![Validate and Check for Completeness](#) action (on page 2941) in the DITA Maps Manager).
- **DITA-OT Project Validation and Completeness Check** engine - Performs a validation process that checks each context from the provided DITA-OT project file (it is similar to the process when using the Validate and Check for Completeness action (on page 2941) in the DITA Maps Manager).

- **Table Layout Validation** engine - Looks for table layout problems (for more information, see the Report table layout problems option (on page 2946)).

**Automatic validation**

If this option is selected, the validation operation defined by this row is also applied by the automatic validation feature (on page 775). If the Automatic validation feature is disabled in the Document Checking preferences page (on page 226), then this option is ignored, as the preference setting has a higher priority.

**Schema**

This option becomes active when you set the **File type** to **XML Document** and allows you to specify the schema used for the validation unit.

**Settings**

Depending on the selected validation engine, clicking the settings button either opens the Specify Schema dialog box or the Configure validation engine dialog box.

- **Specify Schema Dialog Box**

  This dialog box allows you to specify a custom schema to be used for the validation process.

**Figure 228. Specify Schema Dialog Box**

The Specify Schema dialog box contains the following options:

- **Use detected schema**

  Uses the schema detected for the particular document (on page 817).

- **Use custom schema**
Allows you to specify the schema using the following options:

- **URL** - Allows you to specify or select a URL for the schema. It also keeps a history of the last used schemas. The URL must point to the schema file that can be loaded from the local disk or from a remote server through HTTP(S), FTP(S) or a custom protocol (on page 2444). You can specify the URL by using the text field, the history drop-down, the Insert Editor Variables (on page 323) button, or the browsing actions in the Browse drop-down list.

- **Schema type** - Select a possible schema type from this combo box that is populated based on the extension of the schema file that was entered in the URL field. The possible schema types are: XML Schema, DTD, Relax NG, Relax NG Compact, Schematron, or NVDL.

- **Embedded Schematron rules** - If you have selected XML Schema or Relax NG schemas with embedded Schematron rules and you want to use those embedded rules, select this option.

- **Extensions** - Opens a dialog box that allows you to specify Java extension JARs (on page 3220) to be used during the validation.

- **Public ID** - Allows you to specify a public ID if you have selected a DTD.

- **Schematron phase** - If you select a Schematron schema, this drop-down list allows you to select a Schematron phase that you want to use for validation. The listed phases are defined in the Schematron document.

- **Configure Validation Engine Dialog Box**

This dialog box allows you to configure options for checking the DITA map document and all referenced topics and maps (similar to the process done when using the Validate and Check for Completeness action (on page 2941) in the DITA Maps Manager).

**Note:**
The options presented in the Configure validation engine dialog box depends on type of validation engine. For example, when configuring the DITA-OT Project Validation and Completeness Check validation engine, the dialog box has slightly fewer options (omitting those that are not applicable).
The **Configure Validation Engine** dialog box contains the following options:

**Batch validate referenced DITA resources**

This option specifies the level of validation that applies to referenced DITA files:

- If the checkbox is left unchecked (default setting), the DITA files will be validated using the rules defined in the DTD or XML Schema declared in the document.
- If the checkbox is selected, the DITA files will be validated using rules defined in their associated validation scenario (*on page 787*).
Note:
If you want your DITA topics to be validated against a Relax NG schema, this checkbox must be selected.

Check the existence of non-DITA references resources

Extends the validation of referenced resources to non-DITA files.

Include remote resources

Select this option if you want to check that remote referenced binary resources (such as images, movie clips, ZIP archives) exist at the specified location.

Use DITAVAL filters

The content of the map is filtered by applying a profiling condition set before validation. You can choose between the following options:

- **From the current condition set** - The map is filtered using the condition set currently applied in the DITA Maps Manager view (on page 2897). Clicking the Details icon opens a topic in the Oxygen XML Editor User Guide that explains how to create a profiling condition set.

- **From all available condition sets** - For each available condition set, the map content is filtered using that set before validation.

- **From the associated transformation scenario** - The filtering condition set is specified explicitly as a DITAVAL file in the current transformation scenario associated with the DITA map.

- **Other DITAVAL files** - For each DITAVAL file from this list, the map content is filtered using the DITAVAL file before validation. Use the Add or Remove buttons to configure the list. The Add button opens a dialog box that allows you to select a local or remote path to a DITAVAL file. You can specify the path by using the text field, its history drop-down, the Insert Editor Variables (on page 323) button, or the browsing actions in the Browse drop-down list.

Report references to resources outside of the DITA map folder

If selected, it will report any references to DITA resources that are located outside the main DITA map (on page 3223) folder.

Report links to topics not referenced in DITA maps

Checks that all the topics referenced by other topics are also linked in the DITA map.
Report multiple references to the same topic

If selected, it will report warnings when a topic is referenced multiple times in the DITA map, unless a unique @copy-to attribute is used on the <topicref> element for any topic that is referenced multiple times.

For example, it will not report a warning if there is a topic referenced twice, but the second <topicref> has a @copy-to attribute set:

```xml
<topicref href="topic.dita" />
.....
<topicref href="topic.dita" copy-to="topic2.dita" />
```

On the other hand, it will report a warning if there is a topic referenced twice and none of the reference-type elements has a @copy-to attribute set or both of them have the @copy-to attribute set to the same value:

```xml
<topicref href="topic.dita" copy-to="topic2.dita" />
.....
<topicref href="topic.dita" copy-to="topic2.dita" />
```

Check for duplicate topic IDs within the DITA map context

Checks for multiple topics with the same ID in the context of the entire map.

Report duplicate key definitions

Checks the DITA map for multiple key references with the same key defined for them. This is helpful because if you have two different resources with the same value for the @keys attribute, all references will point to the first one encountered and the other will be ignored.

Note:

This option takes key scopes (on page 3058) into account. For example, if you have something like this:

```xml
<topicref href="t2.dita" keys="k2"/>
<topicgroup keyscope="ks">
  <topicref href="t2.dita" keys="k2"/>
</topicgroup>
```

it will not report the "k2" key as a duplicate because it is defined in a key scope (on page 3058) on the second occurrence.

Report unreferenced key definitions
Checks the entire DITA map and reports any key definitions that are not referenced anywhere. Note that if the Use DITAVAL filters option is selected, this check will search for unreferenced key definitions based upon your selected filter.

**Report unreferenced reusable elements**

Checks the entire DITA map and reports any detected reusable elements that are not referenced anywhere. It looks for elements that have an ID specified in the following types of topic references:

- Any `<topicref>` that contains a `@processing-role` attribute set to resource-only.
- Any other referenced topic that contains elements that are reused elsewhere through a `@conref` or `@conkeyref`.

**Report table layout problems**

Looks for table layout problems. The types of errors that may be reported include:

- If a row has fewer cells than the number of columns detected.
- For a CALS table, if a cell has a vertical span greater than the available rows count.
- For a CALS table, if the number of `<colspecs>` is different than the number of columns detected from the table `@cols` attribute.
- For a CALS table, if the number of columns detected from the table `@cols` attribute is different than the number of columns detected in the table structure.
- For a CALS table, if the value of the `@cols`, `@rowsep`, or `@colsep` attributes are not numeric.
- For a CALS table, if the `@namest`, `@nameend`, or `@colname` attributes point to an incorrect column name.

**Identify possible conflicts in profile attribute values**

When the profiling attributes of a topic contain values that are not found in parent topic profiling attributes, the content of the topic is overshadowed when generating profiled output. This option reports these possible conflicts.

**Report attributes and values that conflict with profiling preferences**

Looks for profiling attributes and values that are not defined in the Profiling / Conditional Text preferences page (on page 187) (you can click the \[ Profiling Preferences \] button to open this preferences page). It also checks if profiling attributes defined as single-value have multiple values set in the searched topics.

**Additional Schematron checks**
Allows you to select a Schematron file that Oxygen XML Editor will use for the validation of DITA resources. You can specify the path by using the text field, its history drop-down, the Insert Editor Variables (on page 323) button, or the browsing actions in the Browse drop-down list.

Move Up
Moves the selected validation unit up one spot in the list.

Move Down
Moves the selected validation unit down one spot in the list.

Add
Adds a new validation unit to the list.

Remove
Removes an existing validation unit from the list.

3. Configure any of the existing validation units according to the information above, and you can use the buttons at the bottom of the table to add, remove, or move validation units. Note that if you add a Schematron validation unit, you can also select the validation phase.

4. When you are done configuring the scenario, click OK. The modified validation scenario will now be included in the list of scenarios in the Configure Validation Scenario(s) dialog box. If you chose to duplicate an existing one, the modified scenario will be listed with the word copy at the end of its name.

Sharing Validation Scenarios

The validation scenarios and their settings can be shared with other users by saving them at project level (on page 3222) or by exporting them to a specialized scenarios file (on page 323) that can then be imported. When you create a new validation scenario or edit an existing one, there is a Storage option to control whether the scenarios are stored in Project Options (on page 3222) or Global Options (on page 3219).

Selecting Project Options (on page 3222) stores the scenario in the project file and can be shared with other users that have access to the project. If your project is saved on a source versioning/sharing system (CVS, SVN, Source Safe, etc.) then your team will have access to the scenarios that you define. When you create a scenario at the project level, the URLs from the scenario become relative to the project URL.

Selecting Global Options (on page 3219) stores the scenario in the global options that are stored in the user home directory.

You can also change the storage options on existing validation scenarios by using the Change storage action from the contextual menu of the list of scenarios.
Resolving References to Remote Schemas with an XML Catalog

When a reference to a remote schema must be used in the validated XML document for interoperability purposes, but a local copy of the schema should actually be used for performance reasons, the reference can be resolved to the local copy of the schema with an XML Catalog (on page 3225).

For example, if the XML document contains a reference to a remote schema docbook.rng like this:

```xml
<?xml-model href="http://www.oasis-open.org/docbook/xml/5.0/rng/docbook.rng" type="application/xml" schematypens="http://relaxng.org/ns/structure/1.0"?>
```

it can be resolved to a local copy with a catalog entry like this:

```xml
<uri name="http://www.oasis-open.org/docbook/xml/5.0/rng/docbook.rng"
     url="rng/docbook.rng"/>
```

An XML Catalog can also be used to map an XML Schema specified with a URN in the @xsi:schemaLocation attribute of an XML document to a local copy of the schema. For example, if the XML document specifies the schema with:

```xml
<topic xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
```

the URN can be resolved to a local schema file with a catalog entry like this:

```xml
<uri name="urn:oasis:names:tc:dita:xsd:topic.xsd:1.1"
     url="topic.xsd"/>
```

Validation Example - A DocBook Validation Error

In the following DocBook 4 document, the content of the <listitem> element does not match the rules of the DocBook 4 schema (docbookx.dtd).

```xml
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE article PUBLIC "-//OASIS//DTD DocBook XML V4.4//EN"
   "http://www.docbook.org/xml/4.4/docbookx.dtd">
<article>
  <title>Article Title</title>
  <sect1>
    <title>Section1 Title</title>
    <itemizedlist>
```

Related Information:
- Sharing Application Settings (on page 312)
- Related Information: Working with XML Catalogs (on page 826)
The Validate Document action will return the following error:

```
<listitem>
  <link>a link here</link>
</listitem>
</itemizedlist>
</sect1>
</article>
```

The **Validate Document** action will return the following error:

```
<listitem>
  <link>a link here</link>
</listitem>
</itemizedlist>
</sect1>
</article>
```

This error message is a little more difficult to understand, so understanding of the syntax or processing rules for the DocBook XML DTD `<listitem>` element is recommended. However, the error message does offer a clue as to the source of the problem, indicating that "The content of element type must match".

Fortunately, most standards-based DTDs, XML Schemas, and Relax NG schemas are supplied with reference documentation. This enables you to read about the element. In this case, you should learn about the child elements of `<listitem>` and their nesting rules. Once you have correctly inserted the required child element and nested it in accordance with the XML rules, the document will become valid.

### Embedding Schematron Rules in XML Schema or RELAX NG

Schematron rules can be embedded into an XML Schema through annotations (using the `<appinfo>` element), or in any element on any level of a RELAX NG Schema (taking into account that the RELAX NG validator ignores all elements that are not in the RELAX NG namespace).

Oxygen XML Editor supports Schematron validation schemas and it is able to extract and use the embedded rules.

### Validating XML Documents with XML Schema and Embedded Schematron

To validate an XML document with XML Schema and its embedded Schematron, you can associate the document like this:

```
<?xml-model href="percent.xsd" type="application/xml"
  schematypens="http://purl.oclc.org/dsdl/schematron"?>
```
Validating XML Documents with Relax NG and Embedded Schematron

To validate an XML document with RELAX NG schema and its embedded Schematron rules, you need to associate the document with both schemas like this:

```xml
<?xml-model href="percent.rng" type="application/xml"
    schematypens="http://relaxng.org/ns/structure/1.0"?>
<?xml-model href="percent.rng" type="application/xml"
    schematypens="http://purl.oclc.org/dsdl/schematron"?>
```

The second association validates your document with Schematron rules extracted from the RELAX NG Schema.

**Note:**

When you work with XML Schema or Relax NG documents that have embedded Schematron rules Oxygen XML Editor provides two built-in validation scenarios: Validate XML Schema with embedded Schematron for XML schema, and Validate Relax NG with embedded Schematron for Relax NG. You can use one of these scenarios to validate the embedded Schematron rules.

Example: Embedded Schematron in XML Schema

```xml
<xsd:appinfo>
    <sch:pattern>
        <sch:rule context="...">
            <sch:assert test="...">Message.</sch:assert>
        </sch:rule>
    </sch:pattern>
</xsd:appinfo>
```

Example: Embedded Schematron in Relax NG Schema

```xml
<grammar
    xmlns="http://relaxng.org/ns/structure/1.0"
    xmlns:ch="http://purl.oclc.org/dsdl/schematron" >
    <sch:pattern context="...">
        <sch:assert test="...">Message.</sch:assert>
    </sch:rule>
</sch:pattern>
<start>
    ..............
</start>
</grammar>
```
Ignoring/Unignoring Validation Problems

If the Enable support for ignoring validation problems option (on page 229) is selected in the Ignored Validation Problems preferences page (on page 229), validation problems can be ignored using quick fix actions that are automatically added to the list of proposals for fixing the problem. These quick fix actions are available for validation problems that have an ID in the following places in the interface:

- When the cursor is placed at the location of the problem in the main editor pane, a Quick Fix icon (💡) is displayed in the stripe on the left side of the editor. Clicking that icon opens a pop-up window where the quick fix proposals are presented.
- When you hover over the problem in the editor, the proposals are presented in a tooltip.
- When you right-click a problem in the Results view (on page 548), the proposals are available in the contextual menu.

Note:
For Schematron-based validation, these quick fix actions for ignoring problems are available for validation problems that have an ID set on the Schematron <assert> or <report> elements. That ID, prefixed with the name of the Schematron, is used as the error code (e.g. flowers.sch:xrefFormatID).

The quick fix actions for ignoring problems are:

**Ignore this problem in this document**
When validating this document, this problem will be ignored based on the message, error code, and file location.

**Ignore this type of problem in this document**
When validating this document, this type of problem will be ignored based on the error code and file location.

**Ignore this type of problem in all documents**
When validating any document, this type of problem will be ignored based on the error code.

Figure 230. Schematron Ignored Problems Quick Fix Proposals

![Figure 230. Schematron Ignored Problems Quick Fix Proposals](image)

When a validation problem is ignored:
• It is added to the Ignored Problems Table (on page 229) in the Ignored Validation Problems preferences page (on page 229). The table displays all the problems that you have specified to be ignored and you can remove items from the table by selecting them and clicking the Delete button located under the table.

• It will no longer be highlighted in the editor pane.

• In the vertical range ruler on the right side of the editor, the problem will be marked with a gray color.

• In the Results view (on page 548), the problem will be marked with a disabled (grayed out) action.

• The ignored problem will also be marked when a batch validation operation is executed.

How to Unignore Validation Problems

Validation problems that have previously been ignored (hence, they are added to the Ignored Problems Table (on page 229)) can be unignored (removed from the table) using the Remove from ignored problems list action from the problem's contextual menu in the Results view (on page 548).

Another way to unignore a problem that has been ignored is to hover over its gray marker in the vertical ruler on the right side of the editor until a tooltip is displayed, then use F2 to change the focus to the tooltip, then click the Remove from ignored problems list quick fix link. This results in that problem being removed from the list and the problem is no longer ignored.

XML Quick Fixes

The Oxygen XML Editor Quick Fix support (on page 3223) helps you resolve errors that appear in an XML document by offering Quick Fixes to problems such as missing required attributes or invalid elements. Quick Fixes are available in Text mode and Author mode.

To activate this feature, hover over or place the cursor in the highlighted area of text where a validation error or warning occurs. If a Quick Fix is available for that particular error or warning, you can access the Quick Fix proposals with any of the following methods:

• When hovering over the error or warning, the proposals may be presented in a tooltip pop-up window and the available quick Quick Fixes include a link that can be used to perform the fix.

Figure 231. Quick Fix Presented in a Tooltip in Text Mode
• When hovering over the error or warning in Author mode, a small Quick Fix drop-down menu is presented. You can use the drop-down menu to display a list of available Quick Fixes to select from for the particular error or warning.

Figure 232. Quick Fix Presented in a Tooltip in Author Mode

[Image of a tooltip with Quick Fix options]

• If you place the cursor in the highlighted area where a validation error or warning occurs, a Quick Fix icon (💡) is displayed in the stripe on the left side of the editor. If you click this icon, Oxygen XML Editor displays the list of available fixes.

Figure 233. Quick Fix Drop-Down Menu in Author Mode

[Image of a drop-down menu with Quick Fix options]

• With the cursor placed in the highlighted area of the error or warning, you can also invoke the Quick Fix menu by pressing Alt + 1 (Command + Alt + 1 on macOS) on your keyboard.

Whenever you make a modification in the XML document or you apply a fix, the list of Quick Fixes is recomputed to ensure that you always have valid proposals.

Note:
A Quick Fix that adds an element inserts it along with required and optional elements, and required and fixed attributes, depending on how the Content Completion preferences (on page 211) are configured.

Quick Fixes for DTD, XSD, and Relax NG Errors

Oxygen XML Editor offers Quick Fixes (on page 3223) for common errors that appear in XML documents that are validated against DTD, XSD, or Relax NG schemas.
Note:
For XML documents validated against XSD schemas, the Quick Fixes are only available if you use the default Xerces validation engine.

Quick Fixes are available in Text mode and Author mode.

Oxygen XML Editor provides Quick Fixes for numerous types of problems, including the following:

<table>
<thead>
<tr>
<th>Problem Type</th>
<th>Available Quick Fixes</th>
</tr>
</thead>
<tbody>
<tr>
<td>A specific element is required in the current context</td>
<td>Insert the required element</td>
</tr>
<tr>
<td>An element is invalid in the current context</td>
<td>Remove the invalid element</td>
</tr>
<tr>
<td>The content of the element should be empty</td>
<td>Remove the element content</td>
</tr>
<tr>
<td>An element is not allowed to have child elements</td>
<td>Remove all child elements</td>
</tr>
<tr>
<td>Text is not allowed in the current element</td>
<td>Remove the text content</td>
</tr>
<tr>
<td>A required attribute is missing</td>
<td>Insert the required attribute</td>
</tr>
<tr>
<td>An attribute is not allowed to be set for the current element</td>
<td>Remove the attribute</td>
</tr>
<tr>
<td>The attribute value is invalid</td>
<td>Propose the correct attribute values</td>
</tr>
<tr>
<td>ID value is already defined</td>
<td>Generate a unique ID value</td>
</tr>
<tr>
<td>References to an invalid ID</td>
<td>Change the reference to an already defined ID</td>
</tr>
</tbody>
</table>

Related Information:
Schematron Quick Fixes (SQF) (on page 815)
Ignoring/Unignoring Validation Problems (on page 812)

Schematron Quick Fixes (SQF)

Oxygen XML Editor provides support for Schematron Quick Fixes (on page 3223) (SQF). They help you resolve issues that appear in XML documents that are validated against Schematron schemas by offering you solution proposals. The Schematron Quick Fixes are an extension of the Schematron language and they allow you to define fixes for Schematron validation messages. Specifically, they are associated with assert or report messages.

A typical use case is using Schematron Quick Fixes to assist content authors with common editing tasks. For example, you can use Schematron rules to automatically report certain validation warnings (or errors) when performing regular editing tasks, such as inserting specific elements or changing IDs to...
match specific naming conventions. For more details and examples, see the following blog post: https://blog.oxygenxml.com/topics/SchematronBCs.html.

**Displaying the Schematron Quick Fix Proposals**

The defined Schematron *Quick Fixes* are displayed on validation errors in *Text* mode and *Author* mode.

![Figure 235. Example of a Schematron Quick Fix](image)

**Related Information:**
- Editing Schematron Quick Fixes *(on page 1223)*
- Schematron Quick Fix Specifications
- Presenting Schematron Validation Issues *(on page 1206)*
- Examples of Schematron Rules and Quick Fixes *(on page 1194)*
- Ignoring/Unignoring Validation Problems *(on page 812)*

**Associating a Schema to XML Documents**

To provide as-you-type validation and to compute valid proposals for the *Content Completion Assistant*(on page 3217), Oxygen XML Editor requires a schema to be associated with the XML document. The schema specifies how the internal structure of the XML is defined.

**Supported Types of Schema**

The following schema types are supported:

- **W3C XML Schema 1.0 and 1.1** (with and without embedded Schematron rules) - The association with an XML Schema is added as an attribute of the root element with one of the following:
  - `@xsi:schemaLocation` attribute, if the root element of the document is in the namespace.
  - `@xsi:noNamespaceSchemaLocation` attribute, if the root element is not in the namespace.
- **DTD** - The association with a DTD is added as a `DOCTYPE` declaration.
- **Relax NG - XML Syntax** (with and without embedded Schematron rules) - The association is added as an `xml-model processing instruction`.
- **Relax NG - Compact Syntax** - The association is added as an `xml-model processing instruction`.
- **NVDL** - The association is added as an `xml-model processing instruction`.
- **Schematron** (both ISO Schematron and Schematron 1.5) - The association is added as an `xml-model processing instruction`.
Detecting the Schema(s) for Validation

For validation, Oxygen XML Editor tries to detect one or more schemas by searching multiple locations, in the following order:

1. The schema or multiple schemas referenced in validation stages from the validation scenario(s) (on page 818) associated with the current XML document.
2. If no validation scenario is selected to be used with the current XML document, then it falls back to the schema or multiple schemas defined in validation stages from the validation scenarios specified as default in the particular document type configuration (on page 821).
3. If a schema is still not detected, then it falls back to the schema or multiple schemas associated directly in the XML document (on page 823).

Tip:
To quickly open the schema used for validating the current document, select the Open Associated Schema action from the toolbar (or Document > Schema menu).

4. If a schema is still not detected, then it falls back to the schema defined in the Schema tab of a framework (document type) configuration (on page 825).

Detecting a Schema for Content Completion

For content completion, Oxygen XML Editor uses just one schema and tries to detect that schema by searching multiple locations, in the following order:

1. If no schema is detected in the document, then it falls back to the highest ranking schema defined in validation stages from the validation scenario(s) associated with the current document (on page 818).
2. If a schema is still not detecting, then it falls back to the highest ranking schema defined in validation stages from validation scenarios specified as default in the particular document type configuration (on page 821).
3. Oxygen XML Editor determines the most appropriate or highest ranking schema that is associated directly in the XML document (on page 823) and uses it for content completion.
4. If a schema is still not detecting, then it falls back to the schema defined in the Schema tab of a framework (document type) configuration (on page 825).

Related Information:
Modular Contextual XML Editing Using 'Main Files' Support (on page 829)
W3C: Associating Schemas with XML Documents
Associating a Schema Through a Validation Scenario

Oxygen XML Editor uses the rules defined in the detected schema to report errors and warnings during automatic and manual validations that help maintain the structural integrity of your XML documents. You can specify the schema to be used for validation directly in validation scenarios (on page 787) and there are several methods that can be used to do so.

Configure a Validation Scenario and Specify the Schema

To associate a schema to a validation scenario to be used whenever the scenario is invoked, follow these steps:

1. Select the Configure Validation Scenario(s) from the Validation toolbar drop-down menu, or from the Document > Validate menu (or the Validate submenu when invoking the contextual menu on a file in the Project view (on page 402)).
2. Click the New button to create a new validation scenario or the Edit button to modify an existing one.
3. Add or configure validation units (on page 800) according to your needs and click the Specify Schema button.

Step Result: The Specify Schema dialog box is displayed:

![Specify Schema Dialog Box](image)

The Specify Schema dialog box contains the following options:

- **Use detected schema**
  
  Uses the schema detected for the particular document (on page 817).

- **Use custom schema**
  
  Allows you to specify the schema using the following options:
- **URL** - Allows you to specify or select a URL for the schema. It also keeps a history of the last used schemas. The URL must point to the schema file that can be loaded from the local disk or from a remote server through HTTP(S), FTP(S) or a custom protocol (on page 2444). You can specify the URL by using the text field, the history drop-down, the `Insert Editor Variables (on page 323)` button, or the browsing actions in the `Browse` drop-down list.
- **Schema type** - Select a possible schema type from this combo box that is populated based on the extension of the schema file that was entered in the URL field. The possible schema types are: XML Schema, DTD, Relax NG, Relax NG Compact, Schematron, or NVDL.
- **Embedded Schematron rules** - If you have selected XML Schema or Relax NG schemas with embedded Schematron rules and you want to use those embedded rules, select this option.
- **Public ID** - Allows you to specify a public ID if you have selected a DTD.
- **Extensions** - Opens a dialog box that allows you to specify Java extension JARs (on page 3220) to be used during the validation.
- **Schematron phase** - If you select a Schematron schema, this drop-down list allows you to select a Schematron phase that you want to use for validation. The listed phases are defined in the Schematron document.

4. Select the schema to be associated with the validation unit and configure the rest of the options according to your preferences.
5. Click **OK** on both dialog boxes.

**Result:** The schema is now associated with that validation scenario whenever it is invoked.

**Use the Validate with Action to Specify a Schema for Validating the Current Document**

To validate the current document using a specified schema, follow these steps:

1. Select the **Validation with** action from the `Validation` drop-down menu on the toolbar (or `Document > Validate` menu).

   **Step Result:** The **Validate with** dialog box is displayed:

   ![Figure 237. Validate with Dialog Box](image)
This dialog box contains the following options:

- **URL** - Allows you to specify or select a URL for the schema. It also keeps a history of the last used schemas. The URL must point to the schema file that can be loaded from the local disk or from a remote server through HTTP(S), FTP(S) or a custom protocol (on page 2444). You can specify the URL by using the text field, the history drop-down, the 📦 Insert Editor Variables (on page 323) button, or the browsing actions in the ⌉ Browse drop-down list.

- **Schema type** - Select a possible schema type from this combo box that is populated based on the extension of the schema file that was entered in the URL field. The possible schema types are: XML Schema, DTD, Relax NG, Relax NG Compact, Schematron, or NVDL.

- **Embedded Schematron rules** - If you have selected XML Schema or Relax NG schemas with embedded Schematron rules and you want to use those embedded rules, select this option.

- **Public ID** - Allows you to specify a public ID if you have selected a DTD.

- **Extensions** - Opens a dialog box that allows you to specify Java extension JARs (on page 3220) to be used during the validation.

- **Schematron phase** - If you select a Schematron schema, this drop-down list allows you to select a Schematron phase that you want to use for validation. The listed phases are defined in the Schematron document.

2. Select the schema to be associated with the manual validation and configure the rest of the options according to your preferences.

3. Click **OK**.

**Result:** The current document is validated using the schema you specified.

**Tip:**
To quickly open the schema used for validating the current document, select the [Open Associated Schema](#) action from the toolbar (or **Document > Schema** menu).

**Use the Validate with Schema Action to Specify a Schema for Validating all Selected Documents**

To validate multiple documents using a specified schema, follow these steps:

1. Select all the documents you want to validate in the Project view.

2. Invoke the contextual menu (right-click) and select the **Validate with Schema** action from the **Validate** submenu.

   **Step Result:** The **Validate with** dialog box is displayed:
This dialog box contains the following options:

- **URL** - Allows you to specify or select a URL for the schema. It also keeps a history of the last used schemas. The URL must point to the schema file that can be loaded from the local disk or from a remote server through HTTP(S), FTP(S) or a custom protocol (on page 2444). You can specify the URL by using the text field, the history drop-down, the ![Insert Editor Variables](on page 323) button, or the browsing actions in the ![Browse](drop-down list).

- **Schema type** - Select a possible schema type from this combo box that is populated based on the extension of the schema file that was entered in the **URL** field. The possible schema types are: XML Schema, DTD, Relax NG, Relax NG Compact, Schematron, or NVDL.

- **Embedded Schematron rules** - If you have selected XML Schema or Relax NG schemas with embedded Schematron rules and you want to use those embedded rules, select this option.

- **Public ID** - Allows you to specify a public ID if you have selected a DTD.

- **Extensions** - Opens a dialog box that allows you to specify Java extension JARs (on page 3220) to be used during the validation.

- **Schematron phase** - If you select a Schematron schema, this drop-down list allows you to select a Schematron phase that you want to use for validation. The listed phases are defined in the Schematron document.

3. Select the schema that you want to use to validate all selected documents and configure the rest of the options according to your preferences.

4. Click **OK**.

**Result:** The selected documents are validated using the schema you specified.

### Associating a Schema in Validation Scenarios Defined in the Document Type

To report errors and warnings during automatic and manual validations that help maintain the structural integrity of particular XML document types, Oxygen XML Editor uses rules defined in the schema that is detected in the validation scenarios that are associated to each particular document type.

To associate a schema in validation scenarios defined in the framework (on page 3219) (document type) configuration, follow these steps:
1. Open the Preferences dialog box (Options > Preferences) (on page 124) and go to Document Type Association.

2. Select your particular document type and click the Edit or Duplicate button to modify an existing framework (or use the New button to create a new one).

   **Step Result:** This opens a Document type configuration dialog box (on page 140).

3. Go to the Validation tab (on page 166).

4. Create or edit a validation scenario:
   a. To create a new validation scenario (on page 788), click the + New button.
   b. To edit the properties of an existing validation scenario (on page 798), select it and click the ✎ Edit button (you can also use the Duplicate button to copy an existing scenario and edit its properties).

5. Add or configure validation units (on page 800) according to your needs and click the Specify Schema button.

   **Step Result:** The Specify Schema dialog box is displayed:

   **Figure 239. Specify Schema Dialog Box**

   ![Specify Schema Dialog Box]

   The Specify Schema dialog box contains the following options:

   **Use detected schema**

   Uses the schema detected for the particular document (on page 817).

   **Use custom schema**

   Allows you to specify the schema using the following options:
   - **URL** - Allows you to specify or select a URL for the schema. It also keeps a history of the last used schemas. The URL must point to the schema file that can be loaded from the local disk or from a remote server through HTTP(S), FTP(S) or a custom protocol (on page 2444). You can specify the URL by using the text field,
the history drop-down, the ✒️ Insert Editor Variables (on page 323) button, or the browsing actions in the 📜 Browse drop-down list.

- **Schema type** - Select a possible schema type from this combo box that is populated based on the extension of the schema file that was entered in the URL field. The possible schema types are: XML Schema, DTD, Relax NG, Relax NG Compact, Schematron, or NVDL.

- **Embedded Schematron rules** - If you have selected XML Schema or Relax NG schemas with embedded Schematron rules and you want to use those embedded rules, select this option.

- **Public ID** - Allows you to specify a public ID if you have selected a DTD.

- **Extensions** - Opens a dialog box that allows you to specify Java extension JARs (on page 3220) to be used during the validation.

- **Schematron phase** - If you select a Schematron schema, this drop-down list allows you to select a Schematron phase that you want to use for validation. The listed phases are defined in the Schematron document.

6. Select the schema to be associated with the validation unit and configure the rest of the options according to your preferences.

7. Click OK on both dialog boxes.

**Result:** The schema is now associated with the validation scenario you just configured for that particular document type.

### Associating a Schema Directly in XML Documents

The schema used by the **Content Completion Assistant (on page 3217)** and document validation engine can be directly associated with the current document by using the ✒️ Associate Schema action. For most of the schema types, it uses the `xml-model` processing instruction, with the exceptions of:

- **W3C XML Schema** - The `@xsi:schemaLocation` attribute or `@xsi:noNamespaceSchemaLocation` attribute is used.
- **DTD** - The `DOCTYPE` declaration is used.

The association can specify a relative file path or a URL of the schema. The advantage of relative file path is that you can configure the schema at file level instead of framework (on page 3219) level.

To associate a schema to the current document, follow these steps:

1. Select the ✒️ Associate Schema action from the toolbar (or Document > Schema menu).

**Step Result:** The Associate Schema dialog box is displayed:
Figure 240. Associate Schema Dialog Box

This dialog box contains the following options:

- **URL** - Allows you to specify or select a URL for the schema. It also keeps a history of the last used schemas. The URL must point to the schema file that can be loaded from the local disk or from a remote server through HTTP(S), FTP(S) or a custom protocol (on page 2444).

- **Use path relative to file location** - Select this option if the XML instance document and the associated schema contain relative paths. The location of the schema file is inserted in the XML instance document as a relative file path. This practice allows you, for example, to share these documents with other users without running into problems caused by multiple project locations on physical disk.

- **Schema type** - Select a possible schema type from this combo box that is populated based on the extension of the schema file that was entered in the **URL** field. The possible schema types are: XML Schema, DTD, Relax NG, Relax NG Compact, Schematron, or NVDL.

- **Add additional association for embedded Schematron rules** - If you have selected XML Schema or Relax NG schemas with embedded Schematron rules and you want to use those embedded rules, select this option.

- **Public ID** - Allows you to specify a public ID if you have selected a DTD.

- **Keep existing schema associations** - Select this option to use the existing schema associations of the currently edited document.

2. Select the schema that will be associated with the XML document and configure the rest of the options according to your preferences.

3. Click **OK**.

**Result:** The schema association is created based upon the specified type.
• **XML Schema** - The association with an XML Schema is added as an attribute of the root element with one of the following:
  ◦ `@xsi:schemaLocation` attribute, if the root element of the document is in the namespace.
  ◦ `@xsi:noNamespaceSchemaLocation` attribute, if the root element is not in the namespace.
• **DTD** - The association with a DTD is added as a `DOCTYPE` declaration.
• **Other** - The association with a Relax NG, Schematron, or NVDL schema is added as an `xml-model` processing instruction.

**Tip:**
To quickly open the schema used for validating the current document, select the Open Associated Schema action from the toolbar (or Document > Schema menu).

**Related Information:**
Validating XML Documents ([on page 772])
Content Completion Assistant in Text Mode ([on page 532])
Content Completion Assistant in Author Mode ([on page 616])

### Associating a Schema in a Framework (Document Type) Configuration

The schema used to compute valid proposals in the [Content Completion Assistant ([on page 3217])](#) and by the document validation engine to report errors and warnings can be defined in each particular framework ([on page 3219]) (document type). This schema will be used only if one is not detected in the current XML file ([on page 823]).

To associate a schema in a particular framework (document type), follow these steps:

1. Open the Preferences dialog box ([Options > Preferences](#)) and go to Document Type Association.
2. Select your particular document type and click the Edit ([on page 138]), Extend ([on page 138]), or Duplicate ([on page 138]) button to modify an existing framework (or use the New button to create a new one).

  **Step Result:** This opens a Document type configuration dialog box ([on page 140]).
3. Go to the Schema tab ([on page 144]).
4. Select the schema type and its URI.
5. Click OK.

**Result:** The schema is now associated with the particular document type and will be used by the Content Completion Assistant and validation engine if a schema is not detected in the current XML document.
Learn Document Structure when Schema is not Detected

When working with documents that do not specify a schema, or the schema is not known or does not exist, Oxygen XML Editor is able to learn and translate the document structure to a DTD. You can choose to save the learned structure to a file to provide a DTD as an initialization source for content completion (on page 532) and document validation (on page 772). This feature is also useful for producing DTDs for documents that contain personal or custom element types.

When you open a document that is not associated with a schema, Oxygen XML Editor automatically learns the document structure and uses it for content completion (on page 532). To disable this feature, deselect the Learn on open document option in the user preferences (on page 211).

Related Information:
- Detecting a Schema (on page 816)

Create a DTD from Learn Document Structure Option

When there is no schema associated with an XML document, Oxygen XML Editor can learn the document structure by parsing the document internally. This feature is enabled by the Learn on open document option (on page 211) that is available in the user preferences.

To create a DTD from the learned structure, follow these steps:

1. Open the XML document that will be used to create the DTD.
2. Go to Document > XML Document > Learn Structure (Ctrl + Shift + L (Command + Shift + L on macOS)).
   The Learn Structure action reads the mark-up structure of the current document. The Learn completed message is displayed in the application status bar when the action is finished.
3. Go to Document > XML Document > Save Structure (Ctrl + Shift + S (Command + Shift + S on macOS)) and enter the DTD file path.
4. Click the Save button.

Working with XML Catalogs

Oxygen XML Editor uses XML Catalogs (on page 3225) to resolve references for validations and transformations and they are especially helpful for resolving external resources when internet access is not available or your connection is slow.

Oxygen XML Editor supports any XML Catalog file that conforms to one of the following:

1. OASIS XML Catalogs Committee Specification v1.1.
2. OASIS Technical Resolution 9401:1997, including the plain-text flavor described in that resolution.

The version 1.1 of the OASIS XML Catalog specification introduces the possibility to map a system ID, public ID, or a URI to a local copy using only a suffix of the ID or URI used in the actual document. This is done using the catalog elements systemSuffix and uriSuffix.
Depending on the resource type, Oxygen XML Editor uses different catalog mappings.

**Table 9. Catalog Mappings**

<table>
<thead>
<tr>
<th>Doc Type</th>
<th>Referenced Resource</th>
<th>Mappings</th>
</tr>
</thead>
<tbody>
<tr>
<td>XML</td>
<td>DTD</td>
<td>system or public</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The <strong>Prefer</strong> option (on page 235) controls which one of the mappings should be used.</td>
</tr>
<tr>
<td>XML Schema</td>
<td></td>
<td>The following strategy is used (if one step fails to provide a resource, the next is applied):</td>
</tr>
<tr>
<td>Relax NG</td>
<td>Schematron</td>
<td>1. Resolve the schema using <strong>URI</strong> catalog mappings.</td>
</tr>
<tr>
<td></td>
<td>NVDL</td>
<td>2. Resolve the schema using <strong>system</strong> catalog mappings. This happens only if the <strong>Resolve schema locations also through system mappings</strong> option (on page 235) is selected (it is by default).</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3. Resolve the root <strong>namespace</strong> using <strong>URI</strong> catalog mappings.</td>
</tr>
<tr>
<td>XSL</td>
<td>XSL/ANY</td>
<td><strong>URI</strong></td>
</tr>
<tr>
<td>CSS</td>
<td>CSS</td>
<td><strong>URI</strong></td>
</tr>
<tr>
<td>JSON</td>
<td>JSON</td>
<td><strong>URI</strong></td>
</tr>
<tr>
<td>XML Schema</td>
<td>XML Schema</td>
<td>The following strategy is used (if one step fails to provide a resource, the next is applied):</td>
</tr>
<tr>
<td>Relax NG</td>
<td>Relax NG</td>
<td>1. Resolve schema reference using <strong>URI</strong> catalog mappings.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2. Resolve schema reference using <strong>system</strong> catalog mappings. This happens only if the <strong>Resolve schema locations also through system mappings</strong> option (on page 235) is selected (it is by default).</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3. Resolve schema <strong>namespace</strong> using <strong>URI</strong> catalog mappings. This happens only if the <strong>Process namespaces through URI mappings for XML Schema</strong> option (on page 236) is selected (it is not by default).</td>
</tr>
</tbody>
</table>

**Creating an XML Catalog with a Template**

An **XML Catalog** (on page 3225) file can be created quickly in Oxygen XML Editor starting from the document template called **OASIS XML Catalog**. It is available when creating new document templates (on page 368).

**How Oxygen XML Editor Determines which Catalog to Use**

Oxygen XML Editor uses **XML Catalogs** (on page 3225) to resolve references for validations and transformations and it maps such references to the built-in local copies of the schemas associated with the various frameworks (on page 3219) (DocBook, DITA, TEI, XHTML, SVG, etc.)
Oxygen XML Editor 24.1 | 8 - Editing Documents | 828

Oxygen XML Editor includes default global catalogs and default catalogs for each of the built-in frameworks, and you can also create your own.

Oxygen XML Editor looks for catalogs in the following order:

- Global user-defined catalogs that are set in the XML Catalog preferences page (on page 235).
- User-defined catalogs that are set for each framework (on page 3219) in the Catalog tab (on page 164) of the Document Type configuration dialog box (on page 140).
- Default built-in catalogs.

**Example: Using an XML Catalog to map an Absolute XML Schema Reference to an XML Schema Located Relative to the XML Catalog**

An XML Catalog can be used to map an XML Schema specified with a URN in the @xsi:noNamespaceSchemaLocation attribute of an XML document to a local copy of the schema.

Considering the following XML document code snippet:

```xml
<topic xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
```

The URN can be resolved to a local schema file with an XML catalog entry like this:

```xml
<uri name="urn:oasis:names:tc:dita:xsd:topic.xsd:1.1"
     url="topic.xsd"/>
```

**Example: Using an XML Catalog to map an Imported XML Schema Reference to an XML Schema Located Relative to the XML Catalog**

An XML Catalog can be used to map an xs:import or xs:include XML Schema reference to a local copy of the schema.

Considering the following xs:include inside an XML Schema:

```xml
<xs:include schemaLocation="someFolder/common.xsd"/>
```

The reference can be resolved to a local schema file with an XML catalog entry like this:

```xml
<uriSuffix uriSuffix="someFolder/common.xsd" url="relative/path/to/common.xsd"/>
```

Related Information:

XML Catalog Preferences (on page 235)

**Resolving Schema Locations Through XML Catalogs**

Schema locations can be mapped using an XML Catalog (on page 3225). Oxygen XML Editor resolves the location of a schema in the following order:
• First, it attempts to resolve the schema location as a URI (uri, uriSuffix, rewriteUri, delegateUri mappings from the XML Catalog). If this succeeds, the process end here.

• If the **Resolve schema locations also through system mappings** option (on page 235) is selected in the XML Catalog preferences page, it attempts to resolve the schema location as a system ID (system, systemSuffix, rewriteSuffix, rewriteSystem from the XML Catalog). If this succeeds, the process ends here.

• If the **Process "schemaLocation" namespaces through URI mappings for XML Schema** option (on page 236) is selected in the XML Catalog preferences page, the target namespace of the imported XML Schema is resolved through URI mappings. If the schema specified in the schemaLocation attribute is not resolved successfully, the namespace of the root element is taken into account. If this succeeds, the process ends here.

• If none of these succeeds, the actual schema location (on page 817) is used.

---

### Related Information:

[Working with XML Catalogs (on page 826)]

---

### Modular Contextual XML Editing Using 'Main Files' Support

Smaller interrelated modules that define a complex XML modular structure cannot be correctly edited or validated individually, due to their interdependency with other modules. Oxygen XML Editor provides the support for defining the main module (or modules), allowing you to edit any file from the hierarchy in the context of the [main files (on page 3220)].

You can set a main XML document either using the main files support from the Project view (on page 418), or using a validation scenario.

To set a **main file** using a validation scenario, add validation units that point to the main modules. Oxygen XML Editor warns you if the current module is not part of the dependencies graph computed for the main XML document. In this case, it considers the current module as the main XML document.

The advantages of working with modular XML files in the context of a **main file (on page 3220)** include:

• Correct validation of a module in the context of a larger XML structure.

• **Content Completion Assistant (on page 3217)** displays all collected entities and IDs starting from the main files.

• Oxygen XML Editor uses the schema defined in the **main file** when you edit a module that is included in the hierarchy through the **External Entity** mechanism.

• The **main files** defined for the current module determines the scope of the search and refactoring actions (on page 831) for ID/IDREFS values and for updating references when renaming/moving a resource. Oxygen XML Editor performs the search and refactoring actions in the context that the **main files** determine, improving the speed of execution.

For more information about editing modular XML files in the **main files** context, watch our video demonstration:
Oxygen XML Editor allows you to search for ID declarations and references (IDREFS) and to define the scope of the search and refactor operations (on page 831). These operations are available for XML documents that have an associated DTD, XML Schema, or Relax NG schema. These operations are available through the search and refactor actions in the contextual menu. In Text mode, these actions are also available in the Quick Assist (on page 566) menu.

The search and refactor actions from the contextual menu are grouped in the Manage IDs section:

- **Rename in**
  Renames the ID and all its occurrences. Selecting this action opens the Rename XML ID dialog box. This dialog box lets you insert the new ID value and choose the scope of the rename operation (on page 831). For a preview of the changes you are about to make, click Preview. This opens the Preview dialog box, which presents a list with the files that contain changes and a preview zone of these changes.

- **Rename in File**
  Renames the ID you are editing and all its occurrences from the current file.

  **Note:**
  Available in the Text mode only.

- **Search References**
  Searches for the references of the ID. By default, the scope of this action is the current project. If you configure a scope using the Select the scope for the Search and Refactor operations (on page 831) dialog box, this scope will be used instead.

- **Search References in**
  Searches for the references of the ID. Selecting this action opens the Select the scope for the Search and Refactor operations (on page 831).

- **Search Declarations**
  Searches for the declaration of the ID reference. By default, the scope of this action is the current project. If you configure a scope using the Select the scope for the Search and Refactor operations (on page 831) dialog box, this scope will be used instead.
Search Declarations in

Searches for the declaration of the ID reference. Selecting this action opens the Select the scope for the Search and Refactor operations (on page 831).

Note:
Available in the Text mode only.

Search Occurrences in file

Searches for the declaration and references of the ID in the current document.

Tip:
A quick way to go to the declaration of an ID in Text mode is to move the cursor over an ID reference and use the Ctrl + Single-Click (Command + Single-Click on macOS) navigation.

Selecting an ID that you use for search or refactor operations differs between the Text and Author modes. In the Text mode, you position the cursor inside the declaration or reference of an ID. In the Author mode, Oxygen XML Editor collects all the IDs by analyzing each element from the path to the root. If more IDs are available, you are prompted to choose one of them.

Figure 241. Selecting an ID in the Author Mode

Related Information:
Modular Contextual XML Editing Using 'Main Files' Support (on page 829)

Search and Refactor Operations Scope

The scope is a collection of documents that define the context of a search and refactor operation. To control it you can use the Change scope operation, available in the Quick Assist action set or on the Resource Hierarchy/Dependency View toolbar. You can restrict the scope to the current project or to one or multiple
working sets (on page 3224). The Use only Main Files, if enabled checkbox allows you to restrict the scope of the search and refactor operations to the resources from the Main Files directory. Click read more for details about the Main Files support (on page 418).

Figure 242. Change Scope Dialog Box

The scope you define is applied to all future search and refactor operations until you modify it. Contextual menu actions allow you to add or delete files, folders, and other resources to the working set (on page 3224) structure.

XML Resource Hierarchy/Dependencies View

The Resource Hierarchy/Dependencies view displays the hierarchy or dependencies for resources included in an XML document. The tree structure presented in this view is built based on the Xinclude and External Entity mechanisms. If the view is not displayed, it can be opened by selecting it from the Window > Show View menu.

If you want to see the hierarchy or dependencies of an XML document, select the document in the Project view (on page 402) and choose Resource Hierarchy or Resource Dependencies from the contextual menu.
The following actions are available on the toolbar of the **Resource Hierarchy/Dependencies** view:

- **Refresh**
  Refreshes the Hierarchy/Dependencies structure.

- **Stop**
  Stops the hierarchy/dependencies computing.

- **Show Hierarchy**
  Allows you to choose a resource to compute the hierarchy structure.

- **Show Dependencies**
  Allows you to choose a resource to compute the dependencies structure.

- **Configure dependencies search scope**
  Allows you to configure a scope to compute the dependencies structure. There is also an option for automatically using the defined scope for future operations.

- **History**
  Provides access to the list of previously computed dependencies. Use the **Clear history** button to remove all items from this list.

The contextual menu for a resource listed in the **Resource Hierarchy/Dependencies** view contains the following actions:
Open

Opens the resource. You can also double-click a resource in the Hierarchy/Dependencies structure to open it.

Go to reference

Opens the source document where the resource is referenced.

Copy location

Copies the location of the resource.

Move resource

Moves the selected resource.

Rename resource

Renames the selected resource.

Resource Hierarchy

Shows the hierarchy for the selected resource.

Resource Dependencies

Shows the dependencies for the selected resource.

Add to Main Files

Adds the currently selected resource in the Main Files directory (on page 418).

Expand More

Expands more of the children of the selected resource from the Hierarchy/Dependencies structure.

Collapse All

Collapses all children of the selected resource from the Hierarchy/Dependencies structure.

Tip:

When a recursive reference is encountered in the Hierarchy view, the reference is marked with a special icon.

Note:

The Move resource or Rename resource actions give you the option to update the references to the resource (on page 835). Only the references made through the XInclude and External Entity mechanisms are handled.

Related Information:

Modular Contextual XML Editing Using 'Main Files' Support (on page 829)
Search and Refactor Operations Scope (on page 831)
Moving/Renaming XML Resources

When you select the Rename action in the contextual menu of the Resource/Hierarchy Dependencies view, the Rename resource dialog box is displayed. The following fields are available:

- **New name** - Presents the current name of the edited resource and allows you to modify it.
- **Update references of the renamed resource(s)** - Select this option to update the references to the resource you are renaming. A Preview option is available that allows you to see what will be updated before selecting Rename to process the operation.

When you select the Move action from the contextual menu of the Resource/Hierarchy Dependencies view, the Move resource dialog box is displayed. The following fields are available:

- **Destination** - Presents the path to the current location of the resource you want to move and gives you the option to introduce a new location.
- **New name** - Presents the current name of the moved resource and gives you the option to change it.
- **Update references of the moved resource(s)** - Select this option to update the references to the resource you are moving, in accordance with the new location and name. A Preview option is available that allows you to see what will be updated before selecting Move to process the operation.

Combining XML Content Using DTD Entities and XInclude

When documenting large projects, it is likely that there are multiple people involved. In this case, it is usually more efficient to using a modular approach so that everyone involved in the project can work in parallel. Other advantages of modular documentation include: reusable content possibilities, smaller file units are easier to edit, and better version control.

Two possible solutions for this are to use **DTD Entities** or **XInclude** to combine XML content in a main file (on page 3220). A main document can be created with references to various document parts, users can edit those documents individually, and then apply an XSLT stylesheet over the main document to obtain the output files in various formats (for example, PDF or HTML).

Combining XML Document Content Using DTD Entities

There are two conditions for including a document fragment using DTD entities:

- The main document should declare the DTD to be used, while the external entities should declare the XML fragments to be referenced.
- The referenced documents that contain the fragments cannot also define the DTD because the main document will not be valid. If you want to validate the parts separately you have to use XInclude (on page 837) for assembling the parts together with the main file.

The main document looks like this:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE book SYSTEM "../xml/docbookx.dtd" [...
```
<!ENTITY testing SYSTEM "testing.xml" > ]>
<book>
  <chapter> ...
</chapter>
</book>

The referenced document (*testing.xml*) looks like this:

```xml
<section> ... here is the section content ... </section>
```

**Note:**

The indicated DTD and the element names (*section*, *chapter*) are used here only for illustrating the inclusion mechanism. You can use any DTD and element names you need.

The content from the referenced file (in the example above, it is a `<section>` in the *test.xml* file) can be inserted somewhere in the main document:

```xml
... &testing; ...
```

To obtain output in various formats (for example, PDF or HTML), you simply need to apply an XSLT stylesheet over the main document using a transformation scenario.

**Viewing the Expanded Content in Oxygen XML Editor**

When a transformation scenario is applied on the *main file*, an intermediary file combines all the referenced content and it will be expanded in the final output. If you want to see how the referenced content will be expanded before applying the transformation, you can do one of the following:

- Simply switch to **Author** mode.
- Create a minimal XSLT stylesheet that simply copies the XML content, then create a new XSLT transformation scenario that applies the stylesheet over the XML. The XSLT stylesheet would look like this:

```xml
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    xmlns:xs="http://www.w3.org/2001/XMLSchema"
    xmlns:math="http://www.w3.org/2005/xpath-functions/math"
    exclude-result-prefixes="xs math"
    version="3.0">
  <xsl:template match="node() | @*">
    <xsl:copy>
      <xsl:apply-templates select="node() | @*"/>
    </xsl:copy>
  </xsl:template>
</xsl:stylesheet>
```
Combining XML Documents and Fragments Using XInclude

XInclude is a standard for assembling XML instances into another XML document through inclusion. A main file (on page 3220) can be dynamically created from smaller XML documents without having to physically duplicate the content of the smaller files. The advantage of using XInclude instead of the DTD Entities method (on page 835) is that each of the assembled documents is permitted to contain a Document Type Declaration (DOCTYPE). This means that each file is a valid XML instance and can be independently validated. It also means that the main document, which includes smaller instances, can be validated without having to remove or comment out the DOCTYPE (as is the case with External Entities).

Enabling XInclude Support in Oxygen XML Editor

The XInclude support in Oxygen XML Editor is enabled by default. It is controlled by the Enable XInclude processing option (on page 237) in the XML > XML Parser preferences page (on page 237). When enabled, Oxygen XML Editor will be able to validate and transform documents comprised of parts added using XInclude.

Example: Using XInclude to Combine Files

A chapter file (introduction.xml) looks like this:

```xml
<?xml version="1.0"?>
<chapter>
  <title>Getting started</title>
  <section>
    <title>Section title</title>
    <para>Para text</para>
  </section>
</chapter>
```

The main article (main file) looks like this:

```xml
<?xml version="1.0"?>
<!DOCTYPE article PUBLIC "-//OASIS//DTD DocBook XML V4.3//EN" "http://www.docbook.org/xml/4.3/docbookx.dtd"
[ <!ENTITY % xinclude SYSTEM "../frameworks/docbook/ndt/xinclude.mod">]
%xinclude;
>
<article>
  <title>Install guide</title>
  <para>This is the install guide.</para>
  <xi:include xmlns:xi="http://www.w3.org/2001/XInclude" href="introduction.xml">
    <xi:fallback>
```

```xml
```
In this example, note the following:

- The DOCTYPE declaration defines an entity that references a file containing the information to add the \texttt{xi} namespace to certain elements defined by the DocBook DTD.
- The \texttt{href} attribute of the \texttt{xi:include} element specifies that the \texttt{introduction.xml} file will replace the \texttt{xi:include} element when the document is parsed.
- If the \texttt{introduction.xml} file cannot be found, the parser will use the value of the \texttt{xi:fallback} element - a \texttt{FIXME} message.

**Example: Using XInclude to Combine Fragments of Files**

If you want to include only a fragment of a file in the \texttt{main file (on page 3220)}, the fragment must be contained in a tag having an \texttt{@xml:id} attribute and you must use an XPointer expression pointing to the \texttt{@xml:id} value.

\begin{itemize}
  \item Oxygen XML Editor supports the XPointer Framework and the XPointer element() Scheme, but it does NOT support the XPointer xpointer() Scheme.
\end{itemize}

For example, if the \texttt{main file} is:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<?xml-model href="test.rng" type="application/xml"
  schematypens="http://relaxng.org/ns/structure/1.0"?>
<test>
  <xi:include href="a.xml" xpointer="a1"
    xmlns:xi="http://www.w3.org/2001/XInclude"/>
</test>
```

and the file (\texttt{a.xml}) is:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<test>
  <a xml:id="a1">test</a>
</test>
```

after resolving the XPointer reference, the document is:
Viewing the Expanded Content in Oxygen XML Editor

When a transformation scenario is applied on the main file, an intermediary file combines all the referenced content and it will be expanded in the final output. If you want to see how the referenced content will be expanded before applying the transformation, you can do one of the following:

- Simply switch to **Author** mode.
- Create a minimal XSLT stylesheet that simply copies the XML content, then create a new XSLT transformation scenario that applies the stylesheet over the XML. The XSLT stylesheet would look like this:

```xml
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    xmlns:xs="http://www.w3.org/2001/XMLSchema"
    xmlns:math="http://www.w3.org/2005/xpath-functions/math"
    exclude-result-prefixes="xs math"
    version="3.0">
  <xsl:template match="node() | @*">
    <xsl:copy>
      <xsl:apply-templates select="node() | @*"/>
    </xsl:copy>
  </xsl:template>
</xsl:stylesheet>
```

XInclude 1.1 Features

Oxygen XML Editor offers partial support for XInclude 1.1 features. This includes support for fragment identifiers and attribute copying.

- **Fragment Identifiers**
  
  You can use `<xi:include>` to reference a text file and specify the `@fragid` value so that you only get part of that text file in the main document. For some examples and to see how the `<xi:include>` gets expanded when the `@fragid` specifies a line range or character range, see **Textual Inclusion Examples with RFC5147 Fragment Identifiers**.

- **Attribute Copying**
Any namespaced attribute defined on the `<xi:include>` element will be passed to the root element of the included content.

For example, if you have this:

```xml
<xi:include href="section2.xml" xmlns:xi="http://www.w3.org/2001/XInclude"
   xml:id="sectInner1"/>
```

and `section2.xml` looks like this:

```xml
<sect2 xmlns="http://docbook.org/ns/docbook" version="5.0"
   xmlns:xlink="http://www.w3.org/1999/xlink" xml:id="section2">
   <title>FS2</title>
   <para>P2</para>
</sect2>
```

then the final processed result will have the original `xml:id="section2"` replaced with the value specified in the `xi:included` section.

For more information, see Attribute Copying when Processing XML. Also, to see more examples, see Attribute Copying Examples.

Related information

W3C Specifications: XML Inclusions (XInclude) Version 1.1

Refactoring XML Documents

In the life cycle of XML documents there are instances when the XML structure needs to be changed to accommodate various needs. For example, when an associated schema is updated, an attribute may have been removed, or a new element added to the structure.

These types of situations cannot be resolved with a traditional Find/Replace tool, even if the tool accepts regular expressions. The problem becomes even more complicated if an XML document is computed or referenced from multiple modules, since multiple resources need to be changed.

To assist you with these types of refactoring tasks, Oxygen XML Editor includes a specialized XML Refactoring tool that helps you manage the structure of your XML documents.

XML Refactoring Tool

The XML Refactoring tool is presented in the form of an easy to use wizard that is designed to reduce the time and effort required to perform various structure management tasks. For example, you can insert, delete, or rename an attribute in all instances of a particular element that is found in all documents within your project.

To access the tool, select the XML Refactoring action from one of the following locations:
• The Tools menu.
• The Refactoring submenu from the contextual menu in the Project view (on page 402).
• The Refactoring submenu from the contextual menu in the DITA Maps Manager view (on page 2897).

**Note:**
The built-in refactoring operations are also available from the Refactoring submenu in the contextual menu of Author or Text mode. This is useful because by selecting the operations from the contextual menu, Oxygen XML Editor considers the editing context to skip directly to the wizard page of the appropriate operation and to help you by preconfiguring some of the parameter values. For your convenience, the last 5 operations that were finished (on page 843) or previewed (on page 843) also appear in the Refactoring submenu of the contextual menu in the Project view and the DITA Maps Manager.

**XML Refactoring Wizard**

The XML Refactoring tool includes the following wizard pages:

**Refactoring operations**

The first wizard page presents the available operations, grouped by category. To search for an operation, you can use the filter text box at the top of the page.

*Figure 244. XML Refactoring Wizard*
Configure Operation Parameters

The next wizard page allows you to specify the parameters for the refactoring operation. The parameters are specific to the type of refactoring operation that is being performed. For example, to delete an attribute you need to specify the parent element and the qualified name of the attribute to be removed.

Figure 245. XML Refactoring 2nd Wizard Page (Delete Attribute Operation)

Scope and Filters

The last wizard page allows you to select the set of files that represent the input of the operation.

Figure 246. XML Refactoring - Scope and Filters Wizard Page
Scope section

In the Scope section, you can select from predefined resource sets (such as the current file, your whole project, the current DITA map (on page 3218) hierarchy for DITA projects, etc.) or you can define your own set of resources by creating a working set (on page 3224).

Filters

The Filters section includes the following options:

- **Include files** - Allows you to filter the selected resources by using a file pattern. For example, to restrict the operation to only analyze build files you could use `build*.xml` for the file pattern.
- **Restrict only to known XML file types** - When selected, only resources with a known XML file type will be affected by the operation.
- **Look inside archives** - When selected, the resources inside archives will also be affected.

Preview

You can use the Preview button to open a comparison panel where you can review all the changes that will be made by the refactoring operation before applying the changes.

Finish

After clicking the Finish button, the operation will be processed and Oxygen XML Editor provides no automatic means for reverting the operations. Any Undo action will only revert changes on the current document.

Troubleshooting:

If an operation fails, a notification will be displayed in the Results panel with some information about the error. For example, if the operation was invoked on a read-only resource, the error will indicate that a read-only file cannot be converted.

Tip:

If an operation takes longer than expected you can use the Stop button in the progress bar to cancel the operation.

Restriction:

XML refactoring operations cannot preserve CDATA sections. If your document contains XML CDATA sections, the refactoring operations will convert them to plain text nodes.
Built-in Refactoring Operations

The XML Refactoring tool includes a variety of built-in operations that can be used for common refactoring tasks. They are grouped by category in the Refactoring operations wizard page. You can also access the operations from the Refactoring submenu in the contextual menu of Author or Text mode. The operations are also grouped by category in this submenu. When selecting the operations from the contextual menu, Oxygen XML Editor considers the editing context to get the names and namespaces of the current element or attribute, and uses this information to preconfigure some of the parameter values for the selected refactoring operation.

Tip:
Each operation includes a link in the lower part of the wizard that opens the XML / XSLT-XQuery / XPath preferences page where you can configure XPath options and declare namespace prefixes.

The following built-in operations are available:

Refactoring Operations for Attributes

Add/Change attribute

Use this operation to change the value of an attribute or insert a new one. This operation allows you to specify the following parameters:

Parent element section

Element
The parent element of the attribute to be changed, in the form of a local name from any namespace, a local name with a namespace prefix, or an XPath expression.

Attribute section

Local name
The local name of the affected attribute.

Namespace
The namespace of the affected attribute.

Value
The value for the affected attribute.

Options section

You can choose between one of the following options for the Operation mode:

Add the attribute in the parent elements where it is missing
Adds the attribute to all instances of the specified parent element.
Change the value in the parent elements where the attribute already exists

Replaces the value of the already existing attribute in all instance of the specified parent element.

Both

Adds the attributes to the instances where it is missing and replaces the value in instances where the attribute already exists.

Convert attribute to element

Use this operation to convert a specified attribute to an element. This operation allows you to specify the following parameters:

Parent element section

Element

The parent element of the attribute to be converted, in the form of a local name from any namespace, a local name with a namespace prefix, or an XPath expression.

Attribute section

Local name

The local name of the affected attribute.

Namespace

The namespace of the affected attribute.

New element section

Local name

The local name of the new element.

Namespace

The namespace of the new element.

Delete attribute

Use this operation to remove one or more attributes. This operation requires you to specify the following parameters:

Element

The parent element of the attribute to be deleted, in the form of a local name from any namespace, a local name with a namespace prefix, or an XPath expression.

Attribute

The name of the attribute to be deleted.

Rename attribute
Use this operation to rename an attribute. This operation requires you to specify the following parameters:

**Element**
The parent element of the attribute to be renamed, in the form of a local name from any namespace, a local name with a namespace prefix, or an XPath expression.

**Attribute**
The name of the attribute to be renamed.

**New local name**
The new local name of the attribute.

**Replace in attribute value**
Use this operation to search for a text fragment inside an attribute value and change the fragment to a new value. This operation allows you to specify the following parameters:

**Target attribute section**

**Element**
The parent element of the attribute to be modified, in the form of a local name from any namespace, a local name with a namespace prefix, or an XPath expression.

**Attribute**
The name of the attribute to be modified.

**Find / Replace section**

**Find**
The text fragments to find. You can use Perl-like regular expressions.

**Replace with**
The text fragment to replace the target with. This parameter can bind regular expression capturing groups ($1, $2, etc.) from the find pattern.

**Refactoring Operations for Comments**

**Delete comments**
Use this operation to delete comments from one or more elements. This operation requires you specify the following parameter:

**Element**
The target element (or elements) that will have comments deleted, in the form of a local name from any namespace, a local name with a namespace prefix, or an XPath expression.

**Note:**
Comments that are outside the root element will not be deleted because the serializer preserves the content before and after the root.

**Refactoring Operations for DITA Topics**

**Change topic ID to file name**

Use this operation to change the ID of a topic to be the same as its file name.

**Convert CALS tables to simple tables**

Use this operation to convert DITA CALS tables to simple tables.

**Convert DITA 1.3 Maps and Topics to DITA 2.0**

Use this operation to convert topics and maps that adhere to the DITA 1.3 standard to the DITA 2.0 standard.

- Changes DOCTYPE declarations and XML Schema/Relax NG schema references.
- DITA Map changes:
  - Removes the `@lockmeta` attribute.
  - Removes the `<topicset>` and `<topicsetref>` elements.
  - Removes the `<anchor>` and `<anchorref>` elements and the `@anchorref` attribute.
  - Migrates the `@navtitle` attribute as a `<navtitle>` element.
  - Migrates the `@title` attribute as a `<title>` element.
  - Converts the `@copy-to` attribute to a `<resourceid>` element.
  - Replaces the `@print` attribute with an `@deliveryTarget` attribute.
  - Convert topicmeta `linktext` to `linktitle`.
  - Removed `<hasInstance>, <hasKind>, <hasNarrower>, <hasPart>, <hasRelated>, and <relatedSubjects>` from subject scheme relationship tables in subject scheme, including `<subjectRelTable>, <subjectRelHeader>, <subjectRel>, and <subjectRole>`.
- DITA task changes:
  - Converts the `<substep>` element to a `<step>` element.
  - Converts the `<substeps>` element to a `<steps>` element.
- DITA topic changes:
  - Removes the `@type` attribute with the value `fastpath`.
  - Converts the `@alt` attribute to an `<alt>` element.
  - Replaces the `<index-sort-as>` element with a `<sort-as>` element.
  - Removes the `<itemgroup>` element.
  - Moves the contents of the `<titleAlts>` element inside the `<prolog>`.
  - Removes the `@domains` attribute.
- Renames `<sectiondiv>` to `<div>`.
- Remove `@query` attribute from `<link>` element.
- Remove `@specentry` attribute from `<stentry>` element.
- Remove the `@spectitle` attribute.

Convert conrefs to conkeyrefs

Use this operation to convert `@conref` attributes to `@conkeyref` attributes. For more information and instructions for using this operation, see Converting Conrefs to Conkeyrefs (on page 3041).

Convert Nested Topics to New Topics (Available from the contextual menu of editable maps/nodes in the DITA Maps Manager (on page 2897))

Use this operation on topics that contain nested `<topic>` elements to convert each nested topic to a new topic. Also, the new topics are added in the DITA Maps Manager as the first child topics of the original topic.

Convert Sections to New Topics (Available from the contextual menu of editable maps/nodes in the DITA Maps Manager (on page 2897))

Use this operation on topics that contain multiple sections to convert each section to a new topic. Also, the new topics are added in the DITA Maps Manager as the first child topics of the original topic.

Convert simple tables to CALS tables

Use this operation to convert DITA simple tables to CALS tables.

Convert to Concept

Use this operation to convert a DITA topic (of any type) to a DITA Concept topic type (for example, Topic to Concept). For more information, see Converting DITA Topics to Another Type (on page 2970).

Convert to General Task

Use this operation to convert a DITA topic (of any type) to a DITA General Task topic type (for example, Task to General Task). For more information, see Converting DITA Topics to Another Type (on page 2970).

Convert to Reference

Use this operation to convert a DITA topic (of any type) to a DITA Reference topic type (for example, Topic to Reference). For more information, see Converting DITA Topics to Another Type (on page 2970).

Convert to Task

Use this operation to convert a DITA topic (of any type) to a DITA Task topic type (for example, Topic to Task). For more information, see Converting DITA Topics to Another Type (on page 2970).

Convert to Topic
Use this operation to convert a DITA topic (of any type) to a DITA Topic (for example, Task to Topic). For more information, see Converting DITA Topics to Another Type (on page 2970).

**Convert to Troubleshooting**

Use this operation to convert a DITA topic (of any type) to a DITA Troubleshooting topic type (for example, Topic to Troubleshooting). For more information, see Converting DITA Topics to Another Type (on page 2970).

**Rename Key**

Use this operation to rename a key. It also updates all references to it.

**Note:**

It does not work on DITA 1.3 key scopes.

**Generate IDs**

Use this operation to automatically generate unique IDs for elements.

**Scope and Filters:**

All of the DITA refactoring actions allow you to choose a scope for the operation and some filters:

**Scope**

Select from a variety of options to define the scope that will have resources affected by the operation. For example, you can choose to affect all resources in the Project, All opened files, Current DITA map hierarchy, or just the Current file.

**Filters section**

**Include files**

Specifies files to be excluded from the operation. You can specify multiple files by separating them with commas and the patterns can include wildcards (such as * or ?).

**Restrict to known XML file types only**

Excludes non-XML file types from the operation.

**Look inside archives**

If this option is selected, the scope of the operation will include files inside archives.

**Refactoring Operations for DITA Maps**

**Convert DITA Bookmap to Map**

Convert a DITA bookmap to a DITA map.

**Convert DITA Map to Bookmap**
Convert a DITA map to a DITA bookmap.

**Change or remove profiling attribute value**

Change or remove a value from a DITA profiling attribute. A profiling attribute can have multiple values, separated by spaces (e.g. for `platform="windows redhat"`, you can change the current `redhat` value to `linux`). Select the name of the profiling attribute, the current value to replace, and the new value. If the new value is left empty, the current value is removed from the profiling attribute.

**Define keys for all topic references**

This refactoring action is useful for converting links inside a DITA project from direct to indirect key-based addressing. When applied on DITA resources from your project (DITA maps and topics), this refactoring action defines keys for all of a DITA map's topic references based on the referenced file name and converts each direct reference to a key reference in each DITA topic. If a topic references already has keys defined, the action does not define new ones. Inside the DITA topics, whenever there is a link element (`<xref>` or `<link>`) with a direct reference to another DITA topic or an element with a `@conref`, the action attempts to convert them to indirect key-based addressing. The refactoring action may introduce linking errors or create duplicate keys so it is advised to run the Validate and check for completeness action from the DITA Maps Manager toolbar to manually fix those problems. You can enable the Report duplicate keys checkbox to also report any keys that are defined more than once.

**Scope and Filters:**

All of the DITA refactoring actions allow you to choose a scope for the operation and some filters:

**Scope**

Select from a variety of options to define the scope that will have resources affected by the operation. For example, you can choose to affect all resources in the Project, All opened files, Current DITA map hierarchy, or just the Current file.

**Filters section**

**Include files**

Specifies files to be excluded from the operation. You can specify multiple files by separating them with commas and the patterns can include wildcards (such as * or ?).

**Restrict to known XML file types only**

Excludes non-XML file types from the operation.

**Look inside archives**

If this option is selected, the scope of the operation will include files inside archives.
Refactoring Operations for *Elements*

**Delete element**

Use this operation to delete elements. This operation requires you to specify the following parameter:

**Element**

The target element to be deleted, in the form of a local name from any namespace, a local name with a namespace prefix, or an XPath expression.

**Delete element content**

Use this operation to delete the content of elements. This operation requires you to specify the following parameter:

**Element**

The target element whose content is to be deleted, in the form of a local name from any namespace, a local name with a namespace prefix, or an XPath expression.

**Insert element**

Use this operation to insert new elements. This operation allows you to specify the following parameters:

**Element section**

- **Local name**
  
  The local name of the element to be inserted.

- **Namespace**
  
  The namespace of the element to be inserted.

**Location section**

- **XPath**

  An XPath expression that identifies an existing element to which the new element is relative, in the form of a local name from any namespace, a local name with a namespace prefix, or other XPath expressions.

- **Position**

  The position where the new element will be inserted, in relation to the specified existing element. The possible selections in the drop-down menu are: **After, Before, First child**, or **Last child**.

**Rename element**
Use this operation to rename elements. This operation requires you to specify the following parameters:

**Target elements (XPath)**

The target elements to be renamed, in the form of a local name from any namespace, a local name with a namespace prefix, or other XPath expressions.

**New local name**

The new local name of the element.

**Unwrap element**

Use this operation to remove the surrounding tags of elements, while keeping the content unchanged. This operation requires you to specify the following parameter:

**Target elements (XPath)**

The target elements whose surrounding tags will be removed, in the form of a local name from any namespace, a local name with a namespace prefix, or other XPath expressions.

**Wrap element**

Use this operation to surround elements with element tags. This operation allows you to specify the following parameters:

**Target elements (XPath)**

The target elements to be surrounded with tags, in the form of a local name from any namespace, a local name with a namespace prefix, or other XPath expressions.

**Wrapper element section**

**Local name**

The local name of the *Wrapper element*.

**Namespace**

The namespace of the *Wrapper element*.

**Wrap element content**

Use this operation to surround the content of elements with element tags. This operation allows you to specify the following parameters:

**Target elements (XPath)**

The target elements whose content will be surrounded with tags, in the form of a local name from any namespace, a local name with a namespace prefix, or other XPath expressions.

**Wrapper element section**
Local name

The local name of the *Wrapper element* that will surround the content of the target.

Namespace

The namespace of the *Wrapper element* that will surround the content of the target.

**Refactoring Operations for Fragments**

**Insert XML fragment**

Use this operation to insert an XML fragment. This operation allows you to specify the following:

**XML Fragment**

The XML fragment to be inserted.

**Location section**

**XPath**

An XPath expression that identifies an existing element to which the inserted fragment is relative, in the form of a local name from any namespace, a local name with a namespace prefix, or other XPath expressions.

**Position**

The position where the fragment will be inserted, in relation to the specified existing element. The possible selections in the drop-down menu are: *After*, *Before*, *First child*, or *Last child*.

**Replace element content with XML fragment**

Use this operation to replace the content of elements with an XML fragment. This operation allows you to specify the following parameters:

**Target elements (XPath)**

The target elements whose content will be replaced, in the form of a local name from any namespace, a local name with a namespace prefix, or other XPath expressions.

**XML Fragment**

The XML fragment with which to replace the content of the target element.

**Replace element with XML fragment**

Use this operation to replace elements with an XML fragment. This operation allows you to specify the following parameters:
Target elements (XPath)
The target elements to be replaced, in the form of a local name from any namespace, a local name with a namespace prefix, or other XPath expressions.

XML Fragment
The XML fragment with which to replace the target element.

Refactoring Operations for JATSKit

Add BITS DOCTYPE - NLM/NCBI Book Interchange 2.0
Use this operation to add an NLM 'BITS' 2.0 DOCTYPE declaration.

Add Blue DOCTYPE - NISO JATS Publishing 1.1
Use this operation to add a JATS 'Blue' 1.1 DOCTYPE declaration.

Normalize IDs
Use this operation to normalize assigned IDs and assigned IDs to elements that are missing them.

All of these JATSKit refactoring actions allow you to choose a scope for the operation and some filters:

Scope
Select from a variety of options to define the scope for the resources that will be affected by the operation. For example, you can choose to affect all resources in the Project, All opened files, or just the Current file.

Filters section

Include files
Specifications files to be excluded from the operation. You can specify multiple files by separating them with commas and the patterns can include wildcards (such as * or ?).

Restrict to known XML file types only
Excludes non-XML file types from the operation.

Look inside archives
If this option is selected, the scope of the operation will include files inside archives.

Refactoring Operations for Processing Instructions

Accept all tracked changes, remove all Oxygen-specific comments and highlights
Use this operation to accept all application-specific tracked changes (from elements and attributes) or remove all application-specific comments or highlights. There are several options to choose from:
**Accept all tracked changes**

Accepts all application-specific tracked changes (from elements and attributes).

**Remove comments**

Removes all application-specific comments.

**Remove highlights**

Removes all application-specific highlights.

**Delete processing instructions**

Use this operation to delete all processing instructions that have a certain target name from the processed documents. This operation requires you to specify the following parameter:

**Processing instruction target**

The target name of the processing instructions to delete.

*Note:*

Processing instructions that are outside the root element are not deleted because the serializer preserves the content before and after the root.

**Refactoring Operations for Publishing Template**

These operations are for those who use Oxygen Publishing Templates for WebHelp Responsive output customization.

**Migrate HTML Page Layout Files to v21**

Use this operation to convert custom HTML page layout files (on page 1622) that are included in a custom Publishing Template that was created in Oxygen XML Editor version 20.0 or 20.1 so that they will be compatible with Oxygen XML Editor version 21.0.

**Migrate HTML Page Layout Files to v22**

Use this operation to convert custom HTML page layout files (on page 1622) that are included in a custom Publishing Template that was created in Oxygen XML Editor versions 20.0 - 21.1 so that they will be compatible with Oxygen XML Editor version 22.0.

**Update HTML Pages**

*Attention:*

This operation is only used by Oxygen XML Editor and should not be used manually.
Additional Notes:

• There are some operations that allow `<ANY>` for the `local name` and `namespace` parameters. This value can be used to select an element or attribute regardless of its local name or namespace. Also, the `<NO_NAMESPACE>` value can be used to select nodes that do not belong to a namespace.

• Some operations have parameters that accept XPath expressions to match elements or attributes. In these XPath expressions you can only use the prefixes declared in the `Options > Preferences > XML > XSLT-XQUERY > XPath (on page 258)` page. This preferences page can be easily opened by clicking the link in the note (Each prefix used in an XPath expression must be declared in the Default prefix-namespace mappings section) at the bottom of the `Configure Operation Parameters` wizard page.

Custom Refactoring Operations

While Oxygen XML Editor includes a variety of built-in XML refactoring operations to help you accomplish particular tasks, you can also create custom operations according to your specific needs. For example, you could create a custom refactoring operation to convert an attribute to an element and insert the element as the first child of the parent element.

An XML Refactoring operation is defined as a pair of resources:

• An `XQuery Update script` or `XSLT stylesheet` that Oxygen XML Editor will run to refactor the XML files.
• An `XML Operation Descriptor` file that contains information about the operation (such as the name, description, and parameters).
All the defined custom operations are loaded by the XML Refactoring Tool and presented in the Refactoring Operations wizard page (on page 841), along with the built-in operations.

After the user chooses an operation and specifies its parameters, Oxygen XML Editor processes an XQuery Update or XSLT transformation over the input file. This transformation is executed in a safe mode, which implies the following:

- When loading the document:
  - The XInclude mechanism is disabled. This means that the resources included by using XInclude will not be visible in the transformation.
  - The DTD entities will be processed without being expanded.
  - The associated DTD will be not loaded, so the default attributes declared in the DTD will not be visible in the transformation.

- When saving the updated XML document:
  - The DOCTYPE will be preserved.

**Note:**

This can be changed using Saxon extension functions in XSLT (on page 873).

- The DTD entities will be preserved as they are in the original document when the document is saved.
The attribute values will be kept in their original form without being normalized. The spaces between attributes are preserved. Basically, the spaces are lost by a regular XML serialization since they are not considered important.

The result of this transformation overrides the initial input file.

**Note:**
To achieve some of the previous goals, the XML Refactoring mechanism adds several attributes that are interpreted internally. The attributes belong to the `http://www.oxygenxml.com/ns/xmlRefactoring/additional_attributes` namespace. These attributes should not be taken into account when processing the input XML document since they are discarded when the transformed document is serialized.

**Restriction:**
Comments or processing instructions that are in any node before or after the root element cannot be modified by an XML Refactoring operation. In other words, XML Refactoring operations can only be applied on the root element and the nodes inside it. However, as a work around to this limitation, you can use Saxon extension functions and the XSLT stylesheet method (on page 873) to implement the new custom XML refactoring operation.

**Creating a Custom Refactoring Operation**

To create a custom refactoring operation, follow these steps:

1. Create an XQuery Update script (on page 863) or XSLT stylesheet file (on page 868).
2. Create an XML Refactoring Operation Descriptor file contains the path to the XQuery Update script (on page 866) or XSLT stylesheet (on page 871).
3. Store both files in one of the locations that Oxygen XML Editor (on page 875) scans when loading the custom operations.

**Result:** Once you run the XML Refactoring tool again, the custom operation appears in the Refactoring Operations wizard page (on page 841).

**Custom Refactoring Script**

The first step in creating a custom refactoring operation is to create an XQuery Update script (on page 863) or XSLT stylesheet (on page 868) that is needed to process the refactoring operations. The easiest way to create this script file is to use the New document wizard to create a new XQuery or XSLT file and you can use
the XQuery method example (on page 863) or XSLT method example (on page 868) to help you with the content.

There are cases when it is necessary to add parameters in the XQuery script (on page 863) or XSLT stylesheet (on page 868). For instance, if you want to rename an element, you may want to declare an external parameter associated with the name of the element to be renamed. To allow you to specify the value for these parameters, they need to be declared in the refactoring operation descriptor file that is associated with this operation.

Note:
The XQuery Update processing is disabled by default in Oxygen XML Editor. Thus, if you want to create or edit an XQuery Update script you have to enable this mechanism by creating an XQuery transformation scenario (on page 1536) and choose Saxon EE as the transformation engine. Also, you need to make sure the Enable XQuery update option is selected in the Saxon processor advanced options (on page 1485).

Note:
If you are using an XSLT file, XPath expressions that are passed as parameters will automatically be rewritten to conform with the mapping of the namespace prefixes declared in the XML /XSLT-XQuery / XPath preferences page (on page 258).

The next step in creating a custom refactoring operation is to create an XML Refactoring Operation Descriptor file contains the path to the XQuery Update script (on page 866) or XSLT stylesheet (on page 871).

Related Information:
XQuery Update Script for Creating a Custom Operation (on page 863)
XSLT Stylesheet for Creating a Custom Operation (on page 868)

Custom Refactoring Operation Descriptor File

The second step in creating a custom refactoring operation is to create an operation descriptor file. The easiest way to do this is to use the New document wizard and choose the XML Refactoring Operation Descriptor template.

Introduction to the Descriptor File

This file contains information (such as name, description, and id) that is necessarily when loading an XML Refactoring operation. It also contains the path to the XQuery Update script (on page 863) or XSLT stylesheet (on page 868) that is associated with the particular operation through the <script> element.

You can specify a category for your custom operations to logically group certain operations. The <category> element is optional and if it is not included in the descriptor file, the default name of the category for the custom operations is Other operations.
The descriptor file is edited and validated against the following schema: `frameworks/xml_refactoring/operation_descriptor.xsd`.

**Declaring Parameters in the Descriptor File**

If the XQuery Update script or XSLT stylesheet includes parameters, they should be declared in the `parameters` section of the descriptor file. All the parameters specified in this section of the descriptor file will be displayed in the XML Refactoring tool within the Configure Operation Parameters wizard page (on page 842) for that particular operation.

The value of the first `<description>` element in the `<parameters>` section will be displayed at the top of the Configure Operation Parameters wizard page (on page 842).

To declare a parameter, specify the following information:

- **label** - This value is displayed in the user interface for the parameter.
- **name** - The parameter name used in the XQuery Update script or XSLT stylesheet and it should be the same as the one declared in the script.
- **type** - Defines the type of the parameter and how it will be rendered. There are several types available:
  - **TEXT** - Generic type used to specify a simple text fragment.
  - **XPATH** - Type of parameter whose value is an XPATH expression. For this type of parameter, Oxygen XML Editor will use a text input with corresponding content completion and syntax highlighting.
  - **NAMESPACE** - Used for editing namespace values.
  - **REG_EXP_FIND** - Used when you want to match a certain text by using Perl-like regular expressions.
  - **REG_EXP_REPLACE** - Used along with `REG_EXP_FIND` to specify the replacement string.

**Note:**

The value of this parameter is transferred as plain text to the XQuery Update or XSLT transformation without being evaluated. You should evaluate the XPath expression inside the XQuery Update script or XSLT stylesheet. For example, you could use the `saxon:evaluate` Saxon extension function.

**Note:**

A relative XPath expression is converted to an absolute XPath expression by adding `//` before it (`//XPathExp`). This conversion is done before transferring the XPath expression to the XML refactoring engine.

**Note:**

When writing XPath expressions, you can only use prefixes declared in the Options > Preferences > XML > XSLT-XQuery > XPath (on page 258) options page.
XML_FRAGMENT - This type is used when you want to specify an XML fragment. For this type, Oxygen XML Editor will display a text area specialized for inserting XML documents.

NC_NAME - The parameter for NC_NAME values. It is useful when you want to specify the local part of a QName (on page 3223) for an element or attribute.

BOOLEAN - Used to edit boolean parameters.

TEXT_CHOICE - It is useful for parameters whose value should be from a list of possible values. Oxygen XML Editor renders each possible value as a radio button option.

- **description** - The description of the parameter. It is used by the application to display a tooltip when you hover over the parameter.

- **possibleValues** - Contains the list with possible values for the parameter and you can specify the default value, as in the following example:

  ```xml
  <possibleValues onlyPossibleValuesAllowed="true">
    <value name="before">Before</value>
    <value name="after" default="true">After</value>
    <value name="firstChild">First child</value>
    <value name="lastChild">Last child</value>
  </possibleValues>
  ```

Specialized Parameters to Match Elements or Attributes

If you want to match elements or attributes, you can use some specialized parameters, in which case Oxygen XML Editor will propose all declared elements or attributes based on the schema associated with the currently edited file. The following specialized parameters are supported:

**elementLocation**

This parameter is used to match elements. For this type of parameter, the application displays a text field where you can enter the element name or an XPath expression. The text from the @label attribute is displayed in the application as the label of the text field. The @name attribute is used to specify the name of the parameter from the XQuery Update script or XSLT stylesheet. If the value of the @useCurrentContext attribute is set to true, the element name from the cursor position is used as proposed values for this parameter.

Example of an `<elementLocation>`:

```xml
<elementLocation name="elem_loc" useCurrentContext="false">
  <element label="Element location">
    <description>Element location description.</description>
  </element>
</elementLocation>
```

**attributeLocation**

This parameter is used to match attributes. For this type of parameter, the application displays two text fields where you can enter the parent element name and the attribute name (both text fields accept XPath expressions for a finer match). The text from the @label attributes is
displayed in the application as the label of the associated text fields. The @name attribute is used to specify the name of the parameter from the XQuery Update script or XSLT stylesheet. The value of this parameter is an XPath expression that is computed by using the values of the expression from the element and attribute text fields. For example, if section is entered for the element and a title is entered for the attribute, the XPath expression would be computed as //section/@title. If the value of the useCurrentContext attribute is set to true, the element and attribute name from the cursor position is used as proposed values for the operation parameters.

Example of an <attributeLocation>:

```
<attributeLocation name="attr_xpath" useCurrentContext="true">
  <element label="Element path">
    <description>Element path description.</description>
  </element>
  <attribute label="Attribute">
    <description>Attribute path description.</description>
  </attribute>
</attributeLocation>
```

**elementParameter**

This parameter is used to specify elements by local name and namespace. For this type of parameter, the application displays two combo boxes with elements and namespaces collected from the associated schema of the currently edited file. The text from the @label attribute is displayed in the application as label of the associated combo. The @name attribute is used to specify the name of the parameter from the XQuery Update script or XSLT stylesheet. If you specify the @allowsAny attribute, the application will propose <ANY> as a possible value for the Name and Namespace combo boxes. You can also use the @useCurrentContext attribute and if its value is set to true, the element name and namespace from the cursor position is used as proposed values for the operation parameters.

Example of an <elementParameter>:

```
<elementParameter id="elemID" useCurrentContext="true">
  <localName label="Name" name="element_localName" allowsAny="true">
    <description>Local name of the parent element.</description>
  </localName>
  <namespace label="Namespace" name="element_namespace" allowsAny="true">
    <description>Local name of the parent element</description>
  </namespace>
</elementParameter>
```

**attributeParameter**

This parameter is used to specify attributes by local name and namespace. For this type of parameter, the application displays two combo boxes with attributes and their namespaces
collected from the associated schema of the currently edited file. The text from the @label attribute is displayed in the application as the label of the associated combo box. You can also use the @useCurrentContext attribute and if its value is set to true, the attribute name and namespace from the cursor position is used as proposed values for the operation parameters.

**Note:**

An `<attributeParameter>` is dependant upon an `<elementParameter>`. The list of attributes and namespaces are computed based on the selection in the `elementParameter` combo boxes.

Example of an `<attributeParameter>`:

```xml
<attributeParameter dependsOn="elemID" useCurrentContext="true">
    <localName label="Name" name="attribute_localName">
        <description>The name of the attribute to be converted.</description>
    </localName>
    <namespace label="Namespace" name="attribute_namespace" allowsAny="true">
        <description>Namespace of the attribute to be converted.</description>
    </namespace>
</attributeParameter>
```

**Note:**

All built-in operations are loaded from the `[OXYGEN_INSTALL_DIR]/refactoring` folder.

**Related information**

- Example of an Operation Descriptor File with an XSLT Stylesheet *(on page 871)*
- Example of an Operation Descriptor File with an XQuery Update script *(on page 866)*

**XQuery Update Script for Creating a Custom Operation**

To demonstrate creating a custom operation, suppose that you have a task where you need to convert an attribute into an element and insert it inside another element. A specific example would be if you have a project with a variety of `<image>` elements where a deprecated `@alt` attribute was used for the description and you want to convert all instances of that attribute into an element with the same name and insert it as the first child of the `<image>` element.

Thus, the task is to convert this attribute into an element with the same name and insert it as the first child of the `<image>` element.
An XQuery Update script can be used to implement the new custom XML refactoring operation. The second requirement is an XML Refactoring operation descriptor file (on page 866) that contains the path to the XQuery Update script.

**Restriction:**
There is a limitation to using an XQuery script in that comments or processing instructions that are in any node before or after the root element cannot be modified by an XML Refactoring operation. In other words, XML Refactoring operations can only be performed on comments or processing instructions that are inside the root element. However, as a work around to this limitation, you can use Saxon extension functions and the XSLT stylesheet method (on page 873) to implement the new custom XML refactoring operation.

**Example of an XQuery Update Script for Creating a Custom Operation to Convert an Attribute to an Element**

The XQuery Update script does the following:

- Iterates over all elements from the document that have the specified local name and namespace.
- Finds the attribute that will be converted to an element.
- Computes the QName (on page 3223) of the new element to be inserted and inserts it as the first child of the parent element.

```xml
/: XQuery document used to implement 'Convert attribute to element'

  operation from XML Refactoring tool.

:)
```
declare namespace output = "http://www.w3.org/2010/xslt-xquery-serialization";
declare option output:method   "xml";
declare option output:indent   "no";

(: Local name of the attribute's parent element. :)  
declare variable $element_localName as xs:string external;

(: Namespace of the attribute's parent element. :)  
declare variable $element_namespace as xs:string external;

(: The local name of the attribute to be converted :)  
declare variable $attribute_localName as xs:string external;

(: The namespace of the attribute to be converted :)  
declare variable $attribute_namespace as xs:string external;

(: Local name of the new element. :)  
declare variable $new_element_localName as xs:string external;

(: Namespace of the new element. :)  
declare variable $new_element_namespace as xs:string external;

(: Convert attribute to element:)  
for $node in /*
(: Find the attribute to convert :)  
let $attribute :=
    $node/@*[local-name() = $attribute_localName and
    ($attribute_namespace = '<ANY>' or $attribute_namespace = namespace-uri())]

(: Compute the prefix for the new element to insert :)  
let $prefix :=
    for $p in in-scope-prefixes($node)
    where $new_element_namespace = namespace-uri-for-prefix($p, $node)
    return $p

(: Compute the qname for the new element to insert :)  
let $new_element_qName :=
    if (empty($prefix) or $prefix[1] = '') then $new_element_localName
    else $prefix[1] || ':' || $new_element_localName

where ('<ANY>' = $element_localName or local-name($node) = $element_localName)
    and
Example of an Operation Descriptor File That References the XQuery Script for Creating a Custom Operation to \textit{Convert an Attribute to an Element}

After you have developed the XQuery script, you have to create an XML Refactoring operation descriptor. This descriptor is used by the application to load the operation details such as name, description, or parameters.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<refactoringOperationDescriptor
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns="http://www.oxygenxml.com/ns/xmlRefactoring"
    id="convert-attribute-to-element"
    name="Convert attribute to element">
    <description>Converts the specified attribute to an element. The new element will be inserted as first child of the attribute's parent element.</description>
    <!-- For the XSLT stylesheet option uncomment the following line and comment the line referring the XQuery Update script -->
    <!-- <script type="XSLT" href="convert-attribute-to-element.xsl"/> -->
    <script type="XQUERY_UPDATE" href="convert-attribute-to-element.xq"/>
    <parameters>
        <description>Specify the attribute to be converted to element.</description>
        <section label="Parent element">
            <elementParameter id="elemID">
                <localName label="Name" name="element_localName" allowsAny="true">
                    <description>Local name of the parent element.</description>
                </localName>
                <namespace label="Namespace" name="element_namespace" allowsAny="true">
                    <description>Local name of the parent element</description>
                </namespace>
            </elementParameter>
        </section>
        <section label="Attribute">
            <attributeParameter dependsOn="elemID">
                <localName label="Name" name="attribute_localName">
                </localName>
            </attributeParameter>
        </section>
    </parameters>
</refactoringOperationDescriptor>
```
Results

After you have created these files, copy them into a folder scanned by Oxygen XML Editor when it loads the custom operation (on page 875). When the XML Refactoring tool is started again, you will see the created operation.

Since various parameters can be specified, this custom operation can also be used for other similar tasks. The following image shows the parameters that can be specified in the example of the custom operation to convert an attribute to an element:
XSLT Stylesheet for Creating a Custom Operation

To demonstrate creating a custom operation, suppose that you have a task where you need to convert an attribute into an element and insert it inside another element. A specific example would be if you have a project with a variety of `<image>` elements where a deprecated `@alt` attribute was used for the description and you want to convert all instances of that attribute into an element with the same name and insert it as the first child of the `<image>` element.

Thus, the task is to convert this attribute into an element with the same name and insert it as the first child of the image element.
An XSLT stylesheet can be used to implement the new custom XML refactoring operation. The second requirement is an XML Refactoring operation descriptor file (on page 871) that contains the path to the XSLT stylesheet.

**Example of an XSLT Script for Creating a Custom Operation to Convert an Attribute to an Element**

The XSLT stylesheet does the following:

- Iterates over all elements from the document that have the specified local name and namespace.
- Finds the attribute that will be converted to an element.
- Adds the new element as the first child of the parent element.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    xmlns:xs="http://www.w3.org/2001/XMLSchema"
    exclude-result-prefixes="xs"
    xmlns:xr="http://www.oxygenxml.com/ns/xmlRefactoring"
    version="2.0">


    <xsl:param name="element_localName" as="xs:string" required="yes"/>
    <xsl:param name="element_namespace" as="xs:string" required="yes"/>
    <xsl:param name="attribute_localName" as="xs:string" required="yes"/>
    <xsl:param name="attribute_namespace" as="xs:string" required="yes"/>
    <xsl:param name="new_element_localName" as="xs:string" required="yes"/>
    <xsl:param name="new_element_namespace" as="xs:string" required="yes"/>

    <xsl:template match="node() | @*">
        <xsl:copy>
            <xsl:apply-templates select="node() | @*"/>
        </xsl:copy>
    </xsl:template>

    <xsl:template match="/*[xr:check-local-name($element_localName, ., true())
        and
        xr:check-namespace-uri($element_namespace, .)]"></xsl:template>

    <xsl:variable name="attributeToConvert"
        select="@*[xr:check-local-name($attribute_localName, ., true())
        and
        xr:check-namespace-uri($attribute_namespace, .)]"/>
```

```xml
</xsl:stylesheet>
```
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:choose>
    <xsl:when test="empty($attributeToConvert)">
      <xsl:copy>
        <xsl:apply-templates select="node() | @*"/>
      </xsl:copy>
    </xsl:when>
    <xsl:otherwise>
      <xsl:copy>
        <xsl:for-each select="@*[empty(. intersect $attributeToConvert)]">
          <xsl:copy-of select="."/>
        </xsl:for-each>
        <!-- The new element namespace -->
        <xsl:variable name="nsURI" as="xs:string">
          <xsl:choose>
            <xsl:when test="$new_element_namespace eq $xr:NO-NAMESPACE">
              <xsl:value-of select="''"/>
            </xsl:when>
            <xsl:otherwise>
              <xsl:value-of select="$new_element_namespace"/>
            </xsl:otherwise>
          </xsl:choose>
        </xsl:variable>
        <xsl:element name="{$new_element_localName}" namespace="{$nsURI}"
          >
          <xsl:value-of select="$attributeToConvert"/>
        </xsl:element>
        <xsl:apply-templates select="node()"/>
      </xsl:copy>
    </xsl:otherwise>
  </xsl:choose>
</xsl:stylesheet>

Note:
The XSLT stylesheet imports a module library that contains utility functions and variables. The location of this module is resolved via an XML Catalog (on page 3225) set in the XML Refactoring framework (on page 3219).
Example of an Operation Descriptor File That References the XSLT Stylesheet for Creating a Custom Operation to *Convert an Attribute to an Element*

After you have developed the XSLT stylesheet, you have to create an XML Refactoring operation descriptor. This descriptor is used by the application to load the operation details such as name, description, or parameters.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<refactoringOperationDescriptor
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns="http://www.oxygenxml.com/ns/xmlRefactoring"
    id="convert-attribute-to-element"
    name="Convert attribute to element">
  <description>Converts the specified attribute to an element.
      The new element will be inserted as first child of the attribute's parent element.</description>
  <!-- For the XSLT stylesheet option uncomment the following line and comment the line referring the XQuery Update script -->
  <!-- <script type="XSLT" href="convert-attribute-to-element.xsl"/> -->
  <!-- <script type="XQUERY_UPDATE" href="convert-attribute-to-element.xq"/> -->
  <script type="XQUERY_UPDATE" href="convert-attribute-to-element.xq"/>
  <parameters>
    <description>Specify the attribute to be converted to element.</description>
    <section label="Parent element">
      <elementParameter id="elemID">
        <localName label="Name" name="element_localName" allowsAny="true">
          <description>Local name of the parent element.</description>
        </localName>
        <namespace label="Namespace" name="element_namespace" allowsAny="true">
          <description>Local name of the parent element</description>
        </namespace>
      </elementParameter>
    </section>
    <section label="Attribute">
      <attributeParameter dependsOn="elemID">
        <localName label="Name" name="attribute_localName">
          <description>Name of the attribute to be converted.</description>
        </localName>
        <namespace label="Namespace" name="attribute_namespace" allowsAny="true">
          <description>Namespace of the attribute to be converted.</description>
        </namespace>
      </attributeParameter>
    </section>
  </parameters>
</refactoringOperationDescriptor>
```
<section label="New element">
  <elementParameter>
    <localName label="Name" name="new_element_localName">
      <description>The name of the new element.</description>
    </localName>
    <namespace label="Namespace" name="new_element_namespace">
      <description>The namespace of the new element.</description>
    </namespace>
  </elementParameter>
</section>

Note:
If you are using an XSLT file, the line with the <script> element would look like this:

```xml
<script type="XSLT" href="convert-attribute-to-element.xsl"/>
```

The code exemplified above and other refactoring examples can be found on the DITA Refactoring GitHub sample project.

Results

After you have created these files, copy them into a folder scanned by Oxygen XML Editor when it loads the custom operation (on page 875). When the XML Refactoring tool is started again, you will see the created operation.

Since various parameters can be specified, this custom operation can also be used for other similar tasks. The following image shows the parameters that can be specified in the example of the custom operation to convert an attribute to an element:
Using Saxon Extension Functions to Allow Custom Refactoring Operations to Read and Modify Content Outside the Root Node

One advantage to using an XSLT stylesheet is that there is limitation when using an XQuery Update script (on page 863) in that refactoring operations can only be performed on comments or processing instructions that are inside the root element. Thus, using the XQuery method, comments or processing instructions that are in any node before or after the root element cannot be modified by an XML Refactoring operation.

The XSLT stylesheet method offers a work-around to this limitation through the use of some Saxon extension functions.

To illustrate the use of these functions, consider the following sample XML file:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<!-- comment before root -->
<?pi before root ?>
<root>
    <child/>
</root>
<!-- comment after root -->
<?pi after root ?>
```

The following Saxon extension functions can be used to read and modify content outside the root node:

**Note:**
They belong to the http://www.oxygenxml.com/ns/xmlRefactoring/functions namespace.
• **get-content-after-root()** - Returns the content after root as `xs:string`.

For the XML above, the call of this function will return the following string value:

```xml
<!-- comment after root -->
<?pi after root ?>
```

• **set-content-after-root(xs:string)** - Updates the content that will be serialized in the refactored document after the root node.

The function call `set-content-after-root('<!-- Inserted comment -->')` will result in replacing the nodes after the root element with the comment passed as string argument. The XML document will be modified as follows:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<!-- comment before root -->
<?pi before root ?>
<root>
  <child/>
</root>
<!-- Inserted comment -->
```

• **get-content-before-root()** - Returns the content before root as `xs:string`.

For the XML above, the call of this function will return the following string value:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<!-- comment before root -->
<?pi before root ?>
```

• **set-content-before-root(xs:string)** - Updates the content that will be serialized in the refactored document after the root node.

The function call `set-content-before-root('<!-- Inserted comment -->')` will result in replacing the nodes before the root element with the comment passed as string argument. The XML document will be modified as follows:

```xml
<-- Inserted comment -->
<root>
  <child/>
</root>
<!-- comment after root -->
```

**XSLT Example:**

To process content after the root node, the XSLT would look like this:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  xmlns:xs="http://www.w3.org/2001/XMLSchema" exclude-result-prefixes="xs"/>
```
  <xsl:template match="/">
    <!-- The comment content that will be inserted after the root element -->
    <xsl:variable name="commentAsText"><!-- COMMENT ADDED FROM XR OPERATION-->
    </xsl:variable>
    <!-- Retrieve the content after the root element as is -->
    <xsl:variable name="after-root-content" as="xs:string">
      <!--xf:call to get content after root element-->
    </xsl:variable>
    <xsl:variable name="processedContent">
      <!--xrf:call to set content after root element-->
    </xsl:variable>
    <!-- Update the content after the root element -->
    <xsl:value-of select="xrf:set-content-after-root($processedContent)"/>
  </xsl:template>
  
  <xsl:template match="node() | @*">
    <xsl:copy>
      <xsl:apply-templates select="node() | @*"/>
    </xsl:copy>
  </xsl:template>
</xsl:stylesheet>

Note:
The above XSLT retrieves the nodes after the root element as string, appends a new comment, and then sets back the updated content into the XML document.

Storing and Sharing Refactoring Operations

Oxygen XML Editor scans the following locations when looking for XML Refactoring operations to provide flexibility:

- A **refactoring** folder, created inside a directory that is associated to a *framework* you are customizing. To associate the parent directory to the *framework*, you need to add it in the **Classpath** tab of the [Document type](on page 144) configuration dialog box.
- A folder that you specify in the **Load additional refactoring operations from** text box (on page 269) in the [XML Refactoring](on page 269) preferences page.
Note:
If you share a project with your team, you can also share the custom operation by doing the following:

1. Save the custom operation in a folder that is part of your project.
2. Switch the XML Refactoring option page to project level (on page 3222):
   a. Open the Preferences dialog box (Options > Preferences) (on page 124) and go to XML > XML Refactoring (on page 269).
   b. Select Project Options (on page 3222) at the bottom of the dialog box.
3. In the Load additional refactoring operations from text box (on page 269), use the ${pd} editor variable (on page 331) so that the folder path is declared relative to the project.

- A folder specified by the XML Refactoring Operations Plugin Extension (on page 2432).
- The refactoring folder from the Oxygen XML Editor installation directory ([OXYGEN_INSTALL_DIR]/refactoring/).

Sharing Custom Refactoring Operations

The purpose of Oxygen XML Editor scanning multiple locations for the XML Refactoring operations is to provide more flexibility for developers who want to share the refactoring operations with the other team members. Depending on your particular use case, you can attach the custom refactoring operations to other resources, such as framework (on page 3219) or projects.

After storing custom operations, you can share them with other users by sharing the resources.

Localizing XML Refactoring Operations

Oxygen XML Editor includes localization support for the XML refactoring operations.

The translation keys for the built-in refactoring operations are located in [OXYGEN_INSTALL_DIR]/refactoring/il8n/translation.xml.

The localization support is also available for custom refactoring operations. The following information can be translated:

- The operation name, description, and category.
- The <description> of the <parameters> element.
- The label, description, and possibleValues for each parameter.

Translated refactoring information uses the following form:

\${il8n(translation_key)}

Oxygen XML Editor scans the following locations to find the translation.xml files that are used to load the translation keys:
• A refactoring/i18n folder, created inside a directory that is associated to a customized framework.
• A i18n folder, created inside a directory that is associated to a customized framework.
• An i18n folder inside any specified folder. In this case, you need to open the Preferences dialog box (Options > Preferences) (on page 124), go to XML > XML Refactoring, and specify the folder in the Load additional refactoring operations from text box.
• An i18n folder located in directories specified through the XML Refactoring Operations Plugin Extension (on page 2432).
• The refactoring/i18n folder from the Oxygen XML Editor installation directory (/OXYGEN_INSTALL_DIR/refactoring/i18n).

Example: Refactoring Operation Descriptor File with i18n Support

```xml
<?xml version="1.0" encoding="UTF-8"?>

<refactoringOperationDescriptor
    xmlns="http://www.oxygenxml.com/ns/xmlRefactoring"
    id="remove_text_content"
    name="${i18n(Remove_text_content)}">

    <description>${i18n(Remove_text_content_description)}</description>

    <script type="XQUERY_UPDATE" href="remove_text_content.xq"/>

    <parameters>

        <description>${i18n(parameters_description)}</description>

        <parameter label="${i18n(Element_name)}" name="element_localName"
            type="NC_NAME">

            <description>${i18n(Element_name_descriptor)}</description>

            <possibleValues>

                <value default="true" name="value1">${i18n(value_1)}</value>

                <value name="value2">${i18n(value_2)}</value>

            </possibleValues>

        </parameter>

    </parameters>

</refactoringOperationDescriptor>
```

XML Digital Signatures

This chapter explains how to apply and verify digital signatures on XML documents.

Digital Signatures Overview

Digital signatures are widely used as security tokens, not just in XML. A digital signature provides a mechanism for assuring integrity of data, the authentication of its signer, and the non-repudiation of the entire signature to an external party:
A digital signature must provide a way to verify that the data has not been modified or replaced to ensure integrity.

The signature must provide a way to establish the identity of the data's signer for authentication.

The signature must provide the ability for the data's integrity and authentication to be provable to a third party for non-repudiation.

A public key system is used to create the digital signature and it is also used for verification. The signature binds the signer to the document because digitally signing a document requires the originator to create a hash of the message and then encrypt that hash value with their own private key. Only the originator has that private key and that person is the only one who can encrypt the hash so that it can be unencrypted using their public key. The recipient, upon receiving both the message and the encrypted hash value, can decrypt the hash value, knowing the originator's public key. The recipient must also try to generate the hash value of the message and compare the newly generated hash value with the unencrypted hash value received from the originator. If the hash values are identical, it proves that the originator created the message, because only the actual originator could encrypt the hash value correctly.

XML Signatures can be applied to any digital content (data object), including XML (see W3C Recommendation, XML-Signature Syntax and Processing). An XML Signature may be applied to the content of one or more resources:

- Enveloped or enveloping signatures are applied over data within the same XML document as the signature
- Detached signatures are applied over data external to the signature element; the signature is "detached" from the content it signs. This definition typically applies to separate data objects, but it also includes the instance where the signature and data object reside within the same XML document but are sibling elements.

The XML Signature is a method of associating a key with referenced data. It does not normatively specify how keys are associated with persons or institutions, nor the meaning of the data being referenced and signed.

The original data is not actually signed. Instead, the signature is applied to the output of a chain of canonicalization (on page 3217) and transformation algorithms, which are applied to the data in a designated sequence. This system provides the flexibility to accommodate whatever "normalization" or desired preprocessing of the data that might be required or desired before subjecting it to being signed.

Since the signature is dependent on the content it is signing, a signature produced from a non-canonicalized document could possibly be different from one produced from a canonicalized (on page 3217) document. The canonical (on page 3217) form of an XML document is physical representation of the document produced by the method described in this specification. The XML canonicalization (on page 3217) method is the algorithm defined by this specification that generates the canonical form of a given XML document or document subset. XML canonicalization is designed to be useful for applications that require the ability to test whether or not the information content of a document or document subset has been changed. This is done by comparing the canonical form of the original document before application processing with the canonical form of the document result of the application processing.
A digital signature over the canonical (on page 3217) form of an XML document or document subset allows the signature digest calculations to be oblivious to changes in the original document's physical representation. During signature generation, the digest is computed over the canonical form of the document. The document is then transferred to the relying party, which validates the signature by reading the document and computing a digest of the canonical form of the received document. The equivalence of the digests computed by the signing and relying parties (hence, the equivalence of the canonical forms that they were computed for) ensures that the information content of the document has not been altered since it was signed.

The following canonicalization algorithms are used in Oxygen XML Editor:

- **Canonical XML (or Inclusive XML Canonicalization) (XMLC14N)** - Used for XML where the context doesn't change.

  Inclusive Canonicalization copies all the declarations, even if they are defined outside of the scope of the signature, and all the declarations you might use will be unambiguously specified. Inclusive Canonicalization is useful when it is less likely that the signed data will be inserted in other XML document and it is the safer method from the security standpoint because it requires no knowledge of the data that are to be secured to safely sign them. A problem may occur if the signed document is moved into another XML document that has other declarations because the Inclusive Canonicalization will copy them and the signature will be invalid.

- **Exclusive XML Canonicalization (EXCC14N)** - Designed for canonicalization where the context might change.

  Exclusive Canonicalization just copies the namespaces you are actually using (the ones that are a part of the XML syntax). It does not look into attribute values or element content, so the namespace declarations required to process these are not copied. This is useful if you have a signed XML document that you want to insert into other XML documents (or you need self-signed structures that support placement within various XML contexts), as it will ensure the signature is verified correctly each time.

The canonicalization (on page 3217) method can specify whether or not comments should be included in the canonical form output by the XML canonicalization method. If a canonical form contains comments corresponding to the comment nodes in the input node-set, the result is called canonical XML with comments. In an uncommented canonical form, comments are removed, including the delimiter for comments outside the document element.

The three operations. **Canonicalize (on page 880), Sign (on page 882),** and **Verify Signature (on page 885),** are available from the **Source** submenu when invoking the contextual menu in **Text** mode or from the **Tools** menu.

Related Information:

- Certificates (on page 880)
- Canonicalizing Files (on page 880)
Certificates

A certificate is a digitally signed statement from the issuer (an individual, an organization, a website or a firm), saying that the public key (and some other information) of some other entity has a particular value. When data is digitally signed, the signature can be verified to check the data integrity and authenticity. Integrity means that the data has not been modified. Authenticity means the data comes indeed from the entity that claims to have created and signed it. Certificates are kept in special repositories called keystore.

All keystore entries (key and trusted certificate entries) are accessed via unique aliases. An alias must be assigned for every new entry of either a key or certificate as a reference for that entity. No keystore can store an entity if its alias already exists in that keystore and cannot store trusted certificates generated with keys in its keystore.

Oxygen XML Editor provides two types of keystores: Java Key Store (JKS) and Public-Key Cryptography Standards version 12 (PKCS-12). A keystore file is protected by a password. In a PKCS 12 keystore you should not store a certificate without alias together with other certificates, with or without alias, as in such a case the certificate without alias cannot be extracted from the keystore.

To configure the options for a certificate or to validate it, open the Preferences dialog box (Options > Preferences) and go to XML > XML Signing Certificates. This opens the certificates preferences page.

Related Information:
- Digital Signatures Overview (on page 877)

Canonicalizing Files

You can select the canonicalization (on page 3217) algorithm to be used for a document from the dialog box that is displayed by using the Canonicalize action that is available from the Source submenu when invoking the contextual menu in Text mode or from the Tools menu.
The **Canonicalize** dialog box allows you to set the following options:

- **Input URL** - Available if the **Canonicalize** action was selected from the **Tools** menu. It allows you to specify the location of the input file.
- **Exclusive** - If selected, the exclusive (uncommented) canonicalization (on page 3217) method is used.

**Note:**

*Exclusive Canonicalization* just copies the namespaces you are actually using (the ones that are a part of the XML syntax). It does not look into attribute values or element content, so the namespace declarations required to process these are not copied. This is useful if you have a signed XML document that you want to insert into other XML documents (or you need self-signed structures that support placement within various XML contexts), as it will ensure the signature is verified correctly each time.

- **Exclusive with comments** - If selected, the exclusive with comments canonicalization (on page 3217) method is used.
- **Inclusive** - If selected, the inclusive (uncommented) canonicalization (on page 3217) method is used.

**Note:**

*Inclusive Canonicalization* copies all the declarations, even if they are defined outside of the scope of the signature, and all the declarations you might use will be unambiguously specified. Inclusive **Canonicalization** is useful when it is less likely that the signed data will be inserted
in other XML document and it is the safer method from the security standpoint because it requires no knowledge of the data that are to be secured to safely sign them. A problem may occur if the signed document is moved into another XML document that has other declarations because the Inclusive *Canonicalization* will copy them and the signature will be invalid.

- **Inclusive with comments** - If selected, the inclusive with comments [canonicalization (on page 3217)] method is used.
- **XPath** - The XPath expression provides the fragments of the XML document to be signed.
- **Output** - Available if the Canonicalize action was selected from the **Tools** menu. It allows you to specify the output file path where the signed XML document will be saved.
- **Open in editor** - If selected, the output file will be opened in the editor.

**Related Information:**

Digital Signatures Overview *(on page 877)*

**Signing Files**

You can select the type of signature to be used for documents from a signature settings dialog box. To open this dialog box, select the **Sign** action from the **Source** submenu when invoking the contextual menu in **Text** mode or from the **Tools** menu.
The following options are available:

**Note:**
If Oxygen XML Editor could not find a valid certificate, a link is provided at the top of the dialog box that opens the [XML Signing Certificates preferences page](on page 268) where you can configure a valid certificate.

- **Input** - Available if the **Sign** action was selected from the **Tools** menu. Specifies the location of the input URL.
- **Transformation Options** - See the [Digital Signature Overview](on page 877) section for more information about these options.
None - If selected, no canonicalization (on page 3217) algorithm is used.

Exclusive - If selected, the exclusive (uncommented) canonicalization (on page 3217) method is used.

Note:

Exclusive Canonicalization just copies the namespaces you are actually using (the ones that are a part of the XML syntax). It does not look into attribute values or element content, so the namespace declarations required to process these are not copied. This is useful if you have a signed XML document that you want to insert into other XML documents (or you need self-signed structures that support placement within various XML contexts), as it will ensure the signature is verified correctly each time.

Exclusive with comments - If selected, the exclusive with comments canonicalization (on page 3217) method is used.

Inclusive - If selected, the inclusive (uncommented) canonicalization (on page 3217) method is used.

Note:

Inclusive Canonicalization copies all the declarations, even if they are defined outside of the scope of the signature, and all the declarations you might use will be unambiguously specified. Inclusive Canonicalization is useful when it is less likely that the signed data will be inserted in other XML document and it is the safer method from the security standpoint because it requires no knowledge of the data that are to be secured to safely sign them. A problem may occur if the signed document is moved into another XML document that has other declarations because the Inclusive Canonicalization will copy them and the signature will be invalid.

Inclusive with comments - If selected, the inclusive with comments canonicalization (on page 3217) method is used.

• XPath - The XPath expression provides the fragments of the XML document to be signed.
• ID - Provides ID of the XML element to be signed.
• Envelope - If selected, the enveloped signature is used. See the Digital Signature Overview (on page 877) for more information.
• Detached - If selected, the detached signature is used. See the Digital Signature Overview (on page 877) for more information.
• Append KeyInfo - If this option is selected, the <ds:KeyInfo> element will be added in the signed document.
• Signature algorithm - The algorithm used for signing the document. The following options are available: RSA with SHA1, RSA with SHA256, RSA with SHA384, and RSA with SHA512.
• Output - Available if the Sign action was selected from the Tools menu. Specifies the path of the output file where the signed XML document will be saved.
• Open in editor - If selected, the output file will be opened in Oxygen XML Editor.
Verifying Signature

You can verify the signature of a file by selecting the **Verify Signature** action from the **Source** submenu when invoking the contextual menu in **Text** mode or from the **Tools** menu. The **Verify Signature** dialog box then allows you to specify the location of the file whose signature is verified.

If the signature is valid, a dialog box displays the name of the signer. Otherwise, an error shows details about the problem.

Example of How to Digitally Sign XML Files or Content

Suppose you want to digitally sign an XML document, but more specifically, suppose you have multiple instances of the same element in the document and you just want to sign a specific ID. Oxygen XML Editor includes a signature tool that allows you to digitally sign XML documents or specific content.

The Oxygen XML Editor installation directory includes a **samples** folder that contains a file called **personal.xml**. For the purposes of this example, this file will be used to demonstrate how to digitally sign specific content. Notice that this file has multiple **<person>** elements inside the **<personnel>** element. Suppose you want to digitally sign the specific **<person>** element that contains the **id=robert.taylor**. To do this, follow this procedure:

1. Open the **personal.xml** file in Oxygen XML Editor in **Text** editing mode.
2. Right-click anywhere in the editor and select the **Sign** action from the **Source** submenu. The **Sign** dialog box is displayed.

**Tip:**

If you want to sign a file but create a new output file so that the original file remains unchanged, use the **Sign** action from the **Tools** menu. Selecting the action from this menu will allow you to choose an input file and output file in the **Sign** dialog box.
3. If Oxygen XML Editor cannot find a valid certificate, click the link at the top of the dialog box to **configure a valid certificate**. This opens the XML Signing Certificates preferences page (on page 268) that allows you to configure and validate a certificate.

4. Once a valid certificate is recognized, continue to configure the Sign dialog box.
   a. Select one of the Transformation Options (on page 883). For the purposes of this example, select the Inclusive with comments option.
   b. Specify the appropriate XPath expression for the specific element that needs to be signed. For this example, type /personnel/person in the XPath text box.
   c. Enter the specific ID that needs to be signed. For this example, type robert.taylor in the ID field.
   d. Select the Envelope option (on page 884) and leave the other options as their default values.

The digital signature is added at the end of the XML document, just before the end tag. It is always added at the end of the document, even if you only sign specific content within the document.

5. You can verify the signature by choosing the Verify Signature action from the Source submenu of the contextual menu.

### Related Information:
- Digital Signatures Overview (on page 877)
- Signing Files (on page 882)
- Verifying Signature (on page 885)

---

**Editing XSLT Stylesheets**

Oxygen XML Editor includes a built-in editor for XSLT stylesheets. This section presents the features of the XSLT editor and how these features can be used. The features of the XSLT editor include:

- **Create new XSLT files and templates** - You can use the built-in new file wizards to create new XSLT documents or templates (on page 368).
- **Open and Edit XSLT files** - XSLT files can be opened and edited in the source editor (Text mode (on page 517)).
- **Visual Editing** - XSLT stylesheets are rendered, and can be edited, in the visual Author editing mode (on page 588).
- **Validation** - Presents validation errors in XSLT files.
- **Content completion** - Offers proposals for properties and the values that are available for each property.
- **Syntax highlighting** - The syntax highlighting in Oxygen XML Editor makes XSLT files more readable.

### Resources

For more information about working with XSLT in Oxygen XML Editor, see the following resources:

- Webinar: **Introduction to XSLT Using Oxygen**.
- Webinar: **XSLT Quick Fixes**.
Modular Contextual XSLT Editing Using 'Main Files' Support

Smaller interrelated modules that define a complex stylesheet cannot be correctly edited or validated individually, due to their interdependency with other modules. For example, a function defined in a main stylesheet is not visible when you edit an included or imported module. Oxygen XML Editor provides the support for defining the main module (or modules), allowing you to edit any of the imported/included files in the context of the larger stylesheet structure.

You can set a main XSLT stylesheet either using the main files support from the Project view (on page 418), or using a validation scenario.

To set a main file using a validation scenario, add validation units that point to the main modules. Oxygen XML Editor warns you if the current module is not part of the dependencies graph computed for the main stylesheet. In this case, it considers the current module as the main stylesheet.

The advantages of editing in the context of main file (on page 3220) include:

- Correct validation of a module in the context of a larger stylesheet structure.
- Content Completion Assistant (on page 3217) displays all components valid in the current context.
- The Outline view (on page 899) displays the components collected from the entire stylesheet structure.

For more information about editing XSLT stylesheets in the main files context, watch our video demonstration:

https://www.youtube.com/embed/UZwg385RKNw

Related Information:
- XSLT Resource Hierarchy/Dependencies View (on page 906)
- XSLT Component Dependencies View (on page 909)

Validating XSLT Stylesheets

Numerous XSLT code quality assurance checks are done during automatic validation to help you keep your stylesheets valid and well-formed. Oxygen XML Editor performs the validation of XSLT documents with the help of an XSLT processor that you can configure in the preferences pages (on page 245) according to the XSLT version.

For XSLT 1.0, the options are: Xalan, Saxon 6.5.5, Saxon 10.6 and a JAXP transformer specified by the main Java class. (on page 245) For XSLT 2.0, the options are: Saxon 10.6 and a JAXP transformer specified by the main Java class. (on page 245) For XSLT 3.0, the options are Saxon 10.6 and a JAXP transformer specified by the main Java class. (on page 245)

To access the XSLT preferences (on page 245) quickly, use the Validation options action from the Document > Validate menu.
Creating a Validation Scenario for XSLT Stylesheets

You can validate an XSLT document using the engine defined in the transformation scenario, or a custom validation scenario. If you choose to validate using the engine from transformation scenario, and a transformation scenario is not associated with the current document or the engine has no validation support, the default engine is used. To set the default engine, open the Preferences dialog box (Options > Preferences) (on page 124) and go to XML > XSLT/FO/XQuery > XSLT.

You can also create new validation scenarios or edit existing ones, and you can add JARS (on page 3220) and classes that contain extension functions. To create or edit a validation scenario for an XSLT stylesheet, follow these steps:

1. With the XSLT file open in Oxygen XML Editor, select the Configure Validation Scenario(s) from the Document > Validate menu, or the Validation toolbar drop-down menu, or from the Validate submenu when invoking the contextual menu on the XSLT file in the Project view (on page 402).

   The Configure Validation Scenario(s) dialog box is displayed. It contains the existing scenarios, organized in categories depending on the type of file they apply to. You can use the options in the Settings drop-down menu to filter which scenarios are shown.

2. To edit an existing scenario, select the scenario and click the Edit button. If you try to edit one of the read-only built-in scenarios, Oxygen XML Editor creates a customizable duplicate (you can also use the Duplicate button).

3. To add a new scenario, click the New button.

   The New scenarios dialog box is displayed. It lists all validation units of the scenario.

   Figure 254. Add / Edit a Validation Unit

![New scenario dialog box]

4. Configure the following information in this dialog box:
   a. Name - The name of the validation scenario.
   b. Storage - You can choose between storing the scenario in the Project Options (on page 3222) or Global Options (on page 3219).
c. **URL of the file to validate** - In most cases, leave this field as the default selection (the URL of the current file). If you want to specify a different URL, double-click its cell and enter the URL in the text field, select it from the drop-down list, or use the ![Browse](Image) drop-down menu or ![Insert Editor Variable](Image) button.

d. **File type** - The file type should be XSLT Document.

e. **Validation engine** - Click the cell to select a validation engine. You must select an engine to be able to add or edit extensions.

f. **Automatic validation** - If this option is selected, the validation operation defined by this row is also used by the [automatic validation feature](on page 775).

5. To add or edit extensions, click the ![Edit extensions](Image) button. This button is only available if the **File type** is set as XSLT Document and a **Validation engine** is chosen. The Libraries dialog box is opened. It is used to specify the JARS and classes that contain extension functions called from the XSLT file of the current validation scenario. They will be searched, in the specified extensions, in the order displayed in this dialog box. To change the order of the items, select the item and click the ![Move up](Image) or ![Move down](Image) buttons.

6. Click **OK** to close the New scenario dialog box. The newly created validation scenario is now included in the list of scenarios in the [Configure Validation Scenario(s)](on page 775) dialog box. You can select the scenario in this dialog box to associate it with the current XSLT document and click the **Apply associated** button to run the validation scenario.

### Validating XSLT Stylesheets with Custom Engines

If you need to validate an XSLT stylesheet with a validation engine that is different from the built-in engine, you can configure external engines as custom XSLT validation engines in the Oxygen XML Editor preferences. After a custom validation engine is properly configured (on page 227), it can be applied on the current document by selecting it from the list of custom validation engines in the ![Validation](Image) toolbar drop-down menu. The document is validated against the schema declared in the document.

By default, there are two validators that are configured for XSLT stylesheets:

- **MSXML 4.0 (Deprecated)** - included in Oxygen XML Editor (Windows edition). It is associated to the XSL Editor type in [Preferences page](on page 227).
- **MSXML.NET (Deprecated)** - included in Oxygen XML Editor (Windows edition). It is associated to the XSL Editor type in [Preferences page](on page 227).

### Validating XSLT Stylesheets that Call Java Extensions

It is possible to validate an XSLT that calls Java extensions. This is achieved through a transformation scenario where the Java extensions are specified, and the default validation will be processed using the parameters defined in the transformation scenario.

To validate XSLT with Java extensions, follow this procedure:
1. Create an **XSLT transformation scenario (on page 1510)** for your XSLT document (select \(\text{Configure Transformation Scenario(s)}\) action from the toolbar, then click **New**, and select **XSLT transformation**).

2. In the **New scenario** dialog box, click the **Extensions** button (in the **XSLT** tab), specify the Java extensions (JAR libraries) that are needed, and click **OK**.

3. Once you are finished configuring the transformation scenario, click **OK**, then select **Save and close**.

4. Use the **Validate** button on the toolbar (or **Ctrl + Shift + V (Command + Shift + V on macOS)**) and the default validation will detect and use the transformation scenario profile you just configured and saved.

Related Information:

- Debugging XSLT that Call Java Extensions (on page 2131)

**XSLT Quick Fix Support**

The Oxygen XML Editor **Quick Fix support (on page 3223)** helps you resolve various errors that appear in a stylesheet by proposing **Quick Fixes** to problems such as missing templates, misspelled template names, missing functions, or references to an undeclared variable or parameter.

To activate this feature, hover over or place the cursor in the highlighted area of text where a validation error or warning occurs. If a **Quick Fix** is available for that particular error or warning, you can access the **Quick Fix** proposals with any of the following methods:

- When hovering over the error or warning, the proposals are presented in a tooltip pop-up window.
- If you place the cursor in the highlighted area where a validation error or warning occurs, a **Quick Fix** icon (\(\text{?}\)) is displayed in the stripe on the left side of the editor. If you click this icon, Oxygen XML Editor displays the list of available fixes.
- With the cursor placed in the highlighted area of the error or warning, you can also invoke the **Quick Fix** menu by pressing **Alt + 1 (Command + Alt + 1 on macOS)** on your keyboard.

**Note:**

The **Quick Fixes** are available only when validating an XSLT file with Saxon HE/PE/EE.

**Figure 255. Example of an Undefined XSLT Functions Quick Fix**

![Figure 255. Example of an Undefined XSLT Functions Quick Fix](image-url)
Oxygen XML Editor provides XSLT *Quick Fixes* for the following types of instances:

- **Template does not exist**, when the template name referenced in a `<call-template>` element does not exist. The following fixes are available:
  - *Create template “templateName”* - creates a template and generates its corresponding parameters. The template name and parameter names and types are collected from the `<call-template>` element.
  - *Change reference to “newTemplateName”* - changes the name of the missing template referenced in the `<call-template>` element. The proposed new names are the existing templates with names similar with the missing one.

- **Variable/Parameter not declared**, when a parameter or variable reference cannot be found. The following fixes are available:
  - *Create global variable “varName”* - creates a global variable with the specified name in the current stylesheet. The new variable is added at the beginning of the stylesheet after the last global variable or parameter declaration.
  - *Create global parameter "paramName"* - creates a global parameter with the specified name in the current stylesheet. The new parameter is added at the beginning of the stylesheet after the last global parameter or variable declaration.
  - *Create local variable “varName”* - creates a local variable with the specified name before the current element.
  - *Create template parameter “paramName”* - creates a new parameter with the specified name in the current template. This fix is available if the error is located inside a template.
  - *Create function parameter “paramName”* - creates a new parameter with the specified name in the current function. This fix is available if the error is located inside a function.
  - *Change reference to "varName"* - changes the name of the referenced variable/parameter to an existing local or global variable/parameter, that has a similar name with the current one.

- **Parameter from a called template is not declared**, when a parameter referenced from a `<call-template>` element is not declared. The following fixes are available:
Create parameter "paramName" in the template "templateName" - creates a new parameter with the specified name in the referenced template.

Change "paramName" parameter reference to "newParamName" - changes the parameter reference from the <call-template> element to a parameter that is declared in the called template.

Remove parameter "paramName" from call-template - removes the parameter with the specified name from the <call-template> element.

- No value supplied for required parameter, when a required parameter from a template is not referenced in a <call-template> element. The Add parameter "paramName" in call-template quick-fix is available. It creates a new parameter with the specified name in call-template element.

- Function "prefix:functionName()" has not been defined, when a function declaration is not found. The following Quick Fixes are available:
  - Create function "prefix:functionName(param1, param2)" - creates a new function with the specified signature, after the current top-level element from stylesheet.
  - Change function to "newFunctionName(...)" - changes the referenced function name to an already defined function. The proposed names are collected from functions with similar names and the same number of parameters.

- Attribute-set "attrSetName" does not exist, when the referenced attribute set does not exist. The following Quick Fixes are available:
  - Create attribute-set "attrSetName" - creates a new attribute set with the specified name, after the current top-level element from stylesheet.
  - Change reference to "attrSetName" - changes the referenced attribute set to an already defined one.

- Character-map "characterMap" has not been defined, when the referenced character map declaration is not found. The following Quick Fixes are available:
  - Create character-map "characterMapName" - creates a new character map with the specified name, after the current top-level element from stylesheet.
  - Change reference to "characterMapName" - changes the referenced character map to an already defined one.

Content Completion in XSLT Stylesheets

The list of proposals offered by the Content Completion Assistant (on page 3217) in XSLT are context-sensitive and includes proposals that are valid at the current cursor position. It can be manually activated with the Ctrl + Space (Command + Space on macOS) shortcut.

You can enhance the list of proposals by specifying an additional schema. This schema is defined in the Content Completion / XSLT preferences (on page 213) page and can be any of the following: XML Schema, DTD, RELAX NG schema, or NVDL schema.
The feature is activated in **Text** mode in the following situations:

- After you enter the `<` character when inserting an element, it is automatically activated after a short delay. You can adjust the activation delay with the **Activation delay of the proposals window (ms)** option (on page 213) from the **Content Completion** preferences page.

- After typing a partial element or attribute name, you can manually activate it by pressing **Ctrl + Space** (Command + Space on macOS) or **Alt + ForwardSlash** (Command + Alt + ForwardSlash on macOS). If there is only one valid proposal at the current location, it is inserted without displaying the list of proposals.

The **Content Completion Assistant** proposes numerous item types (such as templates, variables, parameters, keys, etc.) that are defined in the current stylesheet, and in the imported and included XSLT stylesheets. The **Content Completion Assistant** also includes **code templates that can be used to quickly insert code fragments** (on page 536) into stylesheets.

---

**Note:**

For XSL and XSD resources, the **Content Completion Assistant** collects its components starting from the **main files** (on page 3220). The **main files** can be defined in the project or in the associated validation scenario. For further details about the **Main Files** support go to **Defining Main Files at Project Level** (on page 418).

---

The extension functions included in the Saxon 6.5.5 and 10.6 transformation engines are presented in the content completion list only if the Saxon namespace (**http://saxon.sf.net** for XSLT version 2.0 / 3.0 or **http://icl.com/saxon** for XSLT version 1.0) is declared and one of the following conditions is true:
• The edited file has a transformation scenario that uses as transformation engine Saxon 6.5.5 (for XSLT version 1.0), Saxon 10.6 PE or Saxon 10.6 EE (for XSLT version 2.0 / 3.0).
• The edited file has a validation scenario that uses as validation engine Saxon 6.5.5 (for version 1.0), Saxon 10.6 PE or Saxon 10.6 EE (for version 2.0 / 3.0).
• The validation engine specified in Options (on page 245) page is Saxon 6.5.5 (for version 1.0), Saxon 10.6 PE or Saxon 10.6 EE (for version 2.0 / 3.0).

Additionally, the Saxon-CE-specific extension functions and instructions are presented in the list of content completion assistance proposals only if the http://saxonica.com/ns/interactiveXSLT namespace is declared.

Namespace prefixes in the scope of the current context are presented at the top of the content completion assistance window to speed up the insertion into the document of prefixed elements.

Figure 258. Namespace Prefixes in the Content Completion Assistant

For the common namespaces such as XSL namespace (http://www.w3.org/1999/XSL/Transform), XML Schema namespace (http://www.w3.org/2001/XMLSchema), or Saxon namespace (http://icl.com/saxon for version 1.0, http://saxon.sf.net/ for version 2.0 / 3.0), Oxygen XML Editor provides an easy mode to declare them by proposing a prefix for these namespaces.

Content Completion in XPath Expressions

In XSLT stylesheets, the Content Completion Assistant (on page 3217) provides all the features available in the XML editor (on page 532) and also adds some enhancements. In XPath expressions used in attributes of XSLT stylesheets (such as @match, @select, and @test), the Content Completion Assistant offers the names of XPath and XSLT functions, XSLT axes, and user-defined functions (the name of the function and its parameters). If a transformation scenario was defined and associated to the edited stylesheet, the Content Completion Assistant computes and presents elements and attributes based on:

• The input XML document selected in the scenario.
• The current context in the stylesheet.

The associated document is displayed in the XSLT/XQuery Input view (on page 904).

Content completion for XPath expressions is started:
• On XPath operators detected in one of the `@match`, `@select`, and `@test` attributes of XSLT elements: `*`, `+`, `|`, `/`, `//`, `@`, `;`, `$`.

• For attribute value templates of non-XSLT elements, that is the `|` character when detected as the first character of the attribute value.

• On request, if the combination `Ctrl + Space (Command + Space on macOS)` is pressed inside an edited XPath expression.

The proposals presented in the `Content Completion Assistant` are dependent on:

• The context of the current XSLT element.
• The XML document associated with the edited stylesheet in the stylesheet transformation scenario.
• The XSLT version of the stylesheet (1.0, 2.0, or 3.0).

**Note:**
The XSLT 3.0 content completion list of proposals includes specific elements and attributes for the 3.0 version.

For example, if the document associated with the edited stylesheet is:

```xml
<personnel>
  <person id="Big.Boss">
    <name>
      <family>Boss</family>
      <given>Big</given>
    </name>
    <email>chief@oxygenxml.com</email>
    <link subordinates="one.worker"/>
  </person>
  <person id="one.worker">
    <name>
      <family>Worker</family>
      <given>One</given>
    </name>
    <email>one@oxygenxml.com</email>
    <link manager="Big.Boss"/>
  </person>
</personnel>
```

If you enter an `xsl:template` element using the `Content Completion Assistant`, the following actions are triggered:
• The `@match` attribute is inserted automatically.
• The cursor is placed between the quotes.
• The XPath Content Completion Assistant automatically displays a pop-up window with all the XSLT axes, XPath functions and elements and attributes from the XML input document that can be inserted in the current context.

The set of XPath functions depends on the XSLT version declared in the root element `xsl:stylesheet`: 1.0, 2.0, or 3.0. Functions from other namespaces, such as maps, arrays, and math, are presented only if the namespaces are declared.

Figure 259. Content Completion in the `@match` Attribute

![Figure 259](image1)

If the cursor is inside the `@select` attribute of an `<xsl:for-each>`, `<xsl:apply-templates>`, `<xsl:value-of>` or `<xsl:copy-of>` element the content completion proposals depend on the path obtained by concatenating the XPath expressions of the parent XSLT elements `<xsl:template>` and `<xsl:for-each>` as shown in the following figure:

Figure 260. Content Completion in the `@select` Attribute

![Figure 260](image2)

Also XPath expressions typed in the `@test` attribute of an `<xsl:if>` or `<xsl:when>` element benefit of the assistance of the content completion.
XSLT variable references are easier to insert in XPath expressions with the help of the content completion pop-up triggered by the `$` character, which signals the start of such a reference in an XPath expression.

The time delay (configured in the Content Completion preferences page (on page 213)) is also applied for the content completion in XPath expressions.
Tooltip Helper for the XPath Functions Arguments

When editing the arguments of an XPath/XSLT function, Oxygen XML Editor tracks the current entered argument by displaying a tooltip containing the function signature. The currently edited argument is highlighted with a bolder font.

When moving the cursor through the expression, the tooltip is updated to reflect the argument found at the cursor position.

**Examples:**

If you want to concatenate the absolute values of two variables, named $v1$ and $v2$:

```xml
<xsl:template match="/">
  <xsl:value-of select="concat(abs($v1), abs($v2))"></xsl:value-of>
</xsl:template>
```

When moving the cursor before the first `abs` function, Oxygen XML Editor identifies it as the first argument of the `concat` function. The tooltip shows in bold font the following information about the first argument:

- Its name is $arg1.
- Its type is `xdt:anyAtomicType`.
- It is optional (note the `?` sign after the argument type).

The function also takes other arguments that have the same type and returns a `xs:string`.

![Figure 264. XPath Tooltip Helper - Identify the `concat` Function's First Argument](image)

Moving the cursor on the first variable $v1$, the editor identifies the `abs` as context function and shows its signature:

![Figure 265. XPath Tooltip Helper - Identify the `abs` Function's Argument](image)

Further, clicking the second `abs` function name, the editor detects that it represents the second argument of the `concat` function. The tooltip is repainted to display the second argument in bold font.
Figure 266. XPath Tooltip Helper - Identify the `concat` Function's Second Argument

```
match := concat!($arg1 as xdt:atomicType?, $arg2 as xdt:atomicType?, ...) as xs:string

elem := concat(abs($v1), abs($v2))
</xsl:value-of>
</xsl:template>
```

Note:
The tooltip helper is also available in the XPath Builder view (on page 2002) and XPath toolbar (on page 2000).

Related Information:
- Working with XPath Expressions (on page 1999)

Syntax Highlighting in XSLT

Oxygen XML Editor supports syntax highlighting in Text mode to make it easier to read the semantics of the structured content by displaying each type of code in different colors and fonts.

To customize the colors or styles used for the syntax highlighting colors for XSLT files, follow these steps:

1. Open the Preferences dialog box (Options > Preferences) (on page 124).
2. Go to Editor > Syntax Highlight (on page 224).
3. Select and expand the XML section in the top pane.
4. Select the component you want to change and customize the colors or styles using the selectors to the right of the pane.
5. Select the XSL tab in the Preview pane to see the effects of your changes.

Tip:
Oxygen XML Editor also allows you to specify syntax highlighting colors for specific XML elements and attributes with specific namespace prefixes. This can be done in the Editor > Syntax Highlight > Elements/Attributes by Prefix preferences page (on page 225).

Related Information:
- Customize Syntax Highlight colors (on page 224)

XSLT Outline View

The Outline view for XSLT stylesheets displays the list of all the components (templates, attribute-sets, character-maps, variables, functions, keys, outputs) from both the edited stylesheet and its imports or includes. For XSL and XSD resources, the Outline view collects its components starting from the main files (on
The main files can be defined in the project or in the associated validation scenario. For further details about the Main Files support go to Defining Main Files at Project Level (on page 418).

By default, it is displayed on the left side of the editor. If the view is not displayed, it can be opened by selecting it from the Window > Show View menu.

Figure 267. XSLT Outline View

The following actions are available in the Settings menu on the Outline view toolbar:

**Filter returns exact matches**

The text filter of the Outline view returns only exact matches;

**Selection update on cursor move**

Controls the synchronization between Outline view and source document. The selection in the Outline view can be synchronized with the cursor moves or the changes in the XSLT editor. Selecting one of the components from the Outline view also selects the corresponding item in the source document.

When the Show components option is selected, the following actions are available:

**Show XML structure**

Displays the XML document structure in a tree-like structure.

**Show all components**

Displays all components that were collected starting from the main file (on page 3220). This option is set by default.
Show only local components
Displays the components defined in the current file only.

Group by location/type
The stylesheet components can be grouped by location and type.

When the Show XML structure option is selected, the following actions are available:

- Show components
  Switches the Outline view to the components display mode.

- Flat presentation mode of the filtered results
  When active, the application flattens the filtered result elements to a single level.

- Show comments and processing instructions
  Show/hide comments and processing instructions in the Outline view.

- Show element name
  Show/hide element name.

- Show text
  Show/hide additional text content for the displayed elements.

- Show attributes
  Show/hide attribute values for the displayed elements. The displayed attribute values can be changed from the Outline preferences panel (on page 306).

- Configure displayed attributes
  Displays the XML Structured Outline preferences page (on page 306).

The following contextual menu actions are also available when the Show components option is selected in the Settings menu:

- Edit Attributes
  Opens a small in-place editor that allows you to edit the attributes of the selected node.

- Cut
  Cuts the currently selected node.

- Copy
  Copies the currently selected node.

- Delete
  Deletes the currently selected node.

- Search References Ctrl + Shift + R (Command + Shift + R on macOS)
Searches all references of the item found at current cursor position in the defined scope, if any.
See Finding XSLT References and Declarations (on page 911) for more details.

**Search References in**

Searches all references of the item found at current cursor position in the specified scope. See Finding XSLT References and Declarations (on page 911) for more details.

**Component Dependencies**

Opens the Component Dependencies view (on page 909) that allows you to see the dependencies for the currently selected component.

**Resource Hierarchy**

Opens the Resource Hierarchy/Dependencies view (on page 906) that displays the hierarchy for the currently selected resource.

**Resource Dependencies**

Opens the Resource Hierarchy/Dependencies view (on page 906) that displays the dependencies of the currently selected resource.

**Rename Component in**

Renames the selected component. See XSLT Refactoring Actions (on page 915) for more details.

The following contextual menu actions are available in the Outline view when the Show XML structure option is selected in the Settings menu:

**Append Child**

Displays a list of elements that you can insert as children of the current element.

**Insert Before**

Displays a list of elements that you can insert as siblings of the current element, before the current element.

**Insert After**

Displays a list of elements that you can insert as siblings of the current element, after the current element.

**Edit Attributes**

Opens a small in-place editor that allows you to edit the attributes of the selected node.

**Toggle Comment**

Comments/uncomments the currently selected element.

**Search references**

Searches for the references of the currently selected component.

**Search references in**
Searches for the references of the currently selected component in the context of a scope that you define.

Component dependencies

Opens the Component Dependencies view (on page 909) that displays the dependencies of the currently selected component.

Rename Component in

Renames the currently selected component in the context of a scope that you define.

Cut

Cuts the currently selected component.

Copy

Copies the currently selected component.

Delete

Deletes the currently selected component.

Expand More

Expands the structure of a component in the Outline view.

Collapse All

Collapses the structure of all the component in the Outline view.

The stylesheet components information is presented on two columns: the first column presents the @name and @match attributes, the second column the @mode attribute. If you know the component name, match or mode, you can search it in the Outline view by typing one of these pieces of information in the filter text field from the top of the view or directly on the tree structure. When you type de component name, match or mode in the text field, you can switch to the tree structure using:

- Keyboard arrow keys
- Enter key
- Tab key
- Shift-Tab key combination

To switch from tree structure to the filter text field, you can use Tab and Shift-Tab.

Tip:

The search filter is case insensitive. The following wildcards are accepted:

- * - any string
- ? - any character
- , - patterns separator
If no wildcards are specified, the string to search is used as a partial match.

The content of the **Outline** view and the editing area are synchronized. When you select a component in the **Outline** view, its definition is highlighted in the editing area.

Oxygen XML Editor allows you to sort the components of the tree in the **Outline** view.

**Note:**

Sorting groups in the **Outline** view is not supported.

Oxygen XML Editor has a predefined order of the groups in the **Outline** view:

- For location, the names of the files are sorted alphabetically. The file you are editing is located at the top of the list.
- For type, the order is: parameters, variables, templates, functions, set attributes, character-map.

**Note:**

When no grouping is available and the table is not sorted, Oxygen XML Editor sorts the components depending on their order in the document. Oxygen XML Editor also takes into account the name of the file that the components are part of.

**XSLT Input View**

The structure of the XML document associated to the edited XSLT stylesheet is displayed in a tree form in a view called the **XSLT Input** view. If the view is not displayed, it can be opened by selecting it from the **Window > Show View** menu. The tree nodes represent the elements of the documents.

If you click a node in the **XSLT Input** view, the corresponding template from the stylesheet is highlighted. A node can be dragged from this view and dropped in the editor area for quickly inserting `<xsl:template>`, `<xsl:for-each>`, or other XSLT elements that have the `@match`, `@select`, or `@test` attribute already completed. The value of the attribute is the correct XPath expression that refers to the dragged tree node. This value is based on the current editing context of the drop spot.
Example:

For the following XML document:

```xml
<personnel>
  <person id="Big.Boss">
    <name>
      <family>Boss</family>
      <given>Big</given>
    </name>
    <email>chief@oxygenxml.com</email>
    <link subordinates="one.worker"/>
  </person>
  <person id="one.worker">
    <name>
      <family>Worker</family>
      <given>One</given>
    </name>
    <email>one@oxygenxml.com</email>
    <link manager="Big.Boss"/>
  </person>
</personnel>
```

and the following XSLT stylesheet:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  version="2.0">  
<xsl:template match="personnel">
  <xsl:for-each select="*"/>
</xsl:template>
</xsl:stylesheet>
```
if you drag the `<given>` element and drop it inside the `<xsl:for-each>` element, the following pop-up menu is displayed:

```
<xml version="1.0" encoding="UTF-8"/>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="2.0">
  <xsl:template match="personnel">
    <xsl:for-each select="*">
      <xsl:copy-of select="name/given"/>
    </xsl:for-each>
  </xsl:template>
</xsl:stylesheet>
```

if you select **Insert xsl:copy-of** (for example), the resulting document will look like this:

```
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="2.0">
  <xsl:template match="personnel">
    <xsl:for-each select="*">
      <xsl:copy-of select="name/given"/>
    </xsl:for-each>
  </xsl:template>
</xsl:stylesheet>
```

XSLT Resource Hierarchy/Dependencies View

The **Resource Hierarchy/Dependencies** view displays the hierarchy or dependencies for resources included in a stylesheet. If the view is not displayed, it can be opened by selecting it from the **Window > Show View** menu.

If you want to see the hierarchy or dependencies of a stylesheet, select the desired stylesheet in the **Project view (on page 402)** and choose **Resource Hierarchy** or **Resource Dependencies** from the contextual menu.
If you want to see the dependencies of a stylesheet, select the desired stylesheet in the Project view (on page 402) and choose Resource Dependencies from the contextual menu.

The following actions are available on the toolbar of the Resource Hierarchy/Dependencies view:

- **Refresh**
  Refreshes the Hierarchy/Dependencies structure.

- **Stop**
  Stops the hierarchy/dependencies computing.

- **Show Hierarchy**
  Allows you to choose a resource to compute the hierarchy structure.

- **Show Dependencies**
  Allows you to choose a resource to compute the dependencies structure.

- **Configure dependencies search scope**
  Allows you to configure a scope to compute the dependencies structure. There is also an option for automatically using the defined scope for future operations.

- **History**
  Provides access to the list of previously computed dependencies. Use the Clear history button to remove all items from this list.
The contextual menu for a resource listed in the Resource Hierarchy/Dependencies view contains the following actions:

**Open**

Opens the resource. You can also double-click a resource in the Hierarchy/Dependencies structure to open it.

**Go to reference**

Opens the source document where the resource is referenced.

**Copy location**

Copies the location of the resource.

**Move resource**

Moves the selected resource.

**Rename resource**

Renames the selected resource.

**Resource Hierarchy**

Shows the hierarchy for the selected resource.

**Resource Dependencies**

Shows the dependencies for the selected resource.

**Add to Main Files**

Adds the currently selected resource in the Main Files directory (on page 418).

**Expand More**

Expands more of the children of the selected resource from the Hierarchy/Dependencies structure.

**Collapse All**

Collapses all children of the selected resource from the Hierarchy/Dependencies structure.

**Tip:**

When a recursive reference is encountered in the Hierarchy view, the reference is marked with a special icon 🚫.

Related Information:

- Modular Contextual XML Editing Using 'Main Files' Support (on page 829)
- Search and Refactor Operations Scope (on page 831)
Moving/Renaming XSLT Resources

You can move and rename a resource presented in the Resource/Hierarchy Dependencies view, using the Rename resource and Move resource refactoring actions from the contextual menu.

When you select the Rename action in the contextual menu of the Resource/Hierarchy Dependencies view, the Rename resource dialog box is displayed. The following fields are available:

- **New name** - Presents the current name of the edited resource and allows you to modify it.
- **Update references of the renamed resource(s)** - Select this option to update the references to the resource you are renaming. A Preview option is available that allows you to see what will be updated before selecting Rename to process the operation.

When you select the Move action from the contextual menu of the Resource/Hierarchy Dependencies view, the Move resource dialog box is displayed. The following fields are available:

- **Destination** - Presents the path to the current location of the resource you want to move and gives you the option to introduce a new location.
- **New name** - Presents the current name of the moved resource and gives you the option to change it.
- **Update references of the moved resource(s)** - Select this option to update the references to the resource you are moving, in accordance with the new location and name. A Preview option is available that allows you to see what will be updated before selecting Move to process the operation.

XSLT Component Dependencies View

The Component Dependencies view allows you to see the dependencies for a selected component. This is helpful if you want to see where components are used in the entire hierarchy. For example, if you want to find all the references where a given component is used.

If the view is not displayed, it can be opened by selecting it from the Window > Show View menu.

To see the dependencies of an XSLT component:

1. Right-click the desired component in the editor or Outline view.
2. Select the Component Dependencies action from the contextual menu.

The action is available for all named components (templates, variables, parameters, attribute sets, keys, functions, outputs).

If a component contains multiple references, a small table is displayed at the bottom of the view that contains all the references. When a recursive reference is encountered, it is marked with a special icon ©.
The Component Dependencies view includes the following toolbar actions:

- **Refresh**
  
  Refreshes the dependencies structure.

- **Stop**
  
  Stops the dependency computation.

- **Configure**
  
  Allows you to choose the search scope for computing the dependencies structure. This is helpful for making sure all imported/included resources are computed.

- **History**
  
  Allows you to select from a list of the most recently used dependency computations.

In addition, the following actions are available in the contextual menu:

- **Go to First Reference**
  
  Selects the first reference of the currently selected component in the dependencies tree.

- **Go to Component**
  
  Shows the definition of the currently selected component in the dependencies tree.
Highlight Component Occurrences

When a component (for example variable or named template) is found at current cursor position, Oxygen XML Editor performs a search over the entire document to find the component declaration and all its references. When found, they are highlighted both in the document and in the stripe bar, at the right side of the document.

Note:
Oxygen XML Editor also supports occurrences highlight for template modes.

Customizable colors are used: one for the component definition and another one for component references. Occurrences are displayed until another component is selected and a new search is performed. All occurrences are removed when you start to edit the document.

This feature is enabled by default. To configure it, open the Preferences dialog box (Options > Preferences) (on page 124) and go to Editor > Mark Occurrences. A search can also be triggered with the Search > Search Occurrences in File (Ctrl + Shift + U (Command + Shift + U on macOS)) contextual menu action. Matches are displayed in separate tabs of the Results view (on page 548).

Finding XSLT References and Declarations

The following search actions related with XSLT references and declarations are available from the Search submenu of the contextual menu and from the Document > References menu:

**Search References**

Searches all references of the item found at current cursor position in the defined scope, if any. If a scope is defined but the currently edited resource is not part of the range of determined resources, a warning dialog box is displayed that allows you to define another search scope.

**Search References in**

Searches all references of the item found at current cursor position in the file or files that you specify when a scope is defined.

**Search Declarations**

Searches all declarations of the item found at current cursor position in the defined scope, if any. If a scope is defined but the currently edited resource is not part of the range of resources determined by this scope, a warning dialog box is displayed that allows you to define another search scope.

**Search Declarations in**

Searches all declarations of the item found at current cursor position in the file or files that you specify when a scope is defined.
Search Occurrences in File

Searches all occurrences of the item at the cursor position in the currently edited file.

The following action is available from the contextual menu and the Document > Schema menu:

- **Go to Definition**

  Moves the cursor to the location of the definition of the current item.

**Note:**

You can also use the **Ctrl + Single-Click (Command + Single-Click on macOS)** shortcut on a reference to display its definition.

Related Information:

- Search and Refactor Operations Scope (on page 831)

### XSLT Stylesheet Component Documentation Support

Oxygen XML Editor offers built-in support for documenting XSLT stylesheets. If the expanded QName (on page 3223) of the element has a non-null namespace URI, the `<xsl:stylesheet>` element may contain any element not from the XSLT namespace. Such elements are referenced as user-defined data elements. Such elements can contain the documentation for the stylesheet and its elements (top-level elements whose names are in the XSLT namespace). Oxygen XML Editor offers its own XML schema that defines such documentation elements. The schema is named `stylesheet_documentation.xsd` and can be found in `[OXYGEN_INSTALL_DIR]/frameworks/stylesheet_documentation`. The user can also specify a custom schema in XSL Content Completion options (on page 213).

### Content Completion

When content completion is invoked inside an XSLT editor by pressing **Ctrl + Space (Command + Space on macOS)**, it offers elements from the XSLT documentation schema (either the built-in one or one specified by user).

### Adding Documentation Blocks

In **Text** mode, to add documentation blocks, press **Ctrl + Alt + D (Command + Alt + D on macOS)** or select **Add component documentation** from the contextual menu.

In **Author** mode, the following stylesheet documentation actions are available in the contextual menu, Component Documentation submenu:

- **Add component documentation** - Adds documentation blocks for the component at the cursor position.
- **Paragraph** - Inserts a new documentation paragraph.
- **Bold** - Makes the selected documentation text bold.
- **Italic** - Makes the selected documentation text italic.
• **List** - Inserts a new list.
• **List Item** - Inserts a list item.
• **Reference** - Inserts a documentation reference.

If the cursor is positioned inside the `<xsl:stylesheet>` element context, documentation blocks are generated for all XSLT elements. If the cursor is positioned inside a specific XSLT element (such as a template or function), a documentation block is generated for that element only.

**Example: Documentation Block Using Oxygen XML Editor Built-in Schema**

```xml
<xd:doc>
  <xd:desc>
    <xd:p>
      Search inside parameter `<xd:i>string</xd:i>` for the last occurrence of parameter `<xd:i>searched</xd:i>`. The substring starting from the 0 position to the identified last occurrence will be returned.
    </xd:p>
    <xd:ref name="f:substring-after-last" type="function">
      xmlns:f="http://www.oxygenxml.com/doc/xsl/functions">
      See also
    </xd:ref>
  </xd:desc>
  <xd:param name="string">
    <xd:p>
      String to be analyzed
    </xd:p>
  </xd:param>
  <xd:param name="searched">
    <xd:p>
      Marker string. Its last occurrence will be identified
    </xd:p>
  </xd:param>
  <xd:return>
    <xd:p>
      A substring starting from the beginning of `<xd:i>string</xd:i>` to the last occurrence of `<xd:i>searched</xd:i>`. If no occurrence is found an empty string will be returned.
    </xd:p>
  </xd:return>
</xd:doc>
```

**XSLT Documentation Links**

Oxygen XML Editor includes support for links inside XSLT documentation blocks. Using a construct like `<xd:a docid="user-defined-id">TEXT</xd:a>` will cause the browser to scroll to the particular anchor (the defined ID) in the current document. Using a construct like `<xd:a href="http://www.my-web-site">TEXT</xd:a>` or `<xd:a href="local-file-path/filename">TEXT</xd:a>` will open the referenced link in a new tab.

**Example: Documentation Links**
XSLT 3.0 Text Value Templates

Oxygen XML Editor offers built-in support for XSLT 3.0 Text Value Templates, including content completion to present the variables, functions, and parameters from the current context and syntax highlighting.

A text node in the stylesheet is treated as a text value template if the following things are true:

- It is part of a sequence constructor or a child of an <xsl:text> instruction.
- There is an ancestor element with an @xsl:expand-text attribute and on the innermost ancestor element that has such an attribute, the value of the attribute is yes.

Example:

```xml
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    xmlns:xs="http://www.w3.org/2001/XMLSchema"
    expand-text="yes"
    version="3.0">

    <xsl:param name="seq" as="xs:string?" select="'c', 'a', 'b', 'z'"/>
    <xsl:template name="main">
        {sort($seq)}
    </xsl:template>
</xsl:stylesheet>
```

For more information, see: W3C XSLT Specifications: Text Value Templates.

Related Information:

Content Completion in XPath Expressions (on page 894)
XSLT 3.0 Packages (xsl:package Element)

Oxygen XML Editor offers built-in support for XSLT 3.0 Packages (xsl:package element). This element defines a set of stylesheet modules that can be compiled as a unit, independently of other packages.

Oxygen XML Editor includes a new document template called XSLT Package to help you easily create an XSLT 3.0 file with the xsl:package element set as the document’s root element. It is available when creating new documents from templates (on page 368) and can be found in the New Document folder or by typing xslt package in the search field.

Also, when editing XSLT 3.0 documents, the xsl:package element is offered as one of the proposals for the document’s root element in the content completion window.

XSLT Refactoring Actions

Oxygen XML Editor offers a set of actions that allow you to change the structure of an XSLT stylesheet without changing the results of running it in an XSLT transformation. Depending on the selected text, the following XSLT refactoring actions are available from the Refactoring submenu of the contextual menu (or from the Document > Refactoring menu):

- Extract template (Active only when the selection contains well-formed elements)

  Extracts the selected XSLT instructions sequence into a new template. It opens a dialog box that allows you to specify the name of the new template to be created. The possible changes to perform on the document can be previewed before altering the document. After pressing OK, the template is created and the selection is replaced with the xsl:call-template instruction referencing the newly created template.

  **Note:**
  The newly created template is indented and its name is highlighted in the xsl:call-template element.

- Extract function

  Extracts the selected XSLT instructions sequence into a new function. It opens a dialog box that allows you to specify the name of the new function. It then moves the selected lines to a newly created XSLT function and inserts a function call in the place of the selected lines. You can also use parts of an XPath expression to create the new functions.

- Create local variable

  Creates an XSLT variable, wrapped around the selection. It opens a dialog box that allows you to specify the name of the new variable. It then wraps the selection in the variable and you can reference it at anytime in the code.

- Move to another stylesheet (Active only when entire components are selected)
Allows you to move one or more XSLT global components (templates, functions, or parameters) to another stylesheet. It opens a dialog box that allows you to specify where the selected components will be moved to. Follow these steps when using the dialog box:

1. Choose whether you want to move the selected components to a new stylesheet or an existing one.
2. If you choose to move the components to an existing one, select the destination stylesheet. Click the **Choose** button to select the destination stylesheet file. Oxygen XML Editor will automatically check if the destination stylesheet is already contained by the hierarchy of the current stylesheet. If it is not contained, choose whether or not the destination stylesheet will be referenced (imported or included) from the current stylesheet. The following options are available:
   - **Include** - The current stylesheet will use an `<xsl:include>` instruction to reference the destination stylesheet.
   - **Import** - The current stylesheet will use an `<xsl:import>` instruction to reference the destination stylesheet.
   - **None** - There will be created no relation between the current and destination stylesheets.
3. Click the **Move** button to move the components to the destination. The moved components are highlighted in the destination stylesheet.

**Convert attributes to xsl:attributes**

Converts the attributes from the selected element and represents each of them with an `<xsl:attribute>` instruction. For example, the following element:

```xml
<person id="Big{test}Boss"/>
```

is converted to:

```xml
<person>
  <xsl:attribute name="id">
    <xsl:text>Big</xsl:text>
    <xsl:value-of select="test"/>
    <xsl:text>Boss</xsl:text>
  </xsl:attribute>
</person>
```

**Convert xsl:attributes to attributes**

Converts `<xsl:attribute>` elements to inline attributes for elements outside the XSL namespace. For example, the following element: It is the reverse of the **Convert attributes to xsl:attributes** action with the following limitations:
• The `xsl:attribute` element is "text only".
• The `xsl:attribute` element has a single `xsl:text` child element.
• The `xsl:attribute` element has a single `xsl:value-of` child element. In this case, the value of the attribute will be the XPath expression from the `@select` attribute surrounded by curly brackets (text value template).

```
<person>
  <xsl:attribute name="id">john.doe</xsl:attribute>
  <xsl:attribute name="email">john.doe@example.com</xsl:attribute>
  <xsl:attribute name="manager"><xsl:value-of select="person[@id='boss']/name"/></xsl:attribute>
</person>
```

is converted to:

```
<person id="john.doe" email="john.doe@example.com" manager="{person[@id='boss']/name}"/>
```

**Convert xsl:if into xsl:choose/xsl:when**

Converts one or more `xsl:if` element blocks into one or more `xsl:when` blocks surrounded by an `xsl:choose` element. If it is invoked on a selection, the selection must contain a well-formed fragment. If there is no selection, the `xsl:if` element that surrounds the content at the current cursor position is converted.

For example, the following block:

```
<xsl:if test="a">
  <!-- XSLT code -->
</xsl:if>
```

is converted to:

```
<xsl:choose>
  <xsl:when test="a">
    <!-- XSLT code -->
  </xsl:when>
  <xsl:otherwise>
    <!-- XSLT code -->
  </xsl:otherwise>
</xsl:choose>
```

where the `|` character is the current cursor position.

**Convert xsl:choose/xsl:when into xsl:if**
Converts each `<xsl:when>` block into an `<xsl:if>` block. For the `<xsl:otherwise>` branch, it also adds an `and` statement to each negated form of the conditions. For example, the following block:

```xml
<xsl:choose>
  <xsl:when test="c1">
    <!-- XSLT statement 1 -->
  </xsl:when>
  <xsl:when test="c2">
    <!-- XSLT statement 2 -->
  </xsl:when>
  <xsl:when test="c3">
    <!-- XSLT statement 3 -->
  </xsl:when>
  <xsl:otherwise>
    <!-- XSLT "otherwise" statement-->
  </xsl:otherwise>
</xsl:choose>
```

is converted to:

```xml
<xsl:if test="c1">
  <!-- XSLT statement 1 -->
</xsl:if>
<xsl:if test="c2">
  <!-- XSLT statement 2 -->
</xsl:if>
<xsl:if test="c3">
  <!-- XSLT statement 3 -->
</xsl:if>
<xsl:if test="not(c1) and not(c2) and not(c3)">
  <!-- XSLT "otherwise" statement-->
</xsl:if>
```

**Extract local variable (Active on a selection made inside an attribute that contains an XPath expression)**

Allows you to create a new local variable by extracting the selected XPath expression. After creating the new local variable before the current element, Oxygen XML Editor allows you to edit the name of the variable.

**Extract global variable (Active on a selection made inside an attribute that contains an XPath expression)**

Allows you to create a new global variable by extracting the selected XPath expression. After creating the new global variable, Oxygen XML Editor allows you to edit the name of the variable.
Extract template parameter (Active on a selection made inside an attribute that contains an XPath expression)

Allows you to create a new template parameter by extracting the selected XPath expression. After creating the new parameter, Oxygen XML Editor allows you to edit the name of the parameter.

Extract global parameter (Active on a selection made inside an attribute that contains an XPath expression)

Allows you to create a new global parameter by extracting the selected XPath expression. After creating the new parameter, Oxygen XML Editor allows you to edit the name of the parameter.

Note:
Oxygen XML Editor checks if the selected expression depends on local variables or parameters that are not available in the global context where the new parameter is created.

Rename Component

Allows you to rename the current component (in-place). The component and all its references in the document are highlighted with a thin border and the changes you make to the component at the cursor position are updated in real time to all occurrences of the component. To exit the in-place editing, press the Esc or Enter key on your keyboard.

Rename Component in

Opens a dialog box that allows you to rename the selected component by specifying the new component name and the files to be affected by the modification. If you click the Preview button, you can view the files to be affected by the action.
Note:

Many of these refactoring actions are also proposed by the Quick Assist support (on page 920).

For more information about XSLT refactoring, watch our video demonstration:

https://www.youtube.com/embed/4ir5XWyp8Zo

**XSLT Quick Assist Support**

The Quick Assist support (on page 3223) helps you to rapidly access search and refactoring actions. If one or more actions are available in the current context, they are accessible via a yellow bulb help (💡) placed at the current line in the stripe on the left side of the editor. Also, you can invoke the Quick Assist menu by using the **Alt + 1 (Meta + Alt + 1)** on macOS) keyboard shortcuts.

Two categories of actions are available in the Quick Assist menu:

- Actions available on a selection made inside an attribute that contains an XPath expression:
  
  **Extract template**
  
  Extracts the selected XSLT instructions sequence into a new template.
Move to another stylesheet

Allows you to move one or more XSLT global components (templates, functions, or parameters) to another stylesheet.

Extract local variable

Allows you to create a new local variable by extracting the selected XPath expression.

Extract global variable

Allows you to create a new global variable by extracting the selected XPath expression.

Extract template parameter

Allows you to create a new template parameter by extracting the selected XPath expression.

Extract global parameter

Allows you to create a new global parameter by extracting the selected XPath expression.

Figure 272. XSLT Quick Assist Support - Refactoring Actions

- Actions available when the cursor is positioned over the name of a component:

Rename Component

Renames the component and all its dependencies.

Search Declarations

Searches the declaration of the component in a predefined scope. It is available only when the context represents a component name reference.

Search References

Searches all references of the component in a predefined scope.

Component Dependencies

Searches the component dependencies in a predefined scope.

Change Scope

Configures the scope that will be used for future search or refactor operations.

Rename Component
Allows you to rename the current component in-place.

**Search Occurrences**

Searches all occurrences of the component within the current file.

**Figure 273. XSLT Quick Assist Support - Component Actions**

![XSLT Quick Assist Support](image)

Related Information:
- Component Dependencies View *(on page 909)*
- XSLT Hierarchy View *(on page 906)*
- XSLT Refactoring Actions *(on page 915)*
- Search and Refactor Operations Scope *(on page 831)*

**XSLT Unit Test (XSpec)**

XSpec is a behavior driven development (BDD) framework for XSLT, XQuery, and Schematron. XSpec consists of syntax for describing the behavior of your XSLT, XQuery, or Schematron code, and some code that enables you to test your code against those descriptions.

**Creating an XSLT Unit Test**

To create an XSLT Unit Test, go to **File > New > XSLT Unit Test**. You can also create an XSLT Unit Test from the contextual menu of an XSL file in the **Project view (on page 402)**. Oxygen XML Editor allows you to customize the XSpec document when you create it. In the customization dialog box, you can enter the path to an XSL document or to a main XSL document.

When you create an XSpec document based on an XSL document, Oxygen XML Editor uses information from the validation and transformation scenarios associated with the XSL file. From the transformation scenario, Oxygen XML Editor uses extensions and properties of Saxon 10.6, improving the Ant scenario associated with the XSpec document.
Running an XSLT Unit Test

To run a Unit Test, open the XSpec file in an editor and click
Apply Transformation Scenario(s) on the main toolbar. This will run the built-in Run XSpec Test transformation scenario that is defined in the XSpec framework (on page 3219).

Testing a Stylesheet

An XSpec file contains one or more test scenarios. You can test a stylesheet in one of the following ways:

• **Test an entire stylesheet** - Testing is performed in a certain context. You can define a context as follows:

  ° Inline context, building the test based on a string.

  ```xml
  <x:scenario label="when processing a para element">
    <x:context>
  ```
Based on an external file, or on a part of an external file extracted with an XPath expression.

You can reference test files between each other, which allows you to define a suite of tests. For further details about test scenarios, go to https://github.com/xspec/xspec/wiki/Writing-Scenarios.

Adding a Catalog to an XSpec Transformation

If your XSLT needs a catalog, you can add one to the XSpec transformation by doing one of the following:

- If you are using a project (on page 398) in Oxygen XML Editor, create a catalog.xml file in the project directory. This catalog will then be loaded automatically.
- Edit (on page 1545) the Run XSpec Test transformation scenario, go to the Parameters tab (on page 1509), and set the value of the catalog parameter to the location of your catalog file.
Generating Documentation for an XSLT Stylesheet

You can use Oxygen XML Editor to generate detailed documentation in HTML format for the elements (top-level elements whose names are in the XSLT namespace) of an XSLT stylesheet. You can select what XSLT elements to include in the generated documentation and also the level of details to present for each of them. The elements are hyperlinked. To generate documentation in a custom output format (on page 931), you can edit the XSLT stylesheet used to generate the documentation, or create your own stylesheet.

To open the XSLT Stylesheet Documentation dialog box, select XSLT Stylesheet Documentation from the Tools > Generate Documentation menu or from the Generate Documentation submenu in the contextual menu of the Project view (on page 402). You can also open the tool by using the Generate Documentation toolbar button.

![Figure 275. XSLT Stylesheet Documentation Dialog Box](image)

The XSL URL field of the dialog box must contain the full path to the XSL Stylesheet file you want to generate documentation for. The stylesheet may be a local or a remote file. You can specify the path to the stylesheet by entering it in the text field, or by using the Insert Editor Variables button or the options in the Browse drop-down menu.

**Output Tab**

The following options are available in the Output tab:
• **Format** - Allows you to choose between the following formats:
  ◦ **HTML** - The documentation is generated in [HTML output format](#) on page 929).
  ◦ **Custom** - The documentation is generated in a [custom output format](#) on page 931), allowing you to control the output. Click the **Options** button to open a [Custom format options dialog box](#) where you can specify a custom stylesheet for creating the output. There is also an option to **Copy additional resources to the output folder** and you can select the path to the additional Resources that you want to copy. You can also choose to keep the intermediate XML files created during the documentation process by deselecting the **Delete intermediate XML file** option.

• **Output file** - You can specify the path of the output file by entering it in the text field, or by using the [Insert Editor Variables](#) button or the options in the **Browse** drop-down menu.

• **Split output into multiple files** - Instructs the application to split the output into multiple files. For large XSLT stylesheets, choosing another split criterion may generate smaller output files, providing faster documentation browsing. You can choose to split them by namespace, location, or component name.

• **Open in Browser/System Application** - Opens the result in the system application associated with the output file type.

**Note:**
To set the browser or system application that will be used, open the **Preferences** dialog box (Options > Preferences) on page 124), go to **Global**, and set it in the [Default Internet browser](#) field. This will take precedence over the default system application settings.

**Settings Tab**
When you generate documentation for an XSLT stylesheet you can choose what XSLT elements to include in the output (templates, functions, global parameters, global variables, attribute sets, character maps, keys, decimal formats, output formats, XSLT elements from referenced stylesheets) and the details to include in the documentation.
The **Settings** tab allows you to choose whether or not to include the following components: **Templates**, **Functions**, **Global parameters**, **Global variables**, **Attribute sets**, **Character maps**, **Keys**, **Decimal formats**, **Output formats**, **Referenced stylesheets**.

You can choose whether or not to include the following other details:

- **Documentation** - Shows the documentation for each XSLT element. For HTML format, the user-defined data elements that are recognized and transformed in documentation blocks of the XSLT elements they precede, are the ones from the following schemas:
  - Oxygen XML Editor built-in XSLT documentation schema.
  - A subset of DocBook 5 elements. The recognized elements are: `section`, `sect1` to `sect5`, `emphasis`, `title`, `ulink`, `programlisting`, `para`, `orderedlist`, `itemizedlist`.
  - A subset of DITA elements. The recognized elements are: `concept`, `topic`, `task`, `codeblock`, `p`, `b`, `i`, `ul`, `ol`, `pre`, `sl`, `sli`, `step`, `steps`, `li`, `title`, `xref`.
  - Full XHTML 1.0 support.
  - XSLStyle documentation environment. XSLStyle uses DocBook or DITA languages inside its own user-defined data elements. The supported DocBook and DITA elements are the ones mentioned above.
DOXSL documentation framework (on page 3219). Supported elements are: codefrag, description, para, docContent, documentation, parameter, function, docSchema, link, list, listitem, module, parameter, template, attribute-set.

Other XSLT documentation blocks that are not recognized will just be serialized inside an HTML pre element. You can change this behavior by using a custom format (on page 931) instead of the built-in HTML format (on page 929) and providing your own XSLT stylesheets.

- **Use comments** - Controls whether or not the comments that precede an XSLT element is treated as documentation for the element they precede. Comments that precede or succeed the xsl:stylesheet element, are treated as documentation for the whole stylesheet. Note that comments that precede an import or include directive are not collected as documentation for the imported/included module. Also, comments from within the body of the XSLT elements are not collected at all.
- **Namespace** - Shows the namespace for named XSLT elements.
- **Location** - Shows the stylesheet location for each XSLT element.
- **Parameters** - Shows parameters of templates and functions.
- **References** - Shows the named XSLT elements that are referenced from within an element.
- **Used by** - Shows the list of all the XSLT elements that reference the current named element.
- **Supersedes** - Shows the list of all the XSLT elements that are superseded the current element.
- **Overriding** - Shows the list of all the XSLT elements that override the current element.
- **Return type** - Shows the return type of the function.
- **Source** - Shows the text stylesheet source for each XSLT element.
- **Import precedence** - Shows the computed import precedence as declared in the XSL transformation specifications.
- **Generate index** - Creates an index with all the XSLT elements included in the documentation.

Export settings - Save the current settings in a settings file for further use (for example, if you need the exported settings file for generating the documentation from the command-line interface).

Import settings - Reloads the settings from the exported file.

Generate - Use this button to generate the XSLT documentation.

**Tip:**
This function can be executed from an automated command-line script, for more details, see Scripting Oxygen (on page 3196).

Related Information:
XSLT Stylesheet Component Documentation Support (on page 912)
Generate XSLT Documentation in HTML Format

When using the XSLT Stylesheet Documentation dialog box (on page 925) to generate XSLT documentation in HTML format, it is presented in a visual diagram style with various sections, hyperlinks, and options.

Figure 277. XSLT Stylesheet Documentation Example

The generated documentation includes the following:

- **Table of Contents** - You can group the contents by namespace, location, or component type. The XSLT elements from each group are sorted alphabetically (named templates are presented first and the `<match>` elements second).

- **Information about main, imported, and included stylesheets.** This information consists of:
  - XSLT modules included or imported by the current stylesheet.
  - The XSLT stylesheets where the current stylesheet is imported or included.
  - The stylesheet location.
If you choose to split the output into multiple files, the table of contents is displayed in the left frame. The contents are grouped using the same criteria as the split.

After the documentation is generated, you can collapse or expand details for some stylesheet XSLT elements by using the Showing options or the Collapse or Expand buttons.

For each element included in the documentation, the section presents the element type followed by the element name (value of the `@name` or `@match` attribute for match templates).
Generate XSLT Documentation in a Custom Format

XSLT stylesheet documentation can be also generated in a custom format. You must write your custom stylesheet based on the schema xslDocSchema.xsd from [OXYGEN_INSTALL_DIR]/frameworks/stylesheet_documentation. You can create a custom format starting from one of the stylesheets used in the built-in HTML, PDF, and DocBook formats. These stylesheets are available in [OXYGEN_INSTALL_DIR]/frameworks/stylesheet_documentation/xsl.

To generate XSLT documentation in a custom format:

1. Select Tools > Generate Documentation > XSLT Stylesheet Documentation to open the XSLT Stylesheet Documentation dialog box (on page 925).
2. Select Custom for the Format and click the Options button.
3. In this next dialog box, specify your own stylesheet to transform the intermediary XML generated in the documentation process.
4. You can also choose to copy additional resources into the output folder or choose whether or not to keep the intermediate XML files created during the documentation process.
5. Click OK to close this dialog box and then click Generate.
Compiling an XSL Stylesheet for Saxon

As of Saxon 10.6, it is possible to export a compiled form of a stylesheet as a JSON or XML file (called a stylesheet export file or SEF). Oxygen XML Editor includes a simple tool called Compile XSL Stylesheet for Saxon (found in the Tools menu) that does this for you.

Use-Cases for a Stylesheet Export File (SEF)

- **Use Saxon-JS to run transformations in a browser** - A stylesheet export file (SEF) is needed if you want to use the Saxon-JS product to run transformations in a browser, as in the following example:

  ```
  <script type="text/javascript" src="SaxonJS/SaxonJS.min.js"></script>
  <script>
    window.onload = function() {
      SaxonJS.transform({
        stylesheetLocation: "books.sef",
        sourceLocation: "books.xml"
      });
    }
  </script>
  ``

- **Use SEF to run transformations in Oxygen XML Editor** - You can also use a stylesheet export file (SEF) in Oxygen XML Editor to apply an XSLT transformation over an XML file. This requires Saxon-EE or Saxon-PE versions of the Saxon product and you must select one of those two versions for the Target when you configure the SEF file (on page 933). When configuring the XSLT transformation, you will specify the SEF file in the XSL URL field (on page 1465).

Compiling an SEF File

The Compile XSL Stylesheet for Saxon tool can be found in the Tools menu and it compiles a specified stylesheet as a JSON or an XML file (stylesheet export file).

If you choose Saxon-JS as the Target (the type of Saxon product that the export file will be used with), then the compiled stylesheet will be a JSON file with a file extension of .sef by default.
If you choose Saxon-EE, Saxon-PE, or Saxon-HE for the Target, then the compiled stylesheet will be an XML file with a file extension of .xsef by default.

Selecting this tool opens the Compile XSL Stylesheet for Saxon dialog box that allows you to configure some options for conversion.

![Compile XSL Stylesheet for Saxon Dialog Box](image)

This dialog box includes the following options:

**XSL URL**

Allows you to select URL of the source XSL stylesheet. You can specify the URL by using the text field, the history drop-down, or the browsing actions in the **Browse** drop-down list.

**Output file**

You can specify the path where the output file will be saved by entering it in the text field, using the **Insert Editor Variables** button, or using the browsing actions in the **Browse** drop-down list.

**Open in Editor**

Select this option to open the resulting stylesheet export file in the main Oxygen XML Editor editing pane.

**Target**

Allows you to select the type of Saxon product that the export file will be used with. You can choose Saxon-JS, Saxon-EE, Saxon-PE, or Saxon-HE.

**Relocatable**
Can be used to control the Saxon &-relocate parameter. You can select this option to produce a relocatable export package (SEF) that can be deployed to a different location, with a different base URI.

**Set the default namespace for unprefixed element names (&-ns&)**

Can be used to control the &-ns: (uri | ##any | ##html5) Saxon parameter that defines the handling of unprefixed element names that appear as name tests in path expressions and match patterns in the stylesheet:

- The **##any** value declares that unprefixed names are treated as a test on the local name of the element only. They will match regardless of namespace.
- The **##html5** value declares that an unprefixed element name will match either a name in the XHTML namespace or a name in no namespace. This option is primarily intended for use when generating stylesheets to run under Saxon-JS in the browser since the resulting behavior is close to that defined by the special rules in the HTML5 specification for XSLT and XPath running against an HTML5 DOM.
- You can also specify a valid URI by editing the value in the combo box. Specifying a URI sets the default namespace for elements and types (effectively a default value for xpath-default-namespace). Note that an explicit value for this attribute takes precedence.

**Use a configuration file (&-config&)**

Select this option if you want to use a Saxon 10.6 configuration file that will be executed for the XSLT transformation and validation processes. You can specify the path to the configuration file by entering it in the URL field, or by using the Insert Editor Variables button, or using the browsing actions in the Browse drop-down list.

**Compile**

Use this button to generate the stylesheet export file according the options selected in this dialog box.

**Editing Ant Build Files**

Oxygen XML Editor includes an Ant editor that provides a variety of specialized features to assist you with creating and editing Ant build files. The editor includes some specialized views, content completion assistance, automatic validation, syntax highlighting, Quick Assist (on page 3223) and Quick Fix (on page 3223) support, as well as numerous common editing and search features.

**Modular Contextual Ant Build File Editing Using 'Main Files' Support**

Smaller interrelated modules that define a complex Ant build file cannot be correctly edited or validated individually due to their interdependency with other modules. For example, a target defined in a main build file...
is not visible when you edit an included or imported module. Oxygen XML Editor provides support for defining
the main module (or modules), allowing you to edit any of the imported/included files in the context of the
larger Ant build structure.

You can set a main Ant build file either by using the main files support from the Project view (on page 418), or
a validation scenario (on page 935).

To set a main file using a validation scenario, add validation units that point to the main modules. Oxygen XML
Editor warns you if the current module is not part of the dependencies graph computed for the main build file.
In this case, it considers the current module as the main build file.

The advantages of editing in the context of main file (on page 3220) include:

• Correct validation of a module in the context of a larger build structure.
• Content Completion Assistant (on page 3217) displays all components valid in the current context.
• The Outline view (on page 938) displays the components collected from the entire build file structure.

Validating Ant Build Files

Oxygen XML Editor performs the validation of Ant build files with the help of a built-in processor, which is
largely based on the Apache Ant (on page 3216) libraries. The path to these libraries can be configured in the
Ant preferences page (on page 265). The validation processor accesses the parameters set in the associated
Ant transformation scenario (on page 1509) and uses them as Ant properties when validating the current
build script.

Oxygen XML Editor automatically validates Ant build files as you type. You can also validate the currently
edited file by selecting the Validate action from the Validation toolbar drop-down menu or the
Document > Validate menu.

Tip:

To make a custom task available in the Ant validation engine, add a JAR (on page 3220) file that
contains the task implementation to the library directory of the built-in Ant distribution that comes
bundled with Oxygen XML Editor (for example, {OXYGEN_INSTALL_DIR}/tools/ant/lib folder).

Create a Validation Scenario for Ant Build Files

If you want to customize the validation process for Ant build files, you can create a new validation scenario
(or configure an existing one). For example, if you want to validate interrelated modules that define a complex
Ant build file, you can specify the main Ant file by configuring a validation scenario. To create or configure a
validation scenario, select Configure Validation Scenario(s) from the Validation toolbar drop-down menu or the Document > Validate menu.
Passing parameters to the Ant validation engine

Ant validation scenarios cannot be configured to accept custom Ant parameters. However, you can specify values for the parameters in your Ant document using an Ant transformation scenario:

1. Create a new Ant transformation scenario (on page 1506).
2. Edit the transformation scenario and set all parameters (on page 1509) you need to pass to your Ant document.
3. Associate the new scenario with your Ant document (in the Configure Transformation Scenarios(s) dialog box (on page 1548)). You do not need to run the transformation scenario. Every time a validation operation is triggered, the built-in validation engine uses the parameters set in the transformation scenario.

Note:
This behavior is available only for the validation scenarios that use the built-in validation engine. The custom defined validation engines do not benefit from this functionality.

Transforming Ant Build Files

Oxygen XML Editor includes a few built-in transformation scenarios that allow you to transform Ant build files. They are listed in the Ant section in the Configure Transformation Scenario(s) dialog box (on page 1548):

- **Ant** - This transformation scenario runs as an external process that executes the Ant build script with the built-in Ant distribution (Apache Ant (on page 3216) version 1.9.8) that is included with Oxygen XML Editor, or optionally with a custom Ant distribution configured in the scenario.

- **Ant (with Saxon-HE XSLT support)** - This transformation scenario allows Ant XSLT tasks to be performed using Saxon 9 Home Edition libraries that come bundled with Oxygen XML Editor and all defined XML catalogs are also taken into account during the transformation.

Tip:
Certain Ant tasks require additional JAR libraries (for example, Ant mail tasks). The additional libraries can be added by editing the Ant transformation scenario, and in the Output tab, click the Libraries button (on page 1507) in the bottom right corner. This opens a dialog box where you can add JAR libraries. For a list of library dependencies, see https://ant.apache.org/manual/install.html#librarydependencies.

Related Information:
Editing a Transformation Scenario (on page 1545)
Configure Transformation Scenario(s) Dialog Box (on page 1548)
Applying Associated Transformation Scenarios (on page 1548)
Ant Transformation (on page 1506)
Ant Quick Fix Support

The Oxygen XML Editor *Quick Fix support* helps you resolve missing target reference errors that may occur when developing Ant build documents.

To activate this feature, hover over or place the cursor in the highlighted area of text where a validation error or warning occurs. If a *Quick Fix* is available for that particular error or warning, you can access the *Quick Fix* proposals with any of the following methods:

- When hovering over the error or warning, the proposals are presented in a tooltip pop-up window.
- If you place the cursor in the highlighted area where a validation error or warning occurs, a *Quick Fix* icon is displayed in the stripe on the left side of the editor. If you click this icon, Oxygen XML Editor displays the list of available fixes.
- With the cursor placed in the highlighted area of the error or warning, you can also invoke the *Quick Fix* menu by pressing `Alt + 1 (Command + Alt + 1 on macOS)` on your keyboard.

Oxygen XML Editor provides the following types of *Quick Fixes* for Ant build files:

- **Create new target** - Creates a new target with the specified name.
- **Change reference to "targetName"** - Corrects the reference to point to an already defined target.
- **Remove target reference** - Removes the erroneous reference.

Content Completion in Ant Build Files

The list of proposals offered by the *Content Completion Assistant* in Ant build files are context-sensitive and includes proposals that are valid at the current cursor position. It can be manually activated with the `Ctrl + Space (Command + Space on macOS)` shortcut.

The *Content Completion Assistant* proposes various item types that are defined in the current Ant build and in the imported and included builds. The proposals include:

- Element names
- Attribute names
- Property names

**Note:**

In addition to the user-defined properties, the *Content Completion Assistant* offers the following values:

- The system properties set in the Java Virtual Machine.
- The built-in properties that Ant provides.

- Target names
- Task and type reference IDs
Tip:
To make a custom task available in the Content Completion Assistant, add a JAR (on page 3220) file that contains the task implementation to the library directory of the built-in Ant distribution that comes bundled with Oxygen XML Editor (for example, \[OXYGEN_INSTALL_DIR\]/tools/ant/lib folder).

Note:
For Ant resources, the proposals are collected starting from the main files (on page 3220). The main files can be defined in the project or in the associated validation scenario. For further details about the Main Files support go to Defining Main Files at Project Level (on page 418).

Related Information:
http://ant.apache.org/manual/properties.html

Syntax Highlighting in Ant Files

Oxygen XML Editor supports syntax highlighting in Text mode to make it easier to read the semantics of the structured content by displaying each type of code in different colors and fonts.

To customize the colors or styles used for the syntax highlighting colors for Ant build files, follow these steps:

1. Open the Preferences dialog box (Options > Preferences) (on page 124).
2. Go to Editor > Syntax Highlight (on page 224).
3. Select and expand the Ant Property section in the top pane.
4. Select the component you want to change and customize the colors or styles using the selectors to the right of the pane.

You can see the effects of your changes in the Preview pane.

Related Information:
Syntax Highlight Preferences (on page 224)

Ant Outline View

The Outline view for Ant files displays the list of all the components (properties, targets, extension points, task/type definitions and references) from both the edited Ant build file and its imported and included modules. For Ant resources, the Outline view collects its components starting from the main files (on page 3220). The main files can be defined in the project and the main build file can be specified in a validation scenario. For more details, see Defining Main Files at Project Level (on page 418).

By default, it is displayed on the left side of the editor. If the view is not displayed, it can be opened by selecting it from the Window > Show View menu.
The following actions are available in the **Settings** menu on the Outline view toolbar:

**Filter returns exact matches**

The text filter of the **Outline** view returns only exact matches. By default, this filter is not selected.

**Selection update on cursor move**

Controls the synchronization between **Outline** view and source document. The selection in the **Outline** view can be synchronized with the cursor moves or the changes in the Ant editor. Selecting one of the components from the outline view also selects the corresponding item in the source document.

When the **Show components** option is selected, the following actions are available:

**Show XML structure**

Displays the XML document structure in a tree-like manner.

**Sort**

Sorts the components in the **Outline** view alphabetically.

**Show all components**

Displays all components that were collected starting from the *main file (on page 3220)*. This option is set by default.

**Show only local components**

Displays the components defined in the current file only.

**Group by location/type**
The build file components can be grouped by location and type.

When the Show XML structure option is selected, the following actions are available:

- **Show components**
  Switches the Outline view to the components display mode.

- **Flat presentation mode of the filtered results**
  When active, the application flattens the filtered result elements to a single level.

- **Show comments and processing instructions**
  Show/hide comments and processing instructions in the Outline view.

- **Show element name**
  Show/hide element name.

- **Show text**
  Show/hide additional text content for the displayed elements.

- **Show attributes**
  Show/hide attribute values for the displayed elements. The displayed attribute values can be changed from the Outline preferences panel (on page 306).

- **Configure displayed attributes**
  Displays the XML Structured Outline preferences page (on page 306).

The following actions are available in the contextual menu of the Outline view (when the Show XML structure option is selected in the Settings menu):

- **Append Child**
  Displays a list of elements that can be inserted as children of the current element.

- **Insert Before**
  Displays a list of elements that can be inserted as siblings of the current element, before the current element.

- **Insert After**
  Displays a list of elements that can be inserted as siblings of the current element, after the current element.

- **Edit Attributes**
  Displays an in-place attribute editing window.

- **Toggle Comment**
  Comments/uncomments the currently selected element.

- **Search References** Ctrl + Shift + R (Command + Shift + R on macOS)
Searches all references of the item found at current cursor position in the defined scope. See Find References and Declarations of Ant Components (on page 945) for more details.

**Search References in**

Searches all references of the item found at current cursor position in the specified scope. See Find References and Declarations of Ant Components (on page 945) for more details.

**Component Dependencies**

Opens the Ant Component Dependencies view (on page 943) that allows you to see the dependencies for the currently selected component.

**Rename Component in**

Renames the selected component. See Ant Refactoring Actions (on page 945) for more details.

**Cut, Copy, Delete**

Executes the typical editing actions on the currently selected component.

**Expand More**

Expands recursively all sub-components of the selected component.

**Collapse All**

Collapses recursively all sub-components of the selected component.

You can search a component in the Outline view by typing its name in the filter text field at the top of the view or directly on the tree structure. When you type the component name in the text field, you can switch to the tree structure using the following:

- Down arrow key
- Tab key
- Shift-Tab key combination

To switch from tree structure to the filter text field, you can use Tab and Shift-Tab.

**Tip:**

The search filter is case insensitive. The following wildcards are accepted:

- * - any string
- ? - any character
- , - patterns separator

If no wildcards are specified, the string to search is used as a partial match (such as *textToFind*).

The content of the Outline view and the editing area are synchronized. When you select a component in the Outline view, its definition is highlighted in the editing area.
Oxygen XML Editor has a predefined order for the groups in the **Outline** view:

- For location, the names of the files are sorted alphabetically. The file you are editing is located at the top of the list.
- For type, the order is: properties, targets, references.

**Note:**
When no grouping is available Oxygen XML Editor sorts the components depending on their order in the document. Oxygen XML Editor also takes into account the name of the file that the components are part of.

### Ant Resource Hierarchy/Dependencies View

The **Resource Hierarchy/Dependencies** view displays the hierarchy or dependencies for an Ant build file by analyzing imported or included build files. If the view is not displayed, it can be opened by selecting it from the **Window > Show View** menu.

If you want to see the hierarchy or dependencies of a build file, select it in the **Project view** (on page 402) and choose **Resource Hierarchy** or **Resource Dependencies** from the contextual menu.

The following actions are available on the toolbar of the **Resource Hierarchy/Dependencies** view:

- **Refresh**
  Refreshes the Hierarchy/Dependencies structure.

- **Stop**
  Stops the hierarchy/dependencies computing.

- **Show Hierarchy**
  Allows you to choose a resource to compute the hierarchy structure.

- **Show Dependencies**
  Allows you to choose a resource to compute the dependencies structure.

- **Configure dependencies search scope**
  Allows you to configure a scope to compute the dependencies structure. There is also an option for automatically using the defined scope for future operations.

- **History**
  Provides access to the list of previously computed dependencies. Use the **Clear history** button to remove all items from this list.

The contextual menu for a resource listed in the **Resource Hierarchy/Dependencies** view contains the following actions:

- **Open**
Opens the resource. You can also double-click a resource in the Hierarchy/Dependencies structure to open it.

**Go to reference**

Opens the source document where the resource is referenced.

**Copy location**

Copies the location of the resource.

**Move resource**

Moves the selected resource.

**Rename resource**

Renames the selected resource.

**Resource Hierarchy**

Shows the hierarchy for the selected resource.

**Resource Dependencies**

Shows the dependencies for the selected resource.

**Add to Main Files**

Adds the currently selected resource in the **Main Files** directory (on page 418).

**Expand More**

Expands more of the children of the selected resource from the Hierarchy/Dependencies structure.

**Collapse All**

Collapses all children of the selected resource from the Hierarchy/Dependencies structure.

**Tip:**

When a recursive reference is encountered in the Hierarchy view, the reference is marked with a special icon.

### Ant Component Dependencies View

The **Component Dependencies** view allows you to see the dependencies for a selected component. This is helpful if you want to see where components are used in the entire hierarchy. For example, if you want to find all the references where a given component is used.

If the view is not displayed, it can be opened by selecting it from the **Window > Show View** menu.

To see the dependencies of an Ant component:

1. Right-click the desired component in the editor or **Outline** view.
2. Select the **Component Dependencies** action from the contextual menu.
The action is available for the following components: properties, targets, extension-points, and references (those that have an ID set).

If a component contains multiple references, a small table is displayed at the bottom of the view that contains all the references. When a recursive reference is encountered, it is marked with a special icon 🗄.

The **Component Dependencies** view includes the following toolbar actions:

- **Refresh**
  - Refreshes the dependencies structure.

- **Stop**
  - Stops the dependency computation.

- **Configure**
  - Allows you to choose the search scope for computing the dependencies structure. This is helpful for making sure all imported/included resources are computed.

- **History**
  - Allows you to select from a list of the most recently used dependency computations.

In addition, the following actions are available in the contextual menu:

- **Go to First Reference**
  - Selects the first reference of the currently selected component in the dependencies tree.

- **Go to Component**
  - Shows the definition of the currently selected component in the dependencies tree.

**Related Information:**

Search and Refactor Operations Scope *(on page 831)*

---

**Highlight Component Occurrences**

When a component (for example property or target) is found at the current cursor position, they are highlighted in both the document and in the stripe bar at the right side of the document. Oxygen XML Editor also supports occurrences highlight for type and task references.

Customizable colors are used (one for the component definition and another one for component references). Occurrences are displayed until another component is selected and a new search is performed. All highlights are removed when you start to edit the document.

This feature is enabled by default. To configure it, open the Preferences dialog box *(Options > Preferences)* *(on page 124)* and go to Editor > Mark Occurrences. If your particular type of file is not selected, you can perform this search by going to Search > Search Occurrences in File Ctrl + Shift + U (Command + Shift + U on macOS) in the contextual menu. Matches are displayed in separate tabs of the Results view *(on page 548)*.
Related Information:
Mark Occurrences Preferences (on page 225)

Find References and Declarations of Ant Components

The following search actions related to references and declarations of Ant components are available from the Search submenu of the contextual menu and from the Document > References menu:

**Search References**
Searches all references of the item found at the current cursor position in the defined scope.

**Search References in**
Searches all references of the item found at the current cursor position in the file or files that you specify after selecting a scope for the search operation.

**Search Declarations**
Searches all declarations of the item found at the current cursor position in the defined scope.

**Search Declarations in**
Searches all declarations of the item found at the current cursor position in the file or files that you specify when defining a new scope.

**Search Occurrences in File**
Searches all occurrences of the item at the cursor position in the currently edited file.

The following action is available from the contextual menu and the Document > Schema menu:

**Go to Definition**
Moves the cursor to the location of the definition of the current item.

**Note:**
You can also use the **Ctrl + Single-Click (Command + Single-Click on macOS)** shortcut on a reference to display its definition.

Related Information:
Search and Refactor Operations Scope (on page 831)

Ant Refactoring Actions

The following refactoring actions can be applied on targets, extension-points, properties, and references and allow you to consistently rename a component in the entire Ant build file structure. They are available from the Refactoring submenu in the contextual menu of the current editor or from the Document > Refactoring menu:

**Rename Component**
Allows you to rename the current component (in-place). The component and all its references in the document are highlighted with a thin border and the changes you make to the component at the cursor position are updated in real time to all occurrences of the component. To exit the in-place editing, press the Esc or Enter key on your keyboard.

**Rename Component in**

Opens a dialog box that allows you to rename the selected component by specifying the new component name and the files to be affected by the modification. If you click the Preview button, you can view the files to be affected by the action.

**Figure 284. Rename Identity Constraint Dialog Box**

![Rename Identity Constraint Dialog Box]

**Ant Quick Assist Support**

The *Quick Assist* support (on page 3223) helps you to rapidly access search and refactoring actions. If one or more actions are available in the current context, they are accessible via a yellow bulb icon (💡) placed at the current line in the stripe on the left side of the editor. Also, you can invoke the *Quick Assist* menu by using the **Alt + 1** (**Meta + Alt + 1** on macOS) keyboard shortcuts.

The *Quick Assist* support offers direct access to the following actions:

**Rename Component in**
Renames the component and all its dependencies.

**Search Declarations**

Searches the declaration of the component in a predefined scope. It is available only when the context represents a component name reference.

**Search References**

Searches all references of the component in a predefined scope.

**Component Dependencies**

Searches the component dependencies in a predefined scope.

**Change Scope**

Configures the scope that will be used for future search or refactor operations.

**Rename Component**

Allows you to rename the current component in-place.

**Search Occurrences**

Searches all occurrences of the component within the current file.

**Related Information:**

- Ant Component Dependencies View (on page 943)
- Ant Resource Hierarchy/Dependencies View (on page 942)
- Ant Refactoring Actions (on page 945)
- Search and Refactor Operations Scope (on page 831)

---

**Editing XML Schemas**

An XML Schema describes the structure of an XML document and is used to validate XML document instances against it, to check that the XML instances conform to the specified requirements. If an XML instance conforms to the schema then it is said to be valid. Otherwise, it is invalid.

Oxygen XML Editor offers support for both XML Schema 1.0 and 1.1 and you can edit XML Schema files in the following editing modes:

- **Text editing mode (on page 989)** - Allows you to edit XML Schema files in a source editing mode.
- **Grid editing mode (on page 354)** - Displays XML Schema files in a structured spreadsheet-like grid.
- **Design editing mode (on page 355)** - Visual schema designer that helps you understand the structure and develop complex schemas.
- **Author editing mode (on page 588)** - The visual Author mode is also available for XML Schema, allowing you to visually edit the schema annotations. It presents a polished and compact view of the XML Schema, with support for links on imported/included schemas.
For information about applying and detecting schemas, see Associating a Schema to XML Documents (on page 816).

For more information about editing XML Schemas, see our video demonstration:

https://www.youtube.com/embed/vz1eIZELQgc

Related Information:
Associating a Schema to XML Documents (on page 816)

**XML Schema Design Mode (XML Schema Diagram Editor)**

This section includes topics that describe how to work with XML Schema documents in **Design** mode, including various features, actions that are available, and much more.

The **Design** mode in Oxygen XML Editor provides a simple and expressive XML Schema diagram editor for working with XML Schema documents. The schema diagram helps both the content authors who want to understand a schema and schema designers who develop complex schemas.

To switch to this mode, select **Design** at the bottom of the editing area.

The diagram font can be increased using the usual Oxygen XML Editor shortcuts: (Ctrl + "+" (Meta + "+" on macOS)), (Ctrl + "-" (Meta + "-" on macOS)), (Ctrl + 0 (Meta + 0 on macOS)) or (Ctrl + mouse wheel (Meta + mouse wheel on macOS)). The whole diagram can also be zoomed with one of the predefined factors available in the Schema preferences panel (on page 198). The same zoom factor is applied for the print and save actions.

For more information about designing XML Schemas, watch our video demonstration:

https://www.youtube.com/embed/vz1eIZELQgc

**Navigation in the XML Schema Design Mode**

The following editing and navigation features work for all types of schema components in the XML Schema **Design** mode:

- Select consecutive components on the diagram (components from the same level) using the **Shift** key. You can also make discontinuous selections in the schema diagram using the **Ctrl** (Meta on macOS) key. To deselect one of the components, use **Ctrl + Single-Click (Command + Single-Click on macOS)**.
- Use the arrow keys to navigate the diagram vertically and horizontally.
- Use **Home/End** keys to jump to the first/last component from the same level. Use **Ctrl + Home (Command + Home on macOS)** key combination to go to the diagram root and **Ctrl + End (Command + End on macOS)** to go to the last child of the selected component.
- You can easily go back to a previously visited component while moving from left to right. The path will be preserved only if you use the left arrow key or right arrow key. For example, if the current selection is...
on the second attribute from an attribute group and you press the left arrow key to jump to the attribute group, when you press the right arrow key, then the selection will be moved to the second attribute.

- Go back and forward between components viewed or edited in the diagram by selecting them in the **Outline view (on page 993):**
  - Back (go to previous schema component).
  - Forward (go to next schema component).
  - Go to Last Modification (go to last modified schema component).

- Copy, reference, or move global components, attributes, and identity constraints to another position and from one schema to another using the **Cut/Copy** and **Paste/Paste as Reference** actions.

- Go to the definition of an element or attribute with the **Go to Definition** action.

- Search in the diagram using the **Find/Replace dialog box (on page 432)** or the **Quick find toolbar (on page 445).** You can find/replace components only in the current file scope.

- You can expand and see the contents of the imports/includes/redefines in the diagram. To edit components from other schemas, the schema for each component will be opened as a separate file in Oxygen XML Editor.

**Tip:**
If an XML Schema referenced by the currently open schema was modified on disk, the change will be detected and you will be asked to refresh the current schema contents.

- Recursive references are marked with a **recurse symbol (🔍).** Click this symbol to navigate between the element declaration and its reference.

**Figure 285. Recursive Reference**

**XML Schema Palette View (Available in Design Mode)**

The **Palette** view is designed to offer quick access to XML Schema components and to improve the usability of the XML Schema diagram builder. You can use the **Palette** to drag and drop components in the **Design** mode. The components offered in the **Palette** view depend on the XML schema version set in the **XML Schema preferences page (on page 238).** If the view is not displayed, it can be opened by selecting it from the **Window > Show View** menu.
Components are organized functionally into 4 collapsible categories:

- **Basic components:** elements, group, attribute, attribute group, complex type, simple type, type alternative.
- **Compositors and Wildcards:** sequence, choice, all, any, any attribute, open content.
- **Directives:** import, include, redefine, override.
- **Identity constraints:** key, keyref, unique, selector, field, assert.

**Note:**

The type alternative, open content, override, and assert components are available for XML Schema 1.1.

To add a component to the edited schema:

- Click and hold a graphic symbol from the Palette view, then drag the component into the Design view.
- A line dynamically connects the component with the XML schema structure.
- Release the component into a valid position.

**Note:**

You cannot drop a component into an invalid position. When you hover the component into an invalid position, the mouse cursor changes its shape into ![invalid position](image). Also, the connector line changes its color from the usual dark gray to the color defined in the **Validation error highlight color option (on page 226)** (default color is red).

For more information about the Schema palette, watch our video demonstration:

https://www.youtube.com/embed/KaiHUXmpuAA
XML Schema Facets View (Available in Design Mode)

The Facets view for XML Schemas presents the facets for the selected component, if available. If the view is not displayed, it can be opened by selecting it from the Window > Show View menu.

Figure 287. Facets View

The default value of a facet is rendered in the Facets view with a blue color. The facets that can not be edited are rendered with a gray color. The grouping categories (for example: Enumerations and Patterns) are not editable. If these categories contain at least one child they are rendered with bold. Bold facets are facets with values set explicitly to them.

Important:

Usually inherited facets are presented as default in the Facets view but if patterns are inherited from a base type and also specified in the current simple type only the current specified patterns will be presented. You can see the effective pattern value obtained by combining the inherited and the specified patterns as a tooltip on the Patterns category.

Facets for components that do not belong to the currently edited schema are read-only but if you double-click them you can choose to open the corresponding schema and edit them.

You can edit a facet by double-clicking it or by pressing Enter, when that facet is selected. For some facets you can choose valid values from a list or you can specify another value. If a facet has an invalid value or a warning, it will be highlighted in the table with the corresponding foreground color. By default, facets with errors are presented with red and the facets with warnings with yellow. You can customize the error colors from the Document Checking user preferences (on page 226).

The Facets view provides the following actions in its toolbar and contextual menu:

- **Add**
  
  Allows you to add a new enumeration or a new pattern.
Remove

Allows you to remove the value of a facet.

Edit Annotations

Allows you to edit an annotation for the selected facet.

Move Up

Allows you to move up the current enumeration/pattern in Enumerations/Patterns category.

Move Down

Allows you to move down the current enumeration/pattern in Enumerations/Patterns category.

Copy

Copy the attribute value.

Open in Regular Expressions Builder

Rather than editing regular expressions manually, this action allows you to open the pattern in the XML Schema Regular Expressions Builder (on page 1028) that guides you through the process of testing and constructing the pattern.

Facets can be fixed to prevent a derivation from modifying its value. To fix a facet value just click the Pin button.

Schema Editing Actions

You can edit an XML schema using drag and drop operations or contextual menu actions.

Drag and drop is the easiest way to move the existing components to other locations in an XML schema. For example, you can quickly insert an element reference in the diagram with a drag and drop from the Outline view (on page 993) to a compositor in the diagram. Also, the components order in an <xs:sequence> can be easily changed using drag and drop.

If this property has not been set, you can easily set the attribute/element type by dragging over it a simple type or complex type from the diagram. If the type property for a simple type or complex type is not already set, you can set it by dragging over it a simple or complex type.

Depending on the drop area, various actions are available:

- Move - Context dependent, the selected component is moved to the destination.
- Reference - Context dependent, the selected component is referenced from the parent.
- Copy - If the Ctrl (Meta on macOS) key is pressed, a copy of the selected component is inserted to the destination.

Visual clues about the operation type are indicated by the mouse pointer shape:
- When moving a component.

- When referencing a component.

- When copying a component.

You can edit some schema components directly in the diagram. For these components, you can edit the name and the additional properties presented in the diagram by double-clicking the value you want to edit. If you want to edit the name of a selected component, you can also press Enter. The list of properties that can be displayed for each component can be customized in the Preferences (on page 198).

When editing references, you can choose from a list of available components. A component from an imported schema whose target namespace does not have an associated prefix is displayed in the list as componentName#targetNamespace. If the reference is from a target namespace that was not yet mapped, you are prompted to add prefix mappings for the inserted component namespace in the currently edited schema.

You can also change the compositor by double-clicking it and choose the compositor you want from the proposals list.

There are some components that cannot be edited directly in the diagram: imports, includes, redefines. The editing action can be performed if you double-click or press Enter on an import/include/redefine component. An edit dialog box is displayed, allowing you to customize the directives.

Related Information:
- Searching and Refactoring Actions in XML Schemas (on page 1002)
- XML Schema Component Dependencies View (on page 1000)
- XML Schema Resource Hierarchy / Dependencies View (on page 997)
- Generating Sample XML Files (on page 1006)
- Schema Design Preferences (on page 198)

**Contextual Menu Actions in the Design Mode**

The contextual menu of the Design mode includes the following actions:

- **Go to Definition (Ctrl + Shift + Enter)**
  Shows the definition for the currently selected component. For references, this action is available by clicking the arrow displayed in its bottom right corner.

- **Open Schema (Ctrl + Shift + Enter)**
  Opens the selected schema. This action is available for `<xsd:import>`, `<xsd:include>` and `<xsd:redefine>` elements. If the file you try to open does not exist, a warning message is displayed and you have the possibility to create the file.
**Edit Attributes**

Allows you to edit the attributes of the selected component in a small in-place editor that presents the same attributes as in the Attributes view (on page 995) and the Facets view (on page 951). The actions that can be performed on attributes in this dialog box are the same actions presented in the two views.

**Append child**

Offers a list of valid components, depending on the context, and appends your selection as a child of the currently selected component. You can set a name for a named component after it has been added in the diagram.

**Insert before**

Offers a list of valid components, depending on the context, and inserts your selection before the selected component, as a sibling. You can set a name for a named component after it has been added in the diagram.

**Insert after**

Offers a list of valid components, depending on the context, and inserts your selection after the selected component, as a sibling. You can set a name for a named component after it has been added in the diagram.

**New global**

Inserts a global component in the schema diagram. This action does not depend on the current context. If you choose to insert an import you have to specify the URL of the imported file, the target namespace and the import ID. The same information, excluding the target namespace, is requested for an `<xsd:include>` or `<xsd:redefine>` element.

**Note:**

If the imported file has declared a target namespace, the field **Namespace** is completed automatically.

**Edit Schema Namespaces**

When performed on the schema root, it allows you to edit the schema target namespace and namespace mappings. You can also invoke the action by double-clicking the target namespace property from Attributes view (on page 995) for the schema or by double-clicking the schema component.

**Edit Annotations**

Allows you to edit the annotation for the selected schema component in the Edit Annotations dialog box. You can perform the following operations in the dialog box:

- **Edit all appinfo/documentation items for a specific annotation** - All appinfo/documentation items for a specific annotation are presented in a table and can be easily edited. Information about an annotation item includes: type (documentation/appinfo), content,
source (optional, specify the source of the documentation/appinfo element) and xml:lang. The content of a documentation/appinfo item can be edited in the Content area below the table.

- Insert/Insert before/Remove documentation/appinfo - The Add button allows you to insert a new annotation item (documentation/appinfo). You can add a new item before the item selected in table by pressing the Insert Before button. Also, you can delete the selected item using the Remove button.
- Move items up/down - Do this by using the Move up and Move down buttons.
- Insert/Insert before/Remove annotation - Available for components that allow multiple annotations such as schemas or redefines.
- Specify an ID for the component annotation - An optional identifier for the annotation.

Annotations are rendered by default under the graphical representation of the component. When you have a reference to a component with annotations, these annotations are also presented in the diagram below the referenced component. To edit the annotations, use the Edit Annotations action from the contextual menu. If the reference component does not have annotations, you can edit the annotations of the referenced component by double-clicking the annotations area. Otherwise, you can edit the referenced component annotations only if you go to the definition of the component.

Note:
For imported/included components that do not belong to the currently edited schema, the Edit Annotations dialog box presents the annotation as read-only. To edit its annotation, open the schema where the component is defined.

Change XML Schema Version
Use this action to change the XML Schema version of the current document.

Extract Global Element
This action is available for local elements. A local element is made global and is replaced with a reference to the global element. The local element properties that are also valid for the global element declaration are kept.

Figure 288. Extracting a Global Element

If you use the Extract Global Element action on a element, the result is:
Extract Global Attribute

This action is available for local attributes. A local attribute is made global and replaced with a reference to the global attribute. The properties of local attribute that are also valid in the global attribute declaration are kept.

If you use the **Extract Global Attribute** action on a `@note` attribute, the result is:
Figure 291. Extracting a Global Attribute on a @note Attribute

Extract Global Group

This action is available for compositors (sequence, choice, all). This action extracts a global group and makes a reference to it. The action is available only if the parent of the compositor is not a group.

Figure 292. Extracting a Global Group

If you use the Extract Global Group action on the <sequence> element, the Extract Global Component dialog box is displayed and you can choose a name for the group. If you type personGroup, the result is:
Figure 293. Extracting a Global Group on a `<sequence>` Element

Extract Global Type

This action is used to extract an anonymous simple type or an anonymous complex type as global. For anonymous complex types, the action is available on the parent element.

Figure 294. Extracting a Global Simple Type

If you use the action on the `union` component and choose `numericST` for the new global simple type name, the result is:

Figure 295. Extracting a Global Simple Type on a `union` Component
If you use the action on a `<person>` element and choose `person_type` for the new complex type name, the result is:

**Rename Component in**

Renames the selected component.

**Cut Ctrl + X (Command + X on macOS)**

Cuts the selected component(s).

**Copy Ctrl + C (Command + C on macOS)**

Copies the selected component(s) to the clipboard.

**Copy XPath**

This action copies an XPath expression that identifies the selected element or attribute in an instance XML document of the edited schema and places it in the clipboard.

**Paste Ctrl + V (Command + V on macOS)**
Pastes the component(s) from the clipboard as children of the selected component.

**Paste as Reference**

Creates references to the copied component(s). If not possible, a warning message is displayed.

**Remove Delete**

Removes the selected component(s).

**Override component**

Copies the overridden component in the current XML Schema. This option is available for `xs:override` components.

**Redefine component**

The referenced component is added in the current XML Schema. This option is available for `xs:redefine` components.

**Optional**

Can be performed on element/attribute/group references, local attributes, elements, compositors, and element wildcards. The `minOccurs` property is set to 0 and the `use` property for attributes is set to `optional`.

**Unbounded**

Can be performed on element/attribute/group references, local attributes, elements, compositors, and element wildcards. The `maxOccurs` property is set to `unbounded` and the `use` property for attributes is set to `required`.

**Search**

Can be performed on local elements or attributes. This action makes a reference to a global element or attribute.

**Search References**

Searches all references of the item found at current cursor position in the defined scope if any.

**Search References in**

Searches all references of the item found at current cursor position in the specified scope.

**Search Occurrences in File**

Searches all occurrences of the item found at current cursor position in the current file.

**Component Dependencies**

Opens the Component Dependencies view (on page 1000) that allows you to see the dependencies for the currently selected component.

**Resource Hierarchy**

Opens the Resource Hierarchy / Dependencies view (on page 997) that allows you to see the hierarchy for the currently selected resource.

**Resource Dependencies**
Allows you to see the dependencies for the currently selected resource.

**Expand All**
Recursively expands all sub-components of the selected component.

**Collapse All**
Recursively collapses all sub-components of the selected component.

**Save as Image**
Saves the diagram as image, in JPEG, BMP, SVG or PNG format.

**Generate Sample XML Files**
Generates XML files using the current opened schema. The selected component is the XML document root. See more in the Generate Sample XML Files (on page 1006) section.

**Flatten Schema**
Recursively adds the components of included Schema files to the main one. It also flattens every imported XML Schema from the hierarchy.

**Options**
Opens the Schema preferences page (on page 198).

**XML Schema Components**
A schema diagram contains a series of interconnected components. To quickly identify the relation between two connected components, the connection is represented as:

- A thick line to identify a connection with a required component (in the following image, `<family>` is a required element).

**Figure 298. Example: Required Component**

- A thin line to identify a connection with an optional component (in the following image, `<email>` is an optional element).

**Figure 299. Example: Optional Component**
The topics in this section provide details about all of the available components and their symbols as they appear in an XML schema diagram.

xs:schema

Figure 300. The xs:schema Component

Defines the root element of a schema. A schema document contains representations for a collection of schema components, such as type definitions and element declarations, that have a common target namespace. See more info at http://www.w3.org/TR/xmlschema11-1/#element-schema.

By default, it displays the targetNamespace property when rendered.

Table 10. xs:schema Properties

<table>
<thead>
<tr>
<th>Property Name</th>
<th>Description</th>
<th>Possible Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Target Name-space</td>
<td>The schema target namespace</td>
<td>Any URI</td>
</tr>
<tr>
<td>Element Form Default</td>
<td>Determining whether or not local element declarations will be namespace-qualified by default</td>
<td>qualified, unqualified, [Empty] (default value is unqualified)</td>
</tr>
<tr>
<td>Attribute Form Default</td>
<td>Determining whether or not local attribute declarations will be namespace-qualified by default</td>
<td>qualified, unqualified, [Empty] (default value is unqualified)</td>
</tr>
<tr>
<td>Block Default</td>
<td>Default value of the block attribute of xs:element and xs:complexType</td>
<td>#all, extension, restriction, substitution, restriction extension, restriction substitution, extension substitution, restriction extension substitution, [Empty]</td>
</tr>
<tr>
<td>Final Default</td>
<td>Default value of the final attribute of xs:element and xs:complexType</td>
<td>#all, restriction, extension, restriction extension, [Empty]</td>
</tr>
<tr>
<td>Default Attributes</td>
<td>Specifies a set of attributes that apply to every complex Type in a schema document</td>
<td>Any</td>
</tr>
<tr>
<td>XPath Default Name-space</td>
<td>The default namespace used when the XPath expression is evaluated</td>
<td>##defaultNamespace, ##targetNamespace, ##local</td>
</tr>
<tr>
<td>Version</td>
<td>Schema version</td>
<td>Any token</td>
</tr>
</tbody>
</table>
### Table 10. `xs:schema` Properties (continued)

<table>
<thead>
<tr>
<th>Property Name</th>
<th>Description</th>
<th>Possible Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>The schema ID</td>
<td>Any ID</td>
</tr>
<tr>
<td>Component</td>
<td>The edited component name</td>
<td>Not editable property</td>
</tr>
<tr>
<td>System-ID</td>
<td>The schema system ID</td>
<td>Not editable property</td>
</tr>
</tbody>
</table>

### `xs:element`

**Figure 301. The `xs:element` Component**

Defines an element. An element declaration is an association of a name with a type definition, either simple or complex, an (optional) default value and a (possibly empty) set of identity-constraint definitions. See more info at [http://www.w3.org/TR/xmlschema11-1/#element-element](http://www.w3.org/TR/xmlschema11-1/#element-element).

An element by default displays the following properties when rendered in the diagram: `default`, `fixed`, `abstract` and `type`. When referenced or declared locally, the element graphical representation also contains the value for the `minOccurs` and `maxOccurs` properties (for 0..1 and 1..1 occurs the values are implied by the connector style) and the connectors to the element are drawn using dotted lines if the element is optional.

### Table 11. `xs:element` Properties

<table>
<thead>
<tr>
<th>Property Name</th>
<th>Description</th>
<th>Possible Values</th>
<th>Mentions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>The element name (always required)</td>
<td>Any NCName for global or local elements, any QName (on page 3223) for element references</td>
<td>If missing, will be displayed as <code>[element]</code> in diagram</td>
</tr>
<tr>
<td>Is Reference</td>
<td>When set, the local element is a reference to a global element</td>
<td>true/false</td>
<td>Appears only for local elements</td>
</tr>
<tr>
<td>Type</td>
<td>The element type</td>
<td>All declared or built-in types. In addition, the fol-</td>
<td>For all elements.</td>
</tr>
</tbody>
</table>
Table 11. *xs:element* Properties (continued)

<table>
<thead>
<tr>
<th>Property Name</th>
<th>Description</th>
<th>Possible Values</th>
<th>Mentions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Property Name</td>
<td>Description</td>
<td>Possible Values</td>
<td>Mentions</td>
</tr>
<tr>
<td>Base Type</td>
<td>The extended/restricted base type</td>
<td>All declared or built-in types</td>
<td>For elements with complex type, with simple or complex content</td>
</tr>
<tr>
<td>Mixed</td>
<td>Defines if the complex type content model will be mixed</td>
<td>true/false</td>
<td>For elements with complex type</td>
</tr>
<tr>
<td>Content</td>
<td>The content of the complex type</td>
<td>simple/complex</td>
<td>For elements with complex type that extends/restricts a base type. It is automatically detected</td>
</tr>
<tr>
<td>Content Mixed</td>
<td>Defines if the complex content model will be mixed</td>
<td>true/false</td>
<td>For elements with complex type that has a complex content</td>
</tr>
</tbody>
</table>
Table 11. *xs:element* Properties (continued)

<table>
<thead>
<tr>
<th>Property Name</th>
<th>Description</th>
<th>Possible Values</th>
<th>Mentions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default</td>
<td>Default value of the element. A default value is automatically assigned to the element when no other value is specified</td>
<td>Any string</td>
<td>The fixed and default attributes are mutually exclusive</td>
</tr>
<tr>
<td>Fixed</td>
<td>A simple content element may be fixed to a specific value using this attribute. A fixed value is also automatically assigned to the element and you cannot specify another value.</td>
<td>Any string</td>
<td>The fixed and default attributes are mutually exclusive</td>
</tr>
<tr>
<td>Min Occurs</td>
<td>Minimum number of occurrences of the element</td>
<td>A numeric positive value. Default value is 1</td>
<td>Only for references/local elements</td>
</tr>
<tr>
<td>Max Occurs</td>
<td>Maximum number of occurrences of the element</td>
<td>A numeric positive value (default value is 1)</td>
<td>Only for references/local elements</td>
</tr>
<tr>
<td>Substitution Group</td>
<td>Qualified name of the head of the substitution group that this element belongs to</td>
<td>All declared elements. For XML Schema 1.1 this property supports multiple values.</td>
<td>For global and reference elements</td>
</tr>
<tr>
<td>Abstract</td>
<td>Controls whether or not the element may be used directly in instance XML documents. When set to true, the element may still be used to define content models, but it must be substituted through a substitution group in the instance document.</td>
<td>true/false</td>
<td>For global elements and element references</td>
</tr>
<tr>
<td>Form</td>
<td>Defines if the element is &quot;qualified&quot; (belongs to the target namespace) or &quot;unqualified&quot; (doesn't belong to any namespace)</td>
<td>unqualified/qualified</td>
<td>Only for local elements</td>
</tr>
<tr>
<td>Property Name</td>
<td>Description</td>
<td>Possible Values</td>
<td>Mentions</td>
</tr>
<tr>
<td>---------------</td>
<td>-------------</td>
<td>----------------</td>
<td>---------</td>
</tr>
<tr>
<td>Nullable</td>
<td>When this attribute is set to true, the element can be declared as nil using an xsi:nil attribute in the instance documents</td>
<td>true/false</td>
<td>For global elements and element references</td>
</tr>
<tr>
<td>Target Name­space</td>
<td>Specifies the target namespace for local element and attribute declarations. The namespace URI may be different from the schema target namespace. This property is available for local elements only.</td>
<td>Not editable property</td>
<td>For all elements</td>
</tr>
<tr>
<td>Block</td>
<td>Controls if the element can be subject to a type or substitution group substitution. ‘#all’ blocks any substitution, ‘substitution’ blocks any substitution through substitution groups and ‘extension’/’restriction’ block any substitution (both through xsi:type and substitution groups) by elements or types, derived respectively by extension or restriction from the type of the element. Its default value is defined by the blockDefault attribute of the parent xs:schema.</td>
<td>#all, restriction, extension, substitution, extension restriction, extension substitution, restriction substitution, restriction extension substitution</td>
<td>For global elements and element references</td>
</tr>
<tr>
<td>Final</td>
<td>Controls whether the element can be used as the head of a substitution group for elements whose types are derived by extension or restriction from the type of the element. Its default value is defined by the finalDefault attribute of the parent xs:schema.</td>
<td>#all, restriction, extension, restriction extension, [Empty]</td>
<td>For global elements and element references</td>
</tr>
<tr>
<td>ID</td>
<td>The component ID</td>
<td>Any ID</td>
<td>For all elements</td>
</tr>
<tr>
<td>Component</td>
<td>The edited component name</td>
<td>Not editable property</td>
<td>For all elements</td>
</tr>
<tr>
<td>Name­space</td>
<td>The component namespace</td>
<td>Not editable property</td>
<td>For all elements</td>
</tr>
<tr>
<td>System ID</td>
<td>The component system ID</td>
<td>Not editable property</td>
<td>For all elements</td>
</tr>
</tbody>
</table>
**xs:attribute**

**Figure 302. The xs:attribute Component**

![Diagram of xs:attribute component]

The manager ID.

Defines an attribute. See more info at [http://www.w3.org/TR/xmlschema11-1/#element-attribute](http://www.w3.org/TR/xmlschema11-1/#element-attribute).

An attribute by default displays the following properties when rendered in the diagram: **default**, **fixed**, **use** and **type**. Connectors to the attribute are drawn using dotted lines if the attribute use is optional. The attribute name is stroked out if prohibited.

### Table 12. xs:attribute Properties

<table>
<thead>
<tr>
<th>Property Name</th>
<th>Description</th>
<th>Possible Value</th>
<th>Mentions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Attribute name (always required)</td>
<td>Any NCName for global/local attributes, all declared attributes' QName (on page 3223) for references</td>
<td>For all local or global attributes. If missing, will be displayed as '[attribute]' in the diagram.</td>
</tr>
<tr>
<td>Is Reference</td>
<td>When set, the local attribute is a reference</td>
<td>true/false</td>
<td>For local attributes</td>
</tr>
<tr>
<td>Type</td>
<td>Qualified name of a simple type</td>
<td>All global simple types and built-in simple types. In addition another 3 proposals are present: [anonymous restriction], [anonymous list], [anonymous union] for creating anonymous simple types more easily.</td>
<td>For all attributes. For references, the type is set to the referenced attribute.</td>
</tr>
<tr>
<td>Default</td>
<td>Default value. When specified, an attribute is added by the schema processor (if it is missing from the instance XML document) and it is given this value. The default and fixed attributes are mutually exclusive.</td>
<td>Any string</td>
<td>For all local or global attributes. For references the value is from the referenced attribute.</td>
</tr>
<tr>
<td>Property Name</td>
<td>Description</td>
<td>Possible Value</td>
<td>Mentions</td>
</tr>
<tr>
<td>---------------</td>
<td>-------------</td>
<td>---------------</td>
<td>----------</td>
</tr>
<tr>
<td>Fixed</td>
<td>When specified, the value of the attribute is fixed and must be equal to this value. The default and fixed attributes are mutually exclusive.</td>
<td>Any string</td>
<td>For all local or global attributes. For references the value is from the referenced attribute.</td>
</tr>
<tr>
<td>Use</td>
<td>Possible usage of the attribute. Marking an attribute &quot;prohibited&quot; is useful to exclude attributes during derivations by restriction.</td>
<td>optional, required, prohibited</td>
<td>For local attributes</td>
</tr>
<tr>
<td>Form</td>
<td>Specifies whether or not the attribute is qualified (must have a namespace prefix in the instance XML document). The default value for this attribute is specified by the attributeFormDefault attribute of the xs:schema document element.</td>
<td>unqualified/qualified</td>
<td>For local attributes</td>
</tr>
<tr>
<td>Inheritable</td>
<td>Specifies if the attribute is inheritable. Inheritable attributes can be used by &lt;alternative&gt; element on descendant elements</td>
<td>true/false</td>
<td>For all local or global attributes. The default value is false. This property is available for XML Schema 1.1.</td>
</tr>
<tr>
<td>Target Name- space</td>
<td>Specifies the target namespace for local attribute declarations. The namespace URI may be different from the schema target namespace.</td>
<td>Any URI</td>
<td>Setting a target namespace for local attribute is useful only when restricts attributes of a complex type that is declared in other schema with a different target namespace. This property is available for XML Schema 1.1.</td>
</tr>
<tr>
<td>ID</td>
<td>The component ID</td>
<td>Any ID</td>
<td>For all attributes</td>
</tr>
<tr>
<td>Component</td>
<td>The edited component name</td>
<td>Not editable property</td>
<td>For all attributes</td>
</tr>
<tr>
<td>Namespace</td>
<td>The component namespace</td>
<td>Not editable property</td>
<td>For all attributes</td>
</tr>
</tbody>
</table>
Table 12. *xs:attribute* Properties (continued)

<table>
<thead>
<tr>
<th>Property Name</th>
<th>Description</th>
<th>Possible Value</th>
<th>Mentions</th>
</tr>
</thead>
<tbody>
<tr>
<td>System ID</td>
<td>The component system ID</td>
<td>Not editable property</td>
<td>For all attributes</td>
</tr>
</tbody>
</table>

*xs:attributeGroup*

Figure 303. The *xs:attributeGroup* Component

![Diagram](area-properties)

The properties of an area.

Defines an attribute group to be used in complex type definitions. See more info at [http://www.w3.org/TR/xmlschema11-1/#element-attributeGroup](http://www.w3.org/TR/xmlschema11-1/#element-attributeGroup).

Table 13. *xs:attributeGroup* Properties

<table>
<thead>
<tr>
<th>Property Name</th>
<th>Description</th>
<th>Possible Values</th>
<th>Mentions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Attribute group name (always required)</td>
<td>Any NCName for global attribute groups, all declared attribute groups for reference</td>
<td>For all global or referenced attribute groups. If missing, will be displayed as '[attribute-Group]' in diagram.</td>
</tr>
<tr>
<td>ID</td>
<td>The component ID</td>
<td>Any ID</td>
<td>For all attribute groups</td>
</tr>
<tr>
<td>Component</td>
<td>The edited component name</td>
<td>Not editable property</td>
<td>For all attribute groups</td>
</tr>
<tr>
<td>Namespace</td>
<td>The component namespace</td>
<td>Not editable property</td>
<td>For all attribute groups</td>
</tr>
<tr>
<td>System ID</td>
<td>The component system ID</td>
<td>Not editable property</td>
<td>For all attribute groups</td>
</tr>
</tbody>
</table>

*xs:complexType*

Figure 304. The *xs:complexType* Component

![Diagram](person-type)

[Diagram of a person type]
Defines a top-level complex type. Complex Type Definitions provide for: See more data at http://www.w3.org/TR/xmlschema11-1/#element-complexType.

- Constraining element information items by providing Attribute Declarations governing the appearance and content of attributes.
- Constraining element information item children to be empty, or to conform to a specified element-only or mixed content model, or else constraining the character information item children to conform to a specified simple type definition.
- Using the mechanisms of Type Definition Hierarchy to derive a complex type from another simple or complex type.
- Specifying post-schema-validation infoset contributions for elements.
- Limiting the ability to derive additional types from a given complex type.
- Controlling the permission to substitute, in an instance, elements of a derived type for elements declared in a content model to be of a given complex type.

Tip:
A complex type that is a base type to another type will be rendered with yellow background.

Table 14. xs:complexType Properties

<table>
<thead>
<tr>
<th>Property Name</th>
<th>Description</th>
<th>Possible Values</th>
<th>Mentions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>The name of the complex type (always required)</td>
<td>Any NC-Name</td>
<td>Only for global complex types. If missing, will be displayed as '[complexType]' in diagram.</td>
</tr>
<tr>
<td>Base Type Definition</td>
<td>The name of the extended/restricted types</td>
<td>Any from the declared simple or complex types</td>
<td>For complex types with simple or complex content</td>
</tr>
<tr>
<td>Derivation Method</td>
<td>The derivation method</td>
<td>restriction/extension</td>
<td>Only when base type is set. If the base type is a simple type, the derivation method</td>
</tr>
<tr>
<td>Property Name</td>
<td>Description</td>
<td>Possible Values</td>
<td>Mentions</td>
</tr>
<tr>
<td>---------------</td>
<td>-------------</td>
<td>----------------</td>
<td>----------</td>
</tr>
<tr>
<td>Content</td>
<td>The content of the complex type</td>
<td>simple/complex</td>
<td>For complex types that extend/restrict a base type. It is automatically detected.</td>
</tr>
<tr>
<td>Mixed</td>
<td>Specifies if the complex content model will be mixed</td>
<td>true/false</td>
<td>For complex contents</td>
</tr>
<tr>
<td>Abstract</td>
<td>When set to true, this complex type cannot be used directly in the instance documents and needs to be substituted using an xsi:type attribute</td>
<td>true/false</td>
<td>For global and anonymous complex types</td>
</tr>
<tr>
<td>Block</td>
<td>Controls if a substitution (either through a xsi:type or substitution groups) can be performed for a complex type, which is an extension or a restriction of the current complex type. This attribute can only block such substitutions (it cannot &quot;unblock&quot; them), which can also be blocked in the element definition. The default value is defined by the blockDefault attribute of xs:schema.</td>
<td>all, extension, restriction, extension restriction, [Empty]</td>
<td>For global complex types</td>
</tr>
<tr>
<td>Final</td>
<td>Controls whether the complex type can be further derived by extension or restriction to create new complex types</td>
<td>all, extension, restriction, extension restriction, [Empty]</td>
<td>For global complex types</td>
</tr>
</tbody>
</table>
Table 14. xs:complexType Properties (continued)

<table>
<thead>
<tr>
<th>Property Name</th>
<th>Description</th>
<th>Possible Values</th>
<th>Mentions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default Attributes Apply</td>
<td>The schema element can carry a defaultAttributes attribute, which identifies an attribute group. Each complexType defined in the schema document then automatically includes that attribute group, unless this is overridden by the defaultAttributesApply attribute on the complexType element.</td>
<td>true/false</td>
<td>This property is available only for XML Schema 1.1</td>
</tr>
<tr>
<td>ID</td>
<td>The component ID</td>
<td>Any ID</td>
<td>For all complex types</td>
</tr>
<tr>
<td>Component</td>
<td>The edited component name</td>
<td>Not editable property</td>
<td>For all complex types</td>
</tr>
<tr>
<td>Namespace</td>
<td>The component namespace</td>
<td>Not editable property</td>
<td>For all complex types</td>
</tr>
<tr>
<td>System ID</td>
<td>The component system ID</td>
<td>Not editable property</td>
<td>For all complex types</td>
</tr>
</tbody>
</table>

xs:simpleType

Figure 305. The xs:simpleType Component

The person name.

Defines a simple type. A simple type definition is a set of constraints on strings and information about the values they encode, applicable to the normalized value of an attribute information item or of an element information item with no element children. Informally, it applies to the values of attributes and the text-only content of elements. See more info at http://www.w3.org/TR/xmlschema11-1/#element-simpleType.

Tip:
A simple type that is a base type to another type will be rendered with yellow background.
<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Possible Values</th>
<th>Scope</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Simple type name. Always required.</td>
<td>Any NCName</td>
<td>Only for global simple types. If missing, will be displayed as ['simpleType'] in diagram.</td>
</tr>
<tr>
<td>Derivation</td>
<td>A simple type category</td>
<td>restriction, list, or union</td>
<td>For all simple types</td>
</tr>
<tr>
<td>Base Type</td>
<td>A simple type definition component. Required if derivation method is set to restriction</td>
<td>All global simple types and built-in simple types. In addition another 3 proposals are present: [anonymous restriction], [anonymous list], [anonymous union] for easily create anonymous simple types.</td>
<td>For global and anonymous simple types with the derivation method set to restriction</td>
</tr>
<tr>
<td>Item Type</td>
<td>A simple type definition component. Required if derivation method is set to list.</td>
<td>All global simple types and built-in simple types(from schema for schema). In addition another 3 proposals are present: [anonymous restriction], [anonymous list], [anonymous union] for easily create anonymous simple types.</td>
<td>For global and anonymous simple types with the derivation method set to list. Derivation by list is the process of transforming a simple datatype (named the item type) into a whitespace-separated list of values from this datatype. The item type can be defined inline by adding a simpleType definition as a child element of the list element, or by reference, using the itemType attribute (it is an error to use both).</td>
</tr>
<tr>
<td>Member Types</td>
<td>Category for grouping union members</td>
<td>Not editable property</td>
<td>For global and anonymous simple types with the derivation method set to union</td>
</tr>
<tr>
<td>Member</td>
<td>A simple type definition component. Required if derivation method is set to union.</td>
<td>All global simple types and built-in simple types(from schema for schema). In addition another 3 proposals are present: [anonymous restriction], [anonymous list], [anonymous union] for easily create anonymous simple types.</td>
<td>For global and anonymous simple types with the derivation method set to union. Deriving a simple datatype by union merges the lexical spaces of several simple datatypes (called member types) to create a new simple datatype. The member types can be defined either by ref-</td>
</tr>
</tbody>
</table>
### Table 15. \textit{xs:simpleType} Properties (continued)

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Possible Values</th>
<th>Scope</th>
</tr>
</thead>
<tbody>
<tr>
<td>required if derivation method is set to union.</td>
<td></td>
<td>striction], [anonymous list], [anonymous union] for easily create anonymous simple types.</td>
<td>erence (through the memberTypes attribute) or embedded as simple datatype local definitions in the \textit{xs:union} element. Both styles can be mixed.</td>
</tr>
<tr>
<td>Final</td>
<td>Blocks any further derivations of this datatype (by list, union, derivation or all)</td>
<td>#all, list, restriction, union, list restriction, list union, restriction union. In addition, [Empty] proposal is present for set empty string as value.</td>
<td>Only for global simple types</td>
</tr>
<tr>
<td>ID</td>
<td>The component ID</td>
<td>Any ID</td>
<td>For all simple types</td>
</tr>
<tr>
<td>Component</td>
<td>The name of the edited component</td>
<td>Not editable property</td>
<td>Only for global and local simple types</td>
</tr>
<tr>
<td>Name-&lt;br&gt;space</td>
<td>The component name-space</td>
<td>Not editable property</td>
<td>For global simple types</td>
</tr>
<tr>
<td>System-&lt;br&gt;ID</td>
<td>The component system ID</td>
<td>Not editable property</td>
<td>Not present for built-in simple types</td>
</tr>
</tbody>
</table>

### \textit{xs:alternative}

The \textit{type alternatives} mechanism allows you to specify type substitutions on an element declaration.

**Note:**

\textit{xs:alternative} is available for XML Schema 1.1.
Table 16. \textit{xs:alternative} Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Possible Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>Specifies type substitutions for an element, depending on the value of the attributes</td>
<td>All declared or built-in types. In addition, the following anonymous types are available: [ST-restriction], [ST-union], [ST-list], [CT-anonymous], [CT-extension SC], [CT-restriction SC], [CT-restriction CC], [CT-extension CC]</td>
</tr>
<tr>
<td>Test</td>
<td>Specifies an XPath expression. If the XPath condition is valid, the specified type is selected as the element type. The expressions allowed are limited to a subset of XPath 2.0. Only the attributes of the current element and inheritable attributes from ancestor elements are accessible in the XPath expression. When you edit this property, the content completion list of proposals offers XPath expressions.</td>
<td>An XPath expression</td>
</tr>
<tr>
<td>XPath Default Namespace</td>
<td>The default namespace used when the XPath expression is evaluated</td>
<td>##defaultNamespace, ##targetNamespace, ##local</td>
</tr>
<tr>
<td>ID</td>
<td>Specifies the component ID</td>
<td>Any ID</td>
</tr>
<tr>
<td>Component</td>
<td>Specifies the type of XML schema component</td>
<td>Not editable property</td>
</tr>
<tr>
<td>System ID</td>
<td>Points to the document location of the schema</td>
<td>Not editable property</td>
</tr>
</tbody>
</table>

\textbf{xs:group}

Figure 307. The \textit{xs:group} Component

\begin{figure}
\centering
\includegraphics[width=0.5\textwidth]{mixed-level.png}
\caption{mixed-level}
\end{figure}
Defines a group of elements to be used in complex type definitions. See more info at http://www.w3.org/TR/xmlschema11-1/#element-group.

When referenced, the graphical representation also contains the value for the minOccurs and maxOccurs properties (for 0..1 and 1..1 occurs the values are implied by the connector style) and the connectors to the group are drawn using dotted lines if the group is optional.

Table 17. xs:group Properties

<table>
<thead>
<tr>
<th>Property Name</th>
<th>Description</th>
<th>Possible Values</th>
<th>Mentions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>The group name (always required)</td>
<td>Any NCName for global groups, all declared groups for reference</td>
<td>If missing, will be displayed as '[group]' in diagram</td>
</tr>
<tr>
<td>Min Occurs</td>
<td>Minimum number of occurrences of the group</td>
<td>A numeric positive value. Default value is 1</td>
<td>Appears only for reference groups</td>
</tr>
<tr>
<td>Max Occurs</td>
<td>Maximum number of occurrences of the group</td>
<td>A numeric positive value. Default value is 1</td>
<td>Appears only for reference groups</td>
</tr>
<tr>
<td>ID</td>
<td>The component ID</td>
<td>Any ID</td>
<td>For all groups</td>
</tr>
<tr>
<td>Component</td>
<td>The edited component name</td>
<td>Not editable property</td>
<td>For all groups</td>
</tr>
<tr>
<td>Name-space</td>
<td>The component name-space</td>
<td>Not editable property</td>
<td>For all groups</td>
</tr>
<tr>
<td>System ID</td>
<td>The component system ID</td>
<td>Not editable property</td>
<td>For all groups</td>
</tr>
</tbody>
</table>

xs:include

Figure 308. The xs:include Component

Adds multiple schemas with the same target namespace to a document. See more info at http://www.w3.org/TR/xmlschema11-1/#element-include.

Table 18. xs:include properties

<table>
<thead>
<tr>
<th>Property Name</th>
<th>Description</th>
<th>Possible Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Schema Location</td>
<td>Included schema location</td>
<td>Any URI</td>
</tr>
<tr>
<td>ID</td>
<td>Include ID</td>
<td>Any ID</td>
</tr>
<tr>
<td>Component</td>
<td>The component name</td>
<td>Not editable property</td>
</tr>
</tbody>
</table>
**xs:import**

*Figure 309. The *xs:import* Component*

Adds multiple schemas with a different target namespace to a document. See more info at [http://www.w3.org/TR/xmlschema11-1/#element-import](http://www.w3.org/TR/xmlschema11-1/#element-import).

**Table 19. *xs:import* Properties**

<table>
<thead>
<tr>
<th>Property Name</th>
<th>Description</th>
<th>Possible Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Schema Location</td>
<td>Imported schema location</td>
<td>Any URI</td>
</tr>
<tr>
<td>Namespace</td>
<td>Imported schema namespace</td>
<td>Any URI</td>
</tr>
<tr>
<td>ID</td>
<td>Import ID</td>
<td>Any ID</td>
</tr>
<tr>
<td>Component</td>
<td>The component name</td>
<td>Not editable property</td>
</tr>
</tbody>
</table>

**xs:redefine**

*Figure 310. The *xs:redefine* Component*

Redefines simple and complex types, groups, and attribute groups from an external schema. See more info at [http://www.w3.org/TR/xmlschema11-1/#element-redefine](http://www.w3.org/TR/xmlschema11-1/#element-redefine).

**Table 20. *xs:redefine* Properties**

<table>
<thead>
<tr>
<th>Property Name</th>
<th>Description</th>
<th>Possible Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Schema Location</td>
<td>Redefine schema loca-</td>
<td>Any URI</td>
</tr>
<tr>
<td>ID</td>
<td>Redefine ID</td>
<td>Any ID</td>
</tr>
<tr>
<td>Component</td>
<td>The component name</td>
<td>Not editable property</td>
</tr>
</tbody>
</table>

**xs:override**

*Figure 311. The *xs:override* Component*

The override construct allows replacements of old components with new ones without any constraint. See more info at [http://www.w3.org/TR/xmlschema11-1/#element-override](http://www.w3.org/TR/xmlschema11-1/#element-override).
Table 21. \textit{xs:override} Properties

<table>
<thead>
<tr>
<th>Property Name</th>
<th>Description</th>
<th>Possible Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Schema Location</td>
<td>Redefine schema location</td>
<td>Any URI</td>
</tr>
<tr>
<td>ID</td>
<td>Redefine ID</td>
<td>Any ID</td>
</tr>
</tbody>
</table>

\textbf{xs:notation}

Figure 312. The \textit{xs:notation} Component

Describes the format of non-XML data within an XML document. See more info at \url{http://www.w3.org/TR/xmlschema11-1/#element-notation}.

Table 22. \textit{xs:notation} Properties

<table>
<thead>
<tr>
<th>Property Name</th>
<th>Description</th>
<th>Possible values</th>
<th>Mentions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>The notation name (always required)</td>
<td>Any NCName</td>
<td>If missing, will be displayed as '[notation]' in diagram</td>
</tr>
<tr>
<td>System Identifier</td>
<td>The notation system identifier</td>
<td>Any URI</td>
<td>Required if public identifier is absent (otherwise, optional)</td>
</tr>
<tr>
<td>Public Identifier</td>
<td>The notation public identifier</td>
<td>A Public ID value</td>
<td>Required if system identifier is absent (otherwise, optional)</td>
</tr>
<tr>
<td>ID</td>
<td>The component ID</td>
<td>Any ID</td>
<td>For all notations</td>
</tr>
<tr>
<td>Component</td>
<td>The edited component name</td>
<td>Not editable property</td>
<td>For all notations</td>
</tr>
<tr>
<td>Namespace</td>
<td>The component namespace</td>
<td>Not editable property</td>
<td>For all notations</td>
</tr>
<tr>
<td>System ID</td>
<td>The component system ID</td>
<td>Not editable property</td>
<td>For all notations</td>
</tr>
</tbody>
</table>

\textbf{xs:sequence / xs:choice / xs:all}

Figure 313. \textit{xs:sequence}
\textit{xs:sequence} specifies that the child elements must appear in a sequence. Each child element occurs once by default. See more info at \url{http://www.w3.org/TR/xmlschema11-1/#element-sequence}.

\textbf{Figure 314. \textit{xs:choice}}

\textit{xs:choice} allows only one of the elements contained in the declaration to be present within the containing element. See more info at \url{http://www.w3.org/TR/xmlschema11-1/#element-choice}.

\textbf{Figure 315. \textit{xs:all}}

\textit{xs:all} specifies that the child elements can appear in any order. See more info at \url{http://www.w3.org/TR/xmlschema11-1/#element-all}.

The compositor graphical representation also contains the value for the \textit{minOccurs} and \textit{maxOccurs} properties (for 0..1 and 1..1 occurs the values are implied by the connector style) and the connectors to the compositor are drawn using dotted lines if the compositor is optional.

\begin{table}[h]
\centering
\caption{\textit{xs:sequence}, \textit{xs:choice}, \textit{xs:all} Properties}
\begin{tabular}{|l|l|l|l|}
\hline
\textbf{Property Name} & \textbf{Description} & \textbf{Possible Values} & \textbf{Mentions} \\
\hline
Compositor & Compositor type & sequence, choice, all & 'all' is only available as a child of a group or complex type \\
\hline
Min Occurs & Minimum occurrences of compositor & A numeric positive value, Default is 1 & The property is not present if compositor is 'all' and is child of a group \\
\hline
Max Occurs & Maximum occurrences of compositor & A numeric positive value, Default is 1 & The property is not present if compositor is 'all' and is child of a group \\
\hline
ID & The component ID & Any ID & For all compositors \\
\hline
Component & The edited component name & Not editable property & For all compositors \\
\hline
System ID & The component system ID & Not editable property & For all compositors \\
\hline
\end{tabular}
\end{table}

\textit{xs:any}

\textbf{Figure 316. The \textit{xs:any} Component}
Enables the author to extend the XML document with elements not specified by the schema. See more info at http://www.w3.org/TR/xmlschema11-1/#element-any.

The graphical representation also contains the value for the \textit{minOccurs} and \textit{maxOccurs} properties (for 0..1 and 1..1 occurs the values are implied by the connector style) and the connectors to the wildcard are drawn using dotted lines if the wildcard is optional.

\begin{table}[h]
\centering
\begin{tabular}{|l|l|l|}
\hline
\textbf{Property Name} & \textbf{Description} & \textbf{Possible Values} \\
\hline
\textbf{Name-space} & The list of allowed namespaces. The namespace attribute expects a list of namespace URIs. In this list, two values have a specific meaning: ‘##targetNamespace’ stands for the target namespace, and ‘##local’ stands for local attributes (without namespaces). & ##any, ##other, ##targetNamespace, ##local or anyURI \\
\hline
\textbf{not-Namespace} & Specifies the namespace that extension elements or attributes cannot come from & ##local, ##targetNamespace \\
\hline
\textbf{not QName} & Specifies an element or attribute that is not allowed & ##defined \\
\hline
\textbf{ProcessContents} & Type of validation required on the elements allowed for this wildcard & skip, lax, strict \\
\hline
\textbf{Min Occurs} & Minimum occurrences of any & A numeric positive value. Default is 1 \\
\hline
\textbf{Max Occurs} & Maximum occurrences of any & A numeric positive value. Default is 1 \\
\hline
\textbf{ID} & The component ID & Any ID \\
\hline
\textbf{Component} & The name of the edited component & Not editable property \\
\hline
\textbf{System ID} & The component system ID & Not editable property \\
\hline
\end{tabular}
\caption{xs:any Properties}
\end{table}
**xs:anyAttribute**

Figure 317. The *xs:anyAttribute* Component

Enables the author to extend the XML document with attributes not specified by the schema. See more info at http://www.w3.org/TR/xmlschema11-1/#element-anyAttribute.

<table>
<thead>
<tr>
<th>Property Name</th>
<th>Description</th>
<th>Possible Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>NameSpace</td>
<td>The list of allowed namespaces. The namespace attribute expects a list of namespace URIs. In this list, two values have a specific meaning: ‘##targetNamespace’ stands for the target namespace, and ‘##local’ stands for local attributes (without namespaces).</td>
<td>##any, ##other, ##targetNamespace, ##local or anyURI</td>
</tr>
<tr>
<td>ProcessContents</td>
<td>Type of validation required on the elements allowed for this wildcard</td>
<td>skip, lax, strict</td>
</tr>
<tr>
<td>ID</td>
<td>The component ID</td>
<td>Any ID</td>
</tr>
<tr>
<td>Component</td>
<td>The name of the edited component</td>
<td>Not editable property</td>
</tr>
<tr>
<td>SystemID</td>
<td>The component system ID</td>
<td>Not editable property</td>
</tr>
</tbody>
</table>

**xs:unique**

Figure 318. The *xs:unique* Component
Defines that an element or an attribute value must be unique within the scope. See more info at http://www.w3.org/TR/xmlschema11-1/#element-unique.

Table 26. xs:unique Properties

<table>
<thead>
<tr>
<th>Property Name</th>
<th>Description</th>
<th>Possible Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>The unique name (always required)</td>
<td>Any NCName</td>
</tr>
<tr>
<td>ID</td>
<td>The component ID</td>
<td>Any ID</td>
</tr>
<tr>
<td>Component</td>
<td>The edited component name</td>
<td>Not editable property</td>
</tr>
<tr>
<td>Namespace</td>
<td>The component namespace</td>
<td>Not editable property</td>
</tr>
<tr>
<td>System ID</td>
<td>The component system ID</td>
<td>Not editable property</td>
</tr>
</tbody>
</table>

xs:key

Figure 319. The xs:key Component

Specifies an attribute or element value as a key (unique, non-nullable and always present) within the containing element in an instance document. See more info at http://www.w3.org/TR/xmlschema11-1/#element-key.

Table 27. xs:key Properties

<table>
<thead>
<tr>
<th>Property Name</th>
<th>Description</th>
<th>Possible Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>The key name (always required)</td>
<td>Any NCName</td>
</tr>
<tr>
<td>ID</td>
<td>The component ID</td>
<td>Any ID</td>
</tr>
<tr>
<td>Component</td>
<td>The edited component name</td>
<td>Not editable property</td>
</tr>
<tr>
<td>Namespace</td>
<td>The component namespace</td>
<td>Not editable property</td>
</tr>
<tr>
<td>System ID</td>
<td>The component system ID</td>
<td>Not editable property</td>
</tr>
</tbody>
</table>
**xs:keyRef**

Figure 320. The *xs:keyRef* Component

Specifies that an attribute or element value corresponds to that of the specified key or unique element. See more info at [http://www.w3.org/TR/xmlschema11-1/#element-keyref](http://www.w3.org/TR/xmlschema11-1/#element-keyref).

A keyref by default displays the *Referenced Key* property when rendered.

<table>
<thead>
<tr>
<th>Property Name</th>
<th>Description</th>
<th>Possible Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>The keyref name (always required)</td>
<td>Any NCName</td>
</tr>
<tr>
<td>Referenced Key</td>
<td>The name of referenced key</td>
<td>Any declared element constraints</td>
</tr>
<tr>
<td>ID</td>
<td>The component ID</td>
<td>Any ID</td>
</tr>
<tr>
<td>Component</td>
<td>The edited component name</td>
<td>Not editable property</td>
</tr>
<tr>
<td>Namespace</td>
<td>The component namespace</td>
<td>Not editable property</td>
</tr>
<tr>
<td>System ID</td>
<td>The component system ID</td>
<td>Not editable property</td>
</tr>
</tbody>
</table>

**xs:selector**

Figure 321. The *xs:selector* Component

Specifies an XPath expression that selects a set of elements for an identity constraint. See more info at [http://www.w3.org/TR/xmlschema11-1/#element-selector](http://www.w3.org/TR/xmlschema11-1/#element-selector).

<table>
<thead>
<tr>
<th>Property Name</th>
<th>Description</th>
<th>Possible Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>XPath</td>
<td>Relative XPath expression identifying the element that the constraint applies to</td>
<td>An XPath expression</td>
</tr>
<tr>
<td>ID</td>
<td>The component ID</td>
<td>Any ID</td>
</tr>
</tbody>
</table>
Table 29. *xs:selector* Properties (continued)

<table>
<thead>
<tr>
<th>Property Name</th>
<th>Description</th>
<th>Possible Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Component</td>
<td>The edited component name</td>
<td>Not editable property</td>
</tr>
<tr>
<td>System ID</td>
<td>The component system ID</td>
<td>Not editable property</td>
</tr>
</tbody>
</table>

*xs:field*

Figure 322. The *xs:field* Component

Specifies an XPath expression that specifies the value used to define an identity constraint. See more info at http://www.w3.org/TR/xmlschema11-1/#element-field.

Table 30. *xs:field* Properties

<table>
<thead>
<tr>
<th>Property Name</th>
<th>Description</th>
<th>Possible Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>XPath</td>
<td>Relative XPath expression identifying the field(s) composing the key, key reference, or unique constraint</td>
<td>An XPath expression</td>
</tr>
<tr>
<td>ID</td>
<td>The component ID</td>
<td>Any ID</td>
</tr>
<tr>
<td>Component</td>
<td>The edited component name</td>
<td>Not editable property</td>
</tr>
<tr>
<td>System ID</td>
<td>The component system ID</td>
<td>Not editable property</td>
</tr>
</tbody>
</table>

*xs:assert*

Assertions provide a flexible way to control the occurrence and values of elements and attributes available in an XML Schema.

Note: *xs:assert* is available for XML Schema 1.1.
Table 31. `xs:assert` Properties

<table>
<thead>
<tr>
<th>Property Name</th>
<th>Description</th>
<th>Possible Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test</td>
<td>Specifies an XPath expression. If the XPath condition is valid, the specified type is selected as the element type. The expressions allowed are limited to a subset of XPath 2.0. Only the attributes of the current element and inheritable attributes from ancestor elements are accessible in the XPath expression. When you edit this property, the content completion list of proposals offers XPath expressions.</td>
<td>An XPath expression</td>
</tr>
<tr>
<td>XPath Default Namespace</td>
<td>The default namespace used when the XPath expression is evaluated</td>
<td><code>##defaultNamespace</code>, <code>##targetNamespace</code>, <code>##local</code></td>
</tr>
<tr>
<td>ID</td>
<td>Specifies the component ID</td>
<td>Any ID</td>
</tr>
<tr>
<td>Component</td>
<td>The edited component name</td>
<td>Not editable property</td>
</tr>
<tr>
<td>System ID</td>
<td>The component system ID</td>
<td>Not editable property</td>
</tr>
</tbody>
</table>

`xs:openContent`

**Figure 324. The `xs:openContent` Component**

The `openContent` element enables instance documents to contain extension elements to be inserted amongst the elements declared by the schema. You can declare open content for your elements at one place (within the `complexType` definition) or at the schema level.

For further details about the `openContent` component, go to [http://www.w3.org/TR/xmlschema11-1/#element-openContent](http://www.w3.org/TR/xmlschema11-1/#element-openContent).
Table 32. *xs:openContent* Properties

<table>
<thead>
<tr>
<th>Property Name</th>
<th>Description</th>
<th>Possible Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mode</td>
<td>Specifies where the extension elements can be inserted</td>
<td>The value can be: &quot;interleave&quot;, &quot;suffix&quot; or &quot;none&quot;. The default value is &quot;interleave&quot;.</td>
</tr>
<tr>
<td>ID</td>
<td>The component ID</td>
<td>Any ID</td>
</tr>
<tr>
<td>Component</td>
<td>The edited component name</td>
<td>Not editable property</td>
</tr>
<tr>
<td>System ID</td>
<td>The component system ID</td>
<td>Not editable property</td>
</tr>
</tbody>
</table>

Note:
This component is available for XML Schema 1.1 only. To change the version of the XML Schema, open the Preferences dialog box (Options > Preferences) (on page 124) and go to XML > XML Parser > XML Schema.

Constructs Used to Group Schema Components

This section explains the components that can be used for grouping other schema components.

Attributes

**Figure 325. Attributes Construct**

Groups all attributes and attribute groups belonging to a complex type.

Table 33. *attributes* Properties

<table>
<thead>
<tr>
<th>Property Name</th>
<th>Description</th>
<th>Possible Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Component</td>
<td>The element that has the attributes displayed</td>
<td>Not editable property</td>
</tr>
<tr>
<td>System ID</td>
<td>The component system ID</td>
<td>Not editable property</td>
</tr>
</tbody>
</table>
Constraints

Figure 326. Constraints Construct

Groups all constraints (xs:key (on page 982), xs:keyRef (on page 983), or xs:unique (on page 981)) belonging to an element.

Table 34. constraints Properties

<table>
<thead>
<tr>
<th>Property Name</th>
<th>Description</th>
<th>Possible Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Component</td>
<td>The element that has the constraints displayed</td>
<td>Not editable property</td>
</tr>
<tr>
<td>System ID</td>
<td>The component system ID</td>
<td>Not editable property</td>
</tr>
</tbody>
</table>

Substitutions

Figure 327. Substitutions Construct

Groups all elements that can substitute the current element.

Table 35. substitutions Properties

<table>
<thead>
<tr>
<th>Property Name</th>
<th>Description</th>
<th>Possible Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Component</td>
<td>The element that has the substitutions displayed</td>
<td>Not editable property</td>
</tr>
<tr>
<td>System ID</td>
<td>The component system ID</td>
<td>Not editable property</td>
</tr>
</tbody>
</table>
Schema Validation

Validation for the Design mode is seamlessly integrated in the Oxygen XML Editor XML documents validation (on page 772) capability.

Figure 328. XML Schema Validation

A schema validation error is presented by highlighting the invalid component:

- In the Attributes View (on page 995).
- In the diagram by surrounding the component that has the error with a red border.
- A marker on the errors stripe at the right of the diagram view.
- A status label with a red icon (ريد) below the diagram view.

Invalid facets for a component are highlighted in the Facets View (on page 951).

Components with invalid properties are rendered with a red border. This is a default color, but you can customize it in the Document checking user preferences (on page 226). When hovering an invalid component, the tooltip will present the validation errors associated with that component.

When editing a value that is supposed to be a qualified or unqualified XML name, the application provides automatic validation of the entered value. This proves to be very useful in avoiding setting invalid XML names for the given property.

If you validate the entire schema using the Validate action from the Document > Validate menu or from the Validation toolbar drop-down menu, all validation errors will be presented in the Errors tab. To resolve an error, just click it (or double-click for errors located in other schemas) and the corresponding schema component will be displayed as the diagram root so that you can easily correct the error.
Important:
If the schema imports only the namespace of other schema without specifying the schema location and a catalog is set up (on page 826) that maps the namespace to a certain location both the validation and the diagram will correctly identify the imported schema.

Tip:
If the validation action finds that the schema contains unresolved references, the application will suggest the use of validation scenarios, but only if the currently edited schema is an XML Schema module.

Edit Schema Namespaces
You can use the XML Schema Namespaces dialog box to easily set a target namespace and define namespace mappings for a newly created XML Schema. In the Design mode these namespaces can be modified anytime by choosing Edit Schema Namespaces from the contextual menu. You can also do this by double-clicking the schema root in the diagram.

The XML Schema Namespaces dialog box allows you to edit the following information:

- **Target namespace** - The target namespace of the schema.
- **Prefixes** - The dialog box displays a table with namespaces and the mapped prefixes. You can add a new prefix mapping or remove an already existing one.

Editing XML Schema in Text Editing Mode
The Oxygen XML Editor Text editing mode can be used for editing XML Schema in a source editing mode. It offers powerful content completion support, a synchronized Outline view, and multiple refactoring actions (on page 1002). The Outline view has two display modes: the standard outline (on page 539) mode and the components (on page 993) mode.

A diagram of the XML Schema can be presented side by side with the text. To activate the diagram presentation, select the Show Full Model XML Schema diagram option (on page 173) in the Diagram preferences page (on page 173).

Modular Contextual XML Schema Editing Using 'Main Files' Support
Smaller interrelated modules that define a complex XML Schema cannot be correctly edited or validated individually, due to their interdependency with other modules. For example, an element defined in a main schema document is not visible when you edit an included module. Oxygen XML Editor provides the support for defining the main module (or modules), thus allowing you to edit any of the imported/included schema files in the context of the larger schema structure.

You can set a main XML document either using the main files support from the Project view (on page 418), or using a validation scenario.
To set a *main file* using a validation scenario, add validation units that point to the main schemas. Oxygen XML Editor warns you if the current module is not part of the dependencies graph computed for the main schema. In this case, it considers the current module as the main schema.

The advantages of editing in the context of a *main file* include:

- Correct validation of a module in the context of a larger schema structure.
- *Content Completion Assistant* displays all the referable components valid in the current context. This include components defined in modules other than the currently edited one.
- The *Outline view* displays the components collected from the entire schema structure.

### Validating XML Schema Documents

By default, XML Schema files are validated as you type. To change this, open the *Preferences* dialog box (Options > Preferences), go to Editor > Document Checking, and deselect the *Enable automatic validation* option.

To validate an XML Schema document manually, select the *Validate* action from the *Validation* toolbar drop-down menu or the *Document > Validate* menu. When Oxygen XML Editor validates an XML Schema file, it expands all the included modules so the entire schema hierarchy is validated. The validation problems are highlighted directly in the editor, making it easy to locate and fix any issues.

Some validation messages have an icon in the *Info* column in the Results view or at the bottom of the main editor and clicking it opens a dialog box with additional information and a link to the W3C specification exactly at the location where the error is described, thus allowing you to understand the reason for that error.

Validation of an XML Schema containing a type definition with a `@minOccurs` or `@maxOccurs` attribute having a value larger than 256 limits the value to 256 and issues a warning about this restriction in the Message panel at the bottom of the Oxygen XML Editor window. Otherwise, for large values of the `@minOccurs` and `@maxOccurs` attributes, the validator fails with an *OutOfMemory* error that might make Oxygen XML Editor unusable without restarting the entire application.

**Important:**

If the schema imports only a namespace without specifying the schema location and a *catalog is set up* to map the namespace to a certain location, both validation and the schema components will correctly identify the imported schema.

**Related Information:**

- Validating XML Documents Against a Schema
- Embedding Schematron Rules in XML Schema or RELAX NG
- Validation Scenario
- Associating a Schema to XML Documents
Quick Fixes for DTD, XSD, and Relax NG Errors

Oxygen XML Editor offers Quick Fixes (on page 3223) for common errors that appear in XML documents that are validated against DTD, XSD, or Relax NG schemas.

**Note:**
For XML documents validated against XSD schemas, the Quick Fixes are only available if you use the default Xerces validation engine.

Quick Fixes are available in **Text** mode and **Author** mode.

Oxygen XML Editor provides Quick Fixes for numerous types of problems, including the following:

<table>
<thead>
<tr>
<th>Problem Type</th>
<th>Available Quick Fixes</th>
</tr>
</thead>
<tbody>
<tr>
<td>A specific element is required in the current context</td>
<td>Insert the required element</td>
</tr>
<tr>
<td>An element is invalid in the current context</td>
<td>Remove the invalid element</td>
</tr>
<tr>
<td>The content of the element should be empty</td>
<td>Remove the element content</td>
</tr>
<tr>
<td>An element is not allowed to have child elements</td>
<td>Remove all child elements</td>
</tr>
<tr>
<td>Text is not allowed in the current element</td>
<td>Remove the text content</td>
</tr>
<tr>
<td>A required attribute is missing</td>
<td>Insert the required attribute</td>
</tr>
<tr>
<td>An attribute is not allowed to be set for the current element</td>
<td>Remove the attribute</td>
</tr>
<tr>
<td>The attribute value is invalid</td>
<td>Propose the correct attribute values</td>
</tr>
<tr>
<td>ID value is already defined</td>
<td>Generate a unique ID value</td>
</tr>
<tr>
<td>References to an invalid ID</td>
<td>Change the reference to an already defined ID</td>
</tr>
</tbody>
</table>

Related Information:
Schematron Quick Fixes (SQF) (on page 815)
Ignoring/Unignoring Validation Problems (on page 812)
Content Completion in XML Schema

The intelligent Content Completion Assistant (on page 3217) allows you to quickly identify and insert elements, attributes, and attribute values that are valid in the current editing context. All available proposals are listed in a pop-up menu displayed at the current cursor position.

The Content Completion Assistant is enabled by default. To disable it, open the Preferences dialog box (Options > Preferences) (on page 124), go to Editor > Content Completion, and deselect the Enable content completion option (on page 211).

When active, the Content Completion Assistant displays a list of context-sensitive proposals valid at the current cursor position. It can be manually activated with the Ctrl + Space (Command + Space on macOS) shortcut. You can navigate through the list of proposals by using the Up and Down keys on your keyboard. For each selected item in the list, the Content Completion Assistant displays a documentation window. You can customize the size of the documentation window by dragging its top, right, and bottom borders.

To insert the selected proposal in Text mode, do one of the following:

- Press Enter or Tab to insert both the start and end tags and position the cursor inside the start tag in a position suitable for inserting attributes.
- Press Ctrl + Enter (Command + Enter on macOS) to insert both the start and end tags and positions the cursor between the tags in a position where you can start typing content.

Depending on the selected schema version (on page 1032), Oxygen XML Editor populates the proposals list with information taken either from XML Schema 1.0 or 1.1.

Oxygen XML Editor helps you to easily reference a component by providing the list of proposals (complex types, simple types, elements, attributes, groups, attribute groups, or notations) valid in the current context. The components are collected from the current file or from the imported/included schemas.

When editing <xs:annotation> or <xs:appinfo> elements of an XML Schema, the Content Completion Assistant proposes elements and attributes from a custom schema (by default ISO Schematron). This feature can be configured from the XSD Content Completion (on page 215) preferences page.

Syntax Highlighting in XML Schema

Oxygen XML Editor supports syntax highlighting in Text mode to make it easier to read the semantics of the structured content by displaying each type of code in different colors and fonts.

To customize the colors or styles used for the syntax highlighting colors for XML Schema files, follow these steps:

1. Open the Preferences dialog box (Options > Preferences) (on page 124).
2. Go to Editor > Syntax Highlight (on page 224).
3. Select and expand the XML section in the top pane.
4. Select the component you want to change and customize the colors or styles using the selectors to the right of the pane.
5. Select the **XSD** tab in the **Preview** pane to see the effects of your changes.

**Tip:**

Oxygen XML Editor also allows you to specify syntax highlighting colors for specific XML elements and attributes with specific namespace prefixes. This can be done in the **Editor > Syntax Highlight > Elements/Attributes by Prefix** preferences page (on page 225).

**Related Information:**

Customize Syntax Highlight colors (on page 224)

---

**XML Schema Outline View**

The **Outline** view for XML Schemas presents all the global components grouped by their location, namespace, or type. By default, it is displayed on the left side of the editor. If the view is not displayed, it can be opened by selecting it from the **Window > Show View** menu.

**Figure 329. Outline View for XML Schema**
The **Outline** view provides the following options in the 📃 **Settings** menu on the **Outline** view toolbar:

**Filter returns exact matches**

The text filter of the **Outline** view returns only exact matches;

**Selection update on cursor move**

Allows a synchronization between **Outline** view and schema diagram. The selected view from the diagram is also selected in the **Outline** view.

**Sort**

Allows you to sort alphabetically the schema components.

**Show all components**

Displays all components that were collected starting from the *main files (on page 3220)*. Components that are not referable from the current file are marked with an orange underline. To reference them, add an import directive with the `componentNS` namespace.

**Show referable components**

Displays all components (collected starting from the *main files (on page 3220)*) that can be referenced from the current file. This option is set by default.

**Show only local components**

Displays the components defined in the current file only.

**Group by location/namespace/type**

These three operations allow you to group the components by location, namespace, or type. When grouping by namespace, the main schema target namespace is the first presented in the **Outline** view.

The following contextual menu actions are available in the **Outline** view:

**Remove (Delete)**

Removes the selected item from the diagram.

**Search References (Ctrl + Shift + R (Meta + Shift + R on macOS))**

Searches all references of the item found at current cursor position in the defined scope, if any.

**Search References in**

Searches all references of the item found at current cursor position in the specified scope.

**Component Dependencies (Ctrl + Shift + F4 (Meta + Shift + F4 on macOS))**

Opens the **Component Dependencies** view *(on page 1000)* that allows you to see the dependencies for the currently selected component.

**Resource Hierarchy (F4)**

Opens the **Resource Hierarchy / Dependencies** view *(on page 832)* that allows you to see the hierarchy for the currently selected resource.
Resource Dependencies (Shift + F4)

Opens the Resource Hierarchy / Dependencies view (on page 832) that allows you to see the dependencies for the currently selected resource.

 Rename Component in

Renames the selected component.

 Generate Sample XML Files

Generate XML files using the currently open schema. The selected component is the XML document root.

The upper part of the Outline view contains a filter box that allows you to focus on the relevant components. Type a text fragment in the filter box and only the components that match it are presented. For advanced usage you can use wildcard characters (such as * or ?) and separate multiple patterns with commas.

Tip:
The search filter is case insensitive. The following wildcards are accepted:

- * - any string
- ? - any character
- , - patterns separator

If no wildcards are specified, the string to search will be searched as a partial match.

The content of the Outline view and the editing area are synchronized. When you select a component in the Outline view, its definition is highlighted in the editing area.

Related Information:
Searching and Refactoring Actions in XML Schemas (on page 1002)
XML Schema Component Dependencies View (on page 1000)
XML Schema Resource Hierarchy / Dependencies View (on page 997)
Generating Sample XML Files (on page 1006)
Modular Contextual Relax NG Schema Editing Using 'Main Files' Support (on page 1083)

XML Schema Attributes View

The Attributes view for XML Schemas presents the properties for the selected component in the schema diagram. By default, it is displayed on the right side of the editor. If the view is not displayed, it can be opened by selecting it from the Window > Show View menu.
The default value of a property is presented in the Attributes view with blue foreground. The properties that can not be edited are rendered with gray foreground. A non-editable category that contains at least one child is rendered with bold. Bold properties are properties with values set explicitly to them.

Properties for components that do not belong to the currently edited schema are read-only but if you double-click them you can choose to open the corresponding schema and edit them.

You can edit a property by double-clicking by pressing Enter. For most properties you can choose valid values from a list or you can specify another value. If a property has an invalid value or a warning, it will be highlighted in the table with the corresponding foreground color. By default, properties with errors are highlighted with red and the properties with warnings are highlighted with yellow. You can customize these colors from the Document checking user preferences (on page 226).

For imports, includes and redefines, the properties are not edited directly in the Attributes view. A dialog box will open that allows you to specify properties for them.

The schema namespace mappings are not presented in Attributes view. You can view/edit these by choosing Edit Schema Namespaces from the contextual menu on the schema root. See more in the Edit Schema Namespaces (on page 989) section.

The Attributes view has five actions available on the toolbar and also on the contextual menu:

- **Add**
  
  Allows you to add a new member type to an union's member types category.

- **Remove**
  
  Allows you to remove the value of a property.
Move Up
Allows you to move up the current member to an union's member types category.

Move Down
Allows you to move down the current member to an union's member types category.

Copy
Copy the attribute value.

Go to Definition
Shows the definition for the selected type.

Show Facets
Allows you to edit the facets for a simple type.

XML Schema Resource Hierarchy / Dependencies View

The Resource Hierarchy / Dependencies view displays the hierarchy or dependencies for resources included in an XML Schema. If the view is not displayed, it can be opened by selecting it from the Window > Show View menu.

The Resource Hierarchy / Dependencies is useful when you want to start from an XML Schema (XSD) file and build and review the hierarchy of all the other XSD files that are imported, included or redefined in the given XSD file. The view is also able to build the tree structure, that is the structure of all other XSD files that import, include or redefine the given XSD file. The scope of the search is configurable (the current project, a set of local folders, etc.)

If you want to see the hierarchy or dependencies of an XML schema, select the desired schema in the Project view (on page 402) and choose Resource Hierarchy or Resource Dependencies from the contextual menu.
The following actions are available on the toolbar of the Resource Hierarchy/Dependencies view:

- **Refresh**
  
  Refreshes the Hierarchy/Dependencies structure.

- **Stop**
  
  Stops the hierarchy/dependencies computing.

- **Show Hierarchy**
  
  Allows you to choose a resource to compute the hierarchy structure.

- **Show Dependencies**
  
  Allows you to choose a resource to compute the dependencies structure.

- **Configure dependencies search scope**
  
  Allows you to configure a scope to compute the dependencies structure. There is also an option for automatically using the defined scope for future operations.

- **History**
  
  Provides access to the list of previously computed dependencies. Use the Clear history button to remove all items from this list.

The contextual menu for a resource listed in the Resource Hierarchy/Dependencies view contains the following actions:

- **Open**
Opens the resource. You can also double-click a resource in the Hierarchy/Dependencies structure to open it.

**Go to reference**

Opens the source document where the resource is referenced.

**Copy location**

Copies the location of the resource.

**Move resource**

Moves the selected resource.

**Rename resource**

Renames the selected resource.

**Resource Hierarchy**

Shows the hierarchy for the selected resource.

**Resource Dependencies**

Shows the dependencies for the selected resource.

**Add to Main Files**

Adds the currently selected resource in the Main Files directory (on page 418).

**Expand More**

Expands more of the children of the selected resource from the Hierarchy/Dependencies structure.

**Collapse All**

Collapses all children of the selected resource from the Hierarchy/Dependencies structure.

**Tip:**

When a recursive reference is encountered in the Hierarchy view, the reference is marked with a special icon ⚠.

**Note:**

The Move resource or Rename resource actions give you the option to update the references to the resource (on page 1000).

**Related Information:**

Modular Contextual XML Editing Using 'Main Files' Support (on page 829)

Search and Refactor Operations Scope (on page 831)
Moving/Renaming XML Schema Resources

You can move and rename a resource presented in the Resource/Hierarchy Dependencies view, using the Rename resource and Move resource refactoring actions from the contextual menu.

When you select the Rename action in the contextual menu of the Resource/Hierarchy Dependencies view, the Rename resource dialog box is displayed. The following fields are available:

- **New name** - Presents the current name of the edited resource and allows you to modify it.
- **Update references of the renamed resource(s)** - Select this option to update the references to the resource you are renaming. A Preview option is available that allows you to see what will be updated before selecting Rename to process the operation.

When you select the Move action from the contextual menu of the Resource/Hierarchy Dependencies view, the Move resource dialog box is displayed. The following fields are available:

- **Destination** - Presents the path to the current location of the resource you want to move and gives you the option to introduce a new location.
- **New name** - Presents the current name of the moved resource and gives you the option to change it.
- **Update references of the moved resource(s)** - Select this option to update the references to the resource you are moving, in accordance with the new location and name. A Preview option is available that allows you to see what will be updated before selecting Move to process the operation.

XML Schema Component Dependencies View

The Component Dependencies view allows you to see the dependencies for a selected component. This is helpful if you want to see where components are used in the entire hierarchy. For example, if you want to find all the references where a given component is used.

If the view is not displayed, it can be opened by selecting it from the Window > Show View menu.

To see the dependencies of an XML Schema component:

1. Right-click the desired component in the editor or Outline view.
2. Select the Component Dependencies action from the contextual menu.

The action is available for all named components (for example, elements or attributes).

If a component contains multiple references, a small table is displayed at the bottom of the view that contains all the references. When a recursive reference is encountered, it is marked with a special icon 📁.
The **Component Dependencies** view includes the following toolbar actions:

**Refresh**

Refreshes the dependencies structure.

**Stop**

Stops the dependency computation.

**Configure**

Allows you to choose the search scope for computing the dependencies structure. This is helpful for making sure all imported/included resources are computed.

**History**

Allows you to select from a list of the most recently used dependency computations.

In addition, the following actions are available in the contextual menu:

**Go to First Reference**

Selects the first reference of the currently selected component in the dependencies tree.
Go to Component

Shows the definition of the currently selected component in the dependencies tree.

For more information, see the Maintain Complex XML Schemas section of our Developing XML Schemas video demonstration:

https://www.youtube.com/embed/vZ1elZELQgc

Highlight Component Occurrences

When a component (for example types, elements, attributes) is found at current cursor position, Oxygen XML Editor performs a search over the entire document to find the component declaration and all its references. When found, they are highlighted both in the document and in the stripe bar, at the right side of the document. Customizable colors are used: one for the component definition and another one for component references. Occurrences are displayed until another component is selected and a new search is performed. All occurrences are removed when you start to edit the document.

This feature is on by default. To configured it, open the Preferences dialog box (Options > Preferences) (on page 124) and go to Editor > Mark Occurrences. A search can also be triggered with the Search > Search Occurrences in File () contextual menu action. All matches are displayed in a separate tab of the Results view (on page 548).

Searching and Refactoring Actions in XML Schemas

Search Actions

The following search actions can be applied on attribute, attributeGroup, element, group, key, unique, keyref, notation, simple, or complex types and are available from the Search submenu in the contextual menu of the current editor or from the Document > References menu:

Search References

Searches all references of the item found at current cursor position in the defined scope, if any. If a scope is defined, but the currently edited resource is not part of the range of resources determined by this, a warning dialog box is displayed and you have the possibility to define another search scope.

Search References in

Searches all references of the item found at current cursor position in the file or files that you specify when define a scope in the Search References dialog box.

Search Declarations
Searches all declarations of the item found at current cursor position in the defined scope if any. If a scope is defined, but the currently edited resource is not part of the range of resources determined by this, a warning dialog box will be displayed and you have the possibility to define another search scope.

**Search Declarations in**

Searches all declarations of the item found at current cursor position in the file or files that you specify when you define a scope for the search operation.

**Search Occurrences in File**

Searches all occurrences of the item at the cursor position in the currently edited file.

The following action is available from the contextual menu and the Document > Schema menu:

**Go to Definition**

Moves the cursor to the definition of the referenced XML Schema item.

**Note:**

You can also use the `Ctrl + Single-Click (Command + Single-Click on macOS)` shortcut on a reference to display its definition.

**Refactoring Actions**

The following refactoring actions can be applied on attribute, attributeGroup, element, group, key, unique, keyref, notation, simple, or complex types and are available from the Refactoring submenu in the contextual menu of the current editor or from the Document > Refactoring menu:

**Rename Component**

Allows you to rename the current component (in-place). The component and all its references in the document are highlighted with a thin border and the changes you make to the component at the cursor position are updated in real time to all occurrences of the component. To exit the in-place editing, press the **Esc** or **Enter** key on your keyboard.

**Rename Component in**

Opens a dialog box that allows you to rename the selected component by specifying the new component name and the files to be affected by the modification. If you click the **Preview** button, you can view the files to be affected by the action.
Related Information:
Search and Refactor Operations Scope (on page 831)

**XML Schema Quick Assist Support**

The *Quick Assist support* (on page 3223) improves the development work flow, offering fast access to the most commonly used actions when you edit schema documents.

The *Quick Assist feature* (on page 3223) is activated automatically when the cursor is positioned over the name of a component. It is accessible via a yellow bulb icon (💡) placed at the current line in the stripe on the left side of the editor. Also, you can invoke the *quick assist* menu by using the **Alt + 1** (**Meta + Alt + 1** on macOS) keyboard shortcuts.
The *Quick Assist* support offers direct access to the following actions:

- **Rename Component in**
  Renames the component and all its dependencies.

- **Search Declarations**
  Searches the declaration of the component in a predefined scope. It is available only when the context represents a component name reference.

- **Search References**
  Searches all references of the component in a predefined scope.

- **Component Dependencies**
  Searches the component dependencies in a predefined scope.

- **Change Scope**
  Configures the scope that will be used for future search or refactor operations.

- **Rename Component**
  Allows you to rename the current component in-place.

- **Search Occurrences**
  Searches all occurrences of the component within the current file.

For more information about improving schema development using the *Quick Assist* action set, watch our video demonstration:

https://www.youtube.com/embed/X-2-gkrFSGU

Related Information:
- Resource Hierarchy / Dependencies View *(on page 997)*
- Component Dependencies View *(on page 1000)*
- Searching and Refactoring Actions *(on page 1002)*
Generating Sample XML Files

Oxygen XML Editor offers support to generate sample XML files both from XML schema 1.0 and XML schema 1.1, depending on the XML schema version set in XML Schema preferences page (on page 238).

To generate sample XML files from an XML Schema, use the Generate Sample XML Files action from the Tools menu. This action is also available in the contextual menu of the schema Design mode (on page 952). The action opens the Generate Sample XML Files dialog box that allows you to configure a variety of options for generating the files.

The Generate Sample XML Files dialog box contains three tabs with various configurable options. Default values for these options can be set in the Sample XML Files Generator preferences page (on page 242). You can also run the tool from the command line using exported options.

Schema Tab

The first set of options for the Generate Sample XML Files tool are found in the Schema tab.

Figure 335. Generate Sample XML Files Dialog Box (Schema Tab)

This tab includes the following options:

**URL**

Specifies the URL of the Schema location. You can specify the path by using the text field, the history drop-down menu, or the browsing actions in the Browse drop-down list.
Namespace

Displays the namespace of the selected schema.

Root Element

After the schema is selected, this drop-down menu is populated with all root candidates gathered from the schema. Choose the root of the output XML documents.

Output folder

Path to the folder where the generated XML instances will be saved.

Filename prefix and Extension

You can specify the prefix and extension for the file name that will be generated. Generated file names have the following format: `prefixN.extension`, where `N` represents an incremental number from 0 up to the specified **Number of instances**.

Number of instances

The number of XML files to be generated.

Open first instance in editor

When selected, the first generated XML file is opened in the editor.

Namespaces section

You can specify the Default Namespace, as well as the prefixes for the namespaces.

Export settings

Use this button to save the current settings for future use.

Import settings

Use this button to load previously exported settings.

You can click **OK** at any point to generate the sample XML files.

Options Tab

The Options tab allows you to set specific options for namespaces and elements.
This tab includes the following options:

**Namespace / Element table**

Allows you to set a namespace for each element name that appears in an XML document instance. The following prefix-to-namespace associations are available:

- All elements from all namespaces (<ANY> - <ANY>). This is the default setting.
- All elements from a specific namespace.
- A specific element from a specific namespace.

**Settings subtab**

**Namespace**

Displays the namespace specified in the table at the top of the dialog box.

**Element**

Displays the element specified in the table at the top of the dialog box.
Generate optional elements

When selected, all elements are generated, including the optional ones (having the `minOccurs` attribute set to 0 in the schema).

Generate optional attributes

When selected, all attributes are generated, including the optional ones (having the `use` attribute set to `optional` in the schema).

Values of elements and attributes

Controls the content of generated attribute and element values. The following choices are available:

- **None** - No content is inserted.
- **Default** - Inserts a default value depending on the data type descriptor of the particular element or attribute. The default value can be either the data type name or an incremental name of the attribute or element (according to the global option from the Sample XML Files Generator preferences page). Note that type restrictions are ignored when this option is selected. For example, if an element is of a type that restricts an `xs:string` with the `xs:maxLength` facet to allow strings with a maximum length of 3, the XML instance generator tool may generate string element values longer than 3 characters.
- **Random** - Inserts a random value depending on the data type descriptor of the particular element or attribute.

⚠️ Important:

If all of the following are true, the Generate Sample XML Files tool outputs invalid values:

- At least one of the restrictions is a `regexp`.
- The value generated after applying the `regexp` does not match the restrictions imposed by one of the facets.

Preferred number of repetitions

Allows you to set the preferred number of repeating elements related to `minOccurs` and `maxOccurs` facets defined in the XML Schema.

- If the value set here is between `minOccurs` and `maxOccurs`, then that value is used.
- If the value set here is less than `minOccurs`, then the `minOccurs` value is used.
- If the value set here is greater than `maxOccurs`, then `maxOccurs` is used.

Maximum recursion level
If a recursion is found, this option controls the maximum allowed depth of the same element.

**Type alternative strategy**

Used for the `<xs:alternative>` element from XML Schema 1.1. The possible strategies are:

- **First** - The first valid alternative type is always used.
- **Random** - A random alternative type is used.

**Choice strategy**

Used for `<xs:choice>` or `<substitutionGroup>` elements. The possible strategies are:

- **First** - The first branch of `<xs:choice>` or the head element of `<substitutionGroup>` is always used.
- **Random** - A random branch of `<xs:choice>` or a substitute element or the head element of a `<substitutionGroup>` is used.

**Generate the other options as comments**

If selected, generates the other possible choices or substitutions (for `<xs:choice>` and `<substitutionGroup>`). These alternatives are generated inside comments groups so you can uncomment and use them later. Use this option with care (for example, on a restricted namespace and element) as it may generate large result files.

**Element values subtab**

Allows you to add values that are used to generate the content of elements. If there are multiple values, then the values are used in a random order.

**Attribute values subtab**

Allows you to add values that are used to generate the content of attributes. If there are multiple values, then the values are used in a random order.

**Export settings**

Use this button to save the current settings for future use.

**Import settings**

Use this button to load previously exported settings.

You can click **OK** at any point to generate the sample XML files.

**Advanced Tab**

The **Advanced** tab allows you to set some options regarding output values and performance.
Figure 337. Generate Sample XML Files Dialog Box (Advanced Tab)

This tab includes the following options:

**Use incremental attribute / element names as default**
If selected, the value of an element or attribute starts with the name of that element or attribute.
For example, for an `<a>` element the generated values are: `a1`, `a2`, `a3`, and so on. If not selected, the value is the name of the type of that element / attribute (for example: `string`, `decimal`, etc.)

**Maximum length**
The maximum length of string values generated for elements and attributes.

**Discard optional elements after nested level**
The optional elements that exceed the specified nested level are discarded. This option is useful for limiting deeply nested element definitions that can quickly result in very large XML documents.

**Export settings**
Use this button to save the current settings for future use.

**Import settings**
Use this button to load previously exported settings.

**Tip:**
This function can be executed from an automated command-line script, for more details, see Scripting Oxygen (on page 3196).

**Generating Documentation for an XML Schema**

Oxygen XML Editor can generate detailed documentation for the components of an XML Schema in HTML, PDF, DocBook, or other custom formats. You can select the components and the level of detail. The components are hyperlinked in both HTML and DocBook documents.
Note:
You can generate documentation for both XML Schema version 1.0 and 1.1.

To generate documentation for an XML Schema document, select XML Schema Documentation from the Tools > Generate Documentation menu or from the Generate Documentation submenu in the contextual menu of the Project view (on page 402). You can also open the tool by using the Generate Documentation toolbar button.

![XML Schema Documentation Dialog Box](image)

The Schema URL field of the dialog box must contain the full path to the XML Schema (XSD) file that will have documentation generated. The schema may be a local or a remote file. You can specify the path to the schema by entering it in the text field, or by using the Insert Editor Variables button or the options in the Browse drop-down menu.

**Output Tab**

The following options are available in the Output tab:
• **Format** - Allows you to choose between the following formats:
  ◦ **HTML** - The documentation is generated in **HTML output format** *(on page 1016).*
  ◦ **PDF** - The documentation is generated in **PDF output format** *(on page 1019).*
  ◦ **DocBook** - The documentation is generated in **DocBook output format** *(on page 1019).*
  ◦ **DITA** - The documentation is generated in **DITA output format** *(on page 1019).*
  ◦ **Custom** - The documentation is generated in a **custom output format** *(on page 1019),** allowing you to control the output. Click the **Options** button to open a **Custom format options** dialog box where you can specify a custom stylesheet for creating the output. There is also an option to **Copy additional resources to the output folder** and you can select the path to the additional **Resources** that you want to copy. You can also choose to keep the intermediate XML files created during the documentation process by deselecting the **Delete intermediate XML file** option.

• **Output file** - You can specify the path of the output file by entering it in the text field, or by using the **Insert Editor Variables** button or the options in the **Browse** drop-down menu.

• **Split output into multiple files** - Instructs the application to split the output into multiple files. You can choose to split them by namespace, location, or component name.

• **Open in Browser/System Application** - Opens the result in the system application associated with the output file type. For DITA and DocBook documents, this option appears as **Open in Editor** and the result will be opened in Oxygen XML Editor (in the current editor).

  **Note:**
  To set the browser or system application that will be used, open the **Preferences** dialog box *(Options > Preferences)* *(on page 124),* go to **Global**, and set it in the **Default Internet browser** field. This will take precedence over the default system application settings.

• **Keep only the annotations with xml:lang set to** - The generated output will contain only the annotations with the **@xml:lang** attribute set to the selected language. If you choose a primary language code (for example, **en** for English), this includes all its possible variations (**en-us, en-uk**, etc.).

**Settings Tab**

When you generate documentation for an XML schema you can choose what components to include in the output and the details to be included in the documentation.
The **Settings** tab allows you to choose whether or not to include the following components: **Global elements**, **Global attributes**, **Local elements**, **Local attributes**, **Simple Types**, **Complex Types**, **Groups**, **Attribute Groups**, **Redefines**, **Referenced schemas**, **Include notations**.

You can choose whether or not to include the following other details:

- **Diagram** - Displays the diagram for each component. You can choose the image format (JPEG, PNG, GIF, SVG) to use for the diagram section. The generated diagrams are dependent on the options from the [Schema Design Properties](on page 198) page.
- **Diagram annotations** - This option controls whether or not the annotations of the components presented in the diagram sections are included.
- **Namespace** - Displays the namespace for each component.
- **Location** - Displays the schema location for each component.
- **Type** - Displays the component type if it is not an anonymous one.
- **Type hierarchy** - Displays the types hierarchy.
- **Model** - Displays the model (sequence, choice, all) presented in BNF form. The separator characters that are used depend upon the information item used:
- **xs:all** - Its children will be separated by space characters.
- **xs:sequence** - Its children will be separated by comma characters.
- **xs:choice** - Its children will be separated by / characters.

- **Children** - Displays the list of component's children.
- **Instance** - Displays an XML instance generated based on each schema element.
- **Used by** - Displays the list of all the components that reference the current one. The list is sorted by component type and name.
- **Properties** - Displays some of the component's properties.
- **Facets** - Displays the facets for each simple type.
- **Identity constraints** - Displays the identity constraints for each element. For each constraint there are presented the name, type (unique, key, keyref), reference attribute, selector and field(s).
- **Attributes** - Displays the attributes for the component. For each attribute there are presented the name, type, fixed or default value, usage and annotation.
- **Asserts** - Displays the assert elements defined in a complex type. The test, XPath default namespace, and annotation are presented for each assert.
- **Annotations** - Displays the annotations for the component. If you choose **Escape XML Content**, the XML tags are present in the annotations.
- **Source** - Displays the text schema source for each component.
- **Generate index** - Displays an index with the components included in the documentation.
  - **Include local elements and attributes** - If selected, local elements and attributes are included in the documentation index.
  - **Include resource hierarchy** - Specifies whether or not the resource hierarchy for an XML Schema documentation is generated. It is deselected by default.

**Export settings** - Save the current settings in a settings file for further use (for example, if you need the exported settings file for generating the documentation from the command-line interface).

**Import settings** - Reloads the settings from the exported file.

**Generate** - Use this button to generate the XML Schema documentation.

**Tip:**
This function can be executed from an automated command-line script, for more details, see **Scripting Oxygen** *(on page 3196)*.

**Related Information:**
Customizing PDF or DocBook Output of Generated XML Schema Documentation *(on page 1020)*

**Output Formats for Generating XML Schema Documentation**

XML Schema documentation can be generated in HTML, PDF, DocBook, or a custom format. You can choose the format from the **Schema Documentation** *(on page 1011)* dialog box. For the PDF and DocBook formats, the option to split the output in multiple files is not available.
HTML Output Format

The XML Schema documentation generated in HTML format contains images corresponding to the same schema definitions as the ones displayed by the schema diagram editor. (on page 355) These images are divided in clickable areas that are linked to the definitions of the names of types or elements. The documentation of a definition includes a **Used By** section with links to the other definitions that reference it. If the **Escape XML Content** option is unchecked, the HTML or XHTML tags used inside the elements of the input XML Schema for formatting the documentation text (for example, `<b>`, `<i>`, `<u>`, `<ul>`, `<li>`, etc.) are rendered in the generated HTML documentation.

The generated images format is **PNG**. The image of an XML Schema component contains the graphical representation of that component as it is rendered in the **schema diagram panel of the Oxygen XML Editor XSD editor panel**. (on page 356)

![Image of XML Schema Documentation Example]

**Figure 340. XML Schema Documentation Example**

The generated documentation includes a table of contents. You can group the contents by namespace, location, or component type. After the table of contents there is some information about the main, imported, included, and redefined schemas. This information contains the schema target namespace, schema properties (attribute form default, element form default, version), and schema location.
If you choose to split the output into multiple files, the table of contents is displayed in the left frame. The contents are grouped in the same mode. If you split the output by location, each file contains a schema description and the components that you have chosen to include. If you split the output by namespace, each file contains information about schemas from that namespace and the list with all included components. If you choose to split the output by component, each file contains information about a schema component.

After the documentation is generated, you can collapse or expand details for some schema components by using the Showing options or the Collapse or Expand buttons.

For each component included in the documentation, the section presents the component type follow by the component name. For local elements and attributes, the name of the component is specified as parent name/component name. You can easily go to the parent documentation by clicking the parent name.
Figure 343. Documentation for a Schema Component

If the schema contains imported or included modules, their dependencies tree is generated in the documentation.

Figure 344. Example: Generated Documentation

If the schema contains imported or included modules, their dependencies tree is generated in the documentation.
PDF Output Format

For the PDF output format, the documentation is generated in DocBook format and a transformation using the FOP processor is applied to obtain the PDF file. To configure the FOP processor, see the FO Processors (on page 261) preferences page.

For information about customizing the PDF output, see Customizing PDF or DocBook Output of Generated XML Schema Documentation (on page 1020).

DocBook Output Format

If you generate the documentation in DocBook output format, the documentation is generated as a DocBook XML file. You can then apply a built-in DocBook transformation scenario (on page 1458) (such as, DocBook PDF or DocBook HTML) on the output file, or configure your own transformation scenario (on page 1464) to convert it into whatever format you desire.

For information about customizing the DocBook output, see Customizing PDF or DocBook Output of Generated XML Schema Documentation (on page 1020).

DITA Output Format

If you generate the documentation in DITA output format, each element of the schema is converted to a DITA Topic and all the generated topics are referenced in a DITA map (on page 3218) file. You can then apply a built-in DITA transformation scenario (such as, DITA Map PDF or DITA Map XHTML), or configure your own DITA-OT transformation scenario (on page 1490) to convert it into whatever format you desire.

For information about customizing the DITA output, see Customizing DITA Output of Generated XML Schema (on page 1021).

Custom Output Format

For the custom format, you can specify a stylesheet to transform the intermediary XML file generated in the documentation process. You have to edit your stylesheet based on the schema xsdDocSchema.xsd from [OXYGEN_INSTALL_DIR]/frameworks/schema_documentation. You can create a custom format starting from one of the stylesheets used in the built-in HTML, PDF, DocBook, and DITA formats. These stylesheets are available in [OXYGEN_INSTALL_DIR]/frameworks/schema_documentation/xsl.

When using a custom format you can also copy additional resources into the output folder and choose to keep the intermediate XML files created during the documentation process.

Important:

If you create a custom format for DITA, you must select the Split output into multiple files option in the Output tab (on page 1013) and choose Split by component.
Customizing PDF or DocBook Output of Generated XML Schema Documentation

To customize the PDF or DocBook output of the generated XML Schema documentation, use the following procedure:

1. Customize the \( \text{[OXYGEN\_INSTALL\_DIR]/frameworks/schema\_documentation/xsl/}
   
   \text{xsdDocDocbook.xsl} \) stylesheet to include the content that you want to add in the PDF or DocBook output. Add the content in the XSLT template with the \text{match="schemaDoc"} attribute between the \text{<info>} and \text{<xsl:apply-templates>} elements, as commented in the following example:

   \[
   \text{<info>}
   \text{<pubdate>}
   \text{<xsl:value-of select="format-date(current-date(),}
   \text{'[Mn] [D], [Y]', 'en', (), ()")"}>
   \text{</pubdate>}
   \text{</info>}
   \]

   <!-- Add the XSLT template content with match="schemaDoc" attribute here -->

   \[
   \text{<xsl:apply-templates select="schemaHierarchy"/>}
   \]

2. Create an intermediary file that holds the content of your XML Schema documentation by following these steps:

   a. Go to \text{Tools > Generate Documentation > XML Schema Documentation}.
   b. Select \text{Custom} for the output format and click the \text{Options} button.
   c. In the \text{Custom format options} dialog box, do the following:
      i. Enter the customized stylesheet in the \text{Custom XSL field}
      \( \text{([OXYGEN\_INSTALL\_DIR]/frameworks/schema\_documentation/xsl/}
      \text{xsdDocDocbook.xsl}) \).
      ii. Select the \text{Copy additional resources to the output folder} option and leave the default selection in the \text{Resources field}.
      iii. Click \text{OK}.
   d. When you return to the \text{XML Schema Documentation} dialog box, just click the \text{Generate} button to generate a DocBook XML file with an intermediary form of the Schema documentation.

3. If you want the DocBook file to be transformed into a PDF document, follow these steps:

   a. Use the \text{Configure Transformation Scenario(s)} action from the toolbar or the \text{Document > Transformation} menu, click \text{New}, and select \text{XML transformation with XSLT}.
   b. In the \text{New Scenario} dialog box, go to the \text{XSL URL field} and choose \( \text{([OXYGEN\_INSTALL\_DIR]/frameworks/docbook/oxygen/}
   \text{xsdDocDocbookCustomizationFO.xsl}) \).
   c. Go to the \text{FO Processor tab} and select the \text{Perform FO Processing} and \text{XSLT result as input} options.
   d. Go to the \text{Output tab} and select the directory where you want to store the XML Schema documentation output and click \text{OK}.
   e. Click \text{Apply Associated}.
Tip:
If you want your modifications to be permanent so that you can simply select the PDF output format in the XML Schema Documentation dialog box, rather than configuring a custom format each time you generate this documentation, follow this procedure:

1. Create a JAR (on page 3220) or ZIP file that includes the modified stylesheet (customized in step 1 above), placed in the following directory structure: builtin/documentation/schema_documentation/xsdDocDocbook.xsl.
2. Create a new directory named endorsed inside the [OXYGEN_INSTALL_DIR]/lib directory and place the created JAR or ZIP file inside it.
3. Restart Oxygen XML Editor and the PDF output format will now use your customized stylesheet.

Customizing DITA Output of Generated XML Schema

To customize the DITA output of the generated XML Schema documentation, use the following procedure:

1. Customize the [OXYGEN_INSTALL_DIR]/frameworks/schema_documentation/xsl/xsdDocDita.xsl stylesheet to incorporate your desired changes.
2. Create an intermediary file that holds the content of your XML Schema documentation by following these steps:
   a. Go to Tools > Generate Documentation > XML Schema Documentation.
   b. Select Custom for the output format and click the Options button.
   c. In the Custom format options dialog box, do the following:
      i. Enter the customized stylesheet in the Custom XSL field ([OXYGEN_INSTALL_DIR]/frameworks/schema_documentation/xsl/xsdDocDita.xsl).
      ii. Select the Copy additional resources to the output folder option and leave the default selection in the Resources field.
      iii. Click OK.
   d. Make sure the Split output into multiple files option (on page 1013) is selected and choose Split by component.
   e. When you return to the XML Schema Documentation dialog box, just click the Generate button to generate a DITA map file that contains the XML Schema documentation.

Converting Schema to Another Schema Language

The Generate/Convert Schema tool allows you to convert a DTD or Relax NG (full or compact syntax) schema or a set of XML files to an equivalent XML Schema, DTD or Relax NG (full or compact syntax) schema. Where perfect equivalence is not possible due to limitations of the target language, Oxygen XML Editor generates an approximation of the source schema. Oxygen XML Editor uses the Trang multiple format converter to perform the actual schema conversions.
To use this tool, select the **Generate/Convert Schema** (Alt + Shift + C (Command + Alt + C on macOS)) action from the **Tools** menu or from the **Open with** submenu when invoking the contextual menu in the **Project view** (on page 402). This action opens the **Generate/Convert Schema** dialog box that allows you to configure various options for conversion.

**Figure 345. Generate/Convert Schema Dialog Box**

The **Generate/Convert Schema** dialog box includes the following options:

**Input section**

Allows you to select the language of the source schema. If the conversion is based on a set of XML files, rather than just a single XML file, select the **XML Documents** option and use the file selector to add the XML files involved in the conversion.

**Output section**

Allows you to select the language of the target schema.

**Options**

You can choose the **Encoding**, the maximum **Line width**, and the **Indent size** (in number of spaces) for one level of indentation.

**Output file**

Specifies the path for the output file that will be generated.

**Close dialog when finished**
If you deselect this option, the dialog box will remain open after the conversion so that you can easily continue to convert more files.

Advanced options

If you select XML 1.0 DTD for the input, you can click this button to access more advanced options to further fine-tune the conversion. The following advanced options are available:

**XML 1.0 DTD Input section**

These options apply to the source DTD:

- **xmlns** - Specifies the default namespace, that is the namespace used for unqualified element names.
- **attlist-define** - Specifies how to construct the name of the definition representing an attribute list declaration from the name of the element. The specified value must contain exactly one percent character. This percent character is replaced by the name of element (after colon replacement) and the result is used as the name of the definition.
- **colon-replacement** - Replaces colons in element names with the specified chars when constructing the names of definitions used to represent the element declarations and attribute list declarations in the DTD.
- **any-name** - Specifies the name of the definition generated for the content of elements declared in the DTD as having a content model of ANY.
- **element-define** - Specifies how to construct the name of the definition representing an element declaration from the name of the element. The specified value must contain exactly one percent character. This percent character is replaced by the name of element (after colon replacement) and the result is used as the name of the definition.
- **annotation-prefix** - Default values are represented using a @prefix:defaultValue annotation attribute where prefix is the specified value and is bound to http://relaxng.org/ns/compatibility/annotations/1.0 as defined by the RELAX NG DTD Compatibility Committee Specification. By default, the conversion engine will use a for prefix unless that conflicts with a prefix used in the DTD.
- **inline-attlist** - Instructs the application not to generate definitions for attribute list declarations, but instead move attributes declared in attribute list declarations into the definitions generated for element declarations. This is the default behavior when the output language is XSD.
- **strict-any** - Preserves the exact semantics of ANY content models by using an explicit choice of references to all declared elements. By default, the conversion engine uses a wildcard that allows any element.
- **generate-start** - Specifies whether or not the conversion engine should generate a start element. DTD's do not indicate what elements are allowed
as document elements. The conversion engine assumes that all elements that are defined but never referenced are allowed as document elements.

- **xmlns mappings** table - Each row specifies the prefix used for a namespace in the input schema.

### W3C XML Schema Output section

This section is available if you select **W3C XML Schema** for the output.

- **disable-abstract-elements** - Disables the use of abstract elements and substitution groups in the generated XML Schema. This can also be controlled using an annotation attribute.
- **any-process-contents** - One of the values: strict, lax, skip. Specifies the value for the @processContents attribute of any elements. The default is skip (corresponding to RELAX NG semantics) unless the input format is DTD, in which case the default is strict (corresponding to DTD semantics).
- **any-attribute-process-contents** - Specifies the value for the @processContents attribute of `<anyAttribute>` elements. The default is skip (corresponding to RELAX NG semantics).

### Converting Database to XML Schema

Oxygen XML Editor includes a tool that allows you to create an XML Schema from the structure of a database.

To convert a database structure to an XML Schema, use the following procedure:

1. Select the **Convert DB Structure to XML Schema** action from the **Tools** menu.
   
   **Result:** The **Convert DB Structure to XML Schema** dialog box is opened and your current database connections are displayed in the **Connections** section.

2. If the database source is not listed, click the **Configure Database Sources** button to open the **Data Sources preferences page** (on page 276) where you can configure data sources and connections.

3. In the **Format for generated schema** section, select one of the following formats:
   - **Flat schema** - A flat structure that resembles a tree-like view of the database without references to elements.
   - **Hierarchical schema** - Display the table dependencies visually, in a type of tree view where dependent tables are shown as indented child elements in the content model. Select this option if you want to configure the database columns of the tables to be converted.

4. Click **Connect**.
   
   **Result:** The database structure is listed in the **Select database tables** section according to the format you chose.

5. Select the database tables that you want to be included in the XML Schema.

6. If you selected **Hierarchical schema** for the format, you can configure the database columns.
a. Select the database column you want to configure.
b. In the **Criterion** section you can choose to convert the selected database column as an **Element**, **Attribute**, or to be **Skipped** in the resulting XML Schema.
c. You can also change the name of the selected database column by changing it in the **Name** text field.

7. Click **Generate XML Schema**.

**Result:** The database structure is converted to an XML Schema and it is opened for viewing and editing.

### Flatten an XML Schema

You can organize an XML schema linked by `<xs:include>` and `<xs:import>` statements on several levels. In some cases, working on such a schema as if it were a single file is more convenient than working on multiple files separately. The **Flatten Schema** operation allows you to flatten an entire hierarchy of XML schemas. Starting with the main XML schema, Oxygen XML Editor calculates its hierarchy by processing the `<xs:include>` and `<xs:import>` statements.

The **Flatten Schema** action is available from the **Tools** menu or the contextual menu in **Text** mode. This action opens the **Flatten Schema** dialog box that allows you to configure the operation.

*Figure 346. Flatten Schema Dialog Box*
For the main schema file and for each imported schema, a new flattened schema is generated in the specified output folder. These schemas have the same name as the original ones.

Note:
If necessary, the operation renames the resulted schemas to avoid duplicated file names.

A flattened XML schema is obtained by recursively adding the components of the included schemas into the main one. This means Oxygen XML Editor replaces the `<xs:include>`, `<xs:redefine>`, and `<xs:override>` elements with the ones coming from the included files.

Options in the Flatten Schema Dialog Box

The following options are available in the Flatten Schema dialog box:

- **File name**
  The name of the output file.

- **Output directory**
  The path of the output directory where the flattened schema file will be saved.

- **Open the flattened XML Schema file in editor**
  Opens the main flattened schema in the editing area after the operation completes.

- **Use the XML Catalogs when collecting the referenced XML Schemas**
  Enables the imported and included schemas to be resolved through the available XML Catalogs (on page 3225).

  Note:
  Changing this option triggers the recalculation of the dependencies graph for the main schema.

- **Process the imported XML Schemas resolved through the XML Catalogs**
  Specifies whether or not the imported schemas that were resolved through an XML Catalog (on page 3225) are also processed.

- **Flatten the imported XML Schema(s)**
  Specifies whether or not the imported schemas are flattened.

  Note:
  For the schemas skipped by the flatten operation, no files are created in the output folder and the corresponding import statements remain unchanged.
Tip:
This function can be executed from an automated command-line script, for more details, see Scripting Oxygen (on page 3196).

Generating Java Classes from XML Schema

Oxygen XML Editor includes a tool for generating Java classes from an XML Schema (XSD) file. The Generate Java classes from XML Schema (XSD) action for invoking the tool can be found in the Tools menu. It requires an additional add-on to be installed, so the first time you invoke the action, Oxygen XML Editor will present a dialog box asking if you want to install it. Once installed, you need to restart Oxygen XML Editor and the action will invoke the Java class generator tool.

Quick Installation

You can drag the following Install button and drop it into the main editor in Oxygen (version 24.1 or newer) to quickly initiate the installation process:

![Install Button]

Manual Installation

To manually install it the add-on, follow these instructions:

1. Go to Help > Install new add-ons to open an add-on selection dialog box.
2. Enter or paste https://www.oxygenxml.com/InstData/Addons/default/updateSite.xml in the Show add-ons from field or select it from the drop-down menu.
3. Select the Generate Java classes from XML Schema (XSD) add-on and click Next.
4. Read the end-user license agreement. Then select the I accept all terms of the end-user license agreement option and click Finish.
5. Restart the application.

Result: The Generate Java classes from XML Schema (XSD) dialog box is now available and can be selected from the Tools menu.

Generating Java Classes

To generate Java classes, follow these steps:

1. Select the Generate Java classes from XML Schema (XSD) action from the Tools menu.

   Step Result: The Generate Java classes from XML Schema (XSD) dialog box is displayed:
2. Choose or enter the **XSD URL** of the XML Schema document.
3. Choose the path for the **Output folder** where the generated files will be stored.
4. [Optional] You can choose the **Package name** for the Java package that will contain the generated source files. If not specified, the name will be generated based on the value of the @targetNamespace attribute.
5. [Optional] You can select the **Open in Editor** option to open the `ObjectFactory.java` file in the editor. This java class contains factory methods for all other classes in the package.
6. Click the **Generate** button.

**Result:** The Java class files will be generated inside the new package, located in the specified output folder.

**XML Schema Regular Expressions Builder Tool**

The XML Schema regular expressions builder allows you to test regular expressions on a fragment of text as they are applied to an XML instance document. Start the tool by selecting **XML Schema Regular Expressions Builder** from the **Tools** menu.
The dialog box contains the following:

**Regular expressions editor**

Allows you to edit the regular expression to be tested and used. Content completion is available and presents a list with all the predefined expressions. It is triggered by pressing Ctrl + Space (Command + Space on macOS).

**Error display area**

If the edited regular expression is incorrect, an error message will be displayed here. The message contains the description and the exact location of the error. Also, clicking the quick navigation button (↩️) highlights the error inside the regular expression.

**Category**

You can choose from several categories of predefined expressions. The selected category influences the displayed expressions in the Available expressions table.

**Available expressions**

This table includes the available regular expressions and a short description for each of them. The set of expressions depends on the category selected in the previous Category combo box.

You can add an expression in the Regular expressions editor by double-clicking the expression...
row in the table. You will notice that in the case of Character categories and Block names, the expressions are also listed in complementary format.

Evaluate expression on

You can choose between two options:

- **Evaluate expression on each line** - The edited expression will be applied on each line in the Test area.
- **Evaluate expression on all text** - The edited expression will be applied on the whole text.

Test

A text editor that allows you to enter a text sample that will have the regular expression applied. All matches of the edited regular expression will be highlighted.

After editing and testing your regular expression you can insert it in the current editor. The Insert button will become active when an editor is opened in the background and there is an expression in the Regular expressions editor.

The regular expression builder cannot be used to insert regular expressions in the Grid mode (on page 354) or schema Design mode (on page 355). Accordingly, the Insert button will be not available if the current document is edited in these modes.

Note:

Some regular expressions may indefinitely block the Java Regular Expressions engine. If the execution of the regular expression does not end in about five seconds, the application displays a dialog box that allows you to interrupt the operation.

XML Schema 1.1

Oxygen XML Editor offers full support for XML Schema 1.1, including:

- XML Documents Validation (on page 772) and Content Completion (on page 532) based on XML Schema 1.1.
- XML Schema 1.1 Validation (on page 990) and Content Completion (on page 992).
- Editing XML Schema 1.1 files in the Schema Design mode (on page 355).
- The Flatten Schema (on page 1025) action.
- Resource Hierarchy/Dependencies (on page 997) and Refactoring Actions (on page 1002).
- Main files (on page 3220).
- Support for generating XML instances based on XML Schema.
- Support for validating XML documents with an NVDL schema that contains an XML Schema 1.1 validation step.
Note:
To enable XML Schema 1.1 validation in NVDL, you need to pass the following option to the validation engine to specify the schema version: http://www.thaiopensource.com/validate/xsd-version (the possible values are 1.0 or 1.1.

Tip:
To enable the full XPath expression in assertions and type alternatives, you need to set the http://www.thaiopensource.com/validate/full-xpath option.

XML Schema 1.1 is a superset of XML Schema 1.0, that offers lots of new powerful capabilities.

Figure 349. XML Schema 1.1

The significant new features in XSD 1.1 are:

- **Assertions** - Support to define assertions against the document content using XPath 2.0 expressions (an idea borrowed from Schematron).
- **Conditional type assignment** - The ability to select the type of schema an element is validated against, based on the values of the attribute of the element.
- **Open content** - Content models can use the `<openContent>` element to specify content models with open content. These content models allow elements not explicitly mentioned in the content model to appear in the document instance. It is as if wildcards were automatically inserted at appropriate points within the content model. A default may be set that causes all content models to be open unless specified otherwise.

To see the complete list with changes since version 1.0, go to http://www.w3.org/TR/xmlschema11-1/#ch_specs.
XML Schema 1.1 is intended to be mostly compatible with XML Schema 1.0 and to have approximately the same scope. It also addresses bug fixes and brings improvements that are consistent with the constraints on scope and compatibility.

**Note:**
An XML document conforming to a 1.0 schema can be validated using a 1.1 validator, but an XML document conforming to a 1.1 schema may not validate using a 1.0 validator.

If you are constrained to use XML Schema 1.0 (for example, if you develop schemas for a server that uses an XML Schema 1.0 validator that cannot be updated), change the default XML Schema version to 1.0. If you keep the default XML Schema version set to 1.1, the Content Completion Assistant (on page 3217) presents XML Schema 1.1 elements that you can insert accidentally in an 1.0 XML Schema. So even if you make a document invalid conforming with XML Schema 1.0, the validation process does not signal any issues.

To change the default XML Schema version, open the Preferences dialog box (Options > Preferences) (on page 124) and go to XML > XML Parser > XML Schema.

For more information about the XML Schema 1.1 support, watch our video demonstration:

https://www.youtube.com/embed/DAkrubQNm0w

*Related Information:* Setting the XML Schema Version (on page 1032)

### Setting the XML Schema Version

Oxygen XML Editor lets you set the version of the XML Schema you are editing either in the XML Schema preferences page, or through the versioning attributes. If you want to use the versioning attributes, set the minVersion and maxVersion attributes, from the http://www.w3.org/2007/XMLSchema-versioning namespace, on the schema root element.

**Note:**
The versioning attributes take priority over the XML Schema version defined in the preferences page.

<table>
<thead>
<tr>
<th>Versioning Attributes</th>
<th>XML Schema Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>minVersion = &quot;1.0&quot; maxVersion = &quot;1.1&quot;</td>
<td>1.0</td>
</tr>
<tr>
<td>minVersion = &quot;1.1&quot;</td>
<td>1.1</td>
</tr>
<tr>
<td>minVersion = &quot;1.0&quot; maxVersion = greater than &quot;1.1&quot;</td>
<td>The XML Schema version defined in the XML Schema preferences page (on page 238)</td>
</tr>
</tbody>
</table>
Table 36. Using the *minVersion* and *maxVersion* Attributes to Set the XML Schema Version (continued)

<table>
<thead>
<tr>
<th>Versioning Attributes</th>
<th>XML Schema Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>Not set in the XML Schema document</td>
<td>The XML Schema version defined in the XML Schema preferences page (on page 238)</td>
</tr>
</tbody>
</table>

To change the XML Schema version of the current document, use the **Change XML Schema version** action from the contextual menu. This is available both in the **Text** mode, and in the **Design** mode and opens the **Change XML Schema version** dialog box. The following options are available:

- **XML Schema 1.0** - Inserts the *minVersion* and *maxVersion* attributes on the *schema* element and gives them the values "1.0" and "1.1" respectively. Also, the namespace declaration (`xmlns:vc=http://www.w3.org/2007/XMLSchema-versioning`) is inserted automatically if it does not exist.
- **XML Schema 1.1** - Inserts the *minVersion* attribute on the *schema* element and gives it the value "1.1". Also, removes the *maxVersion* attribute if it exists and adds the versioning namespace (`xmlns:vc=http://www.w3.org/2007/XMLSchema-versioning`) if it is not declared.
- **Default XML Schema version** - Removes the *minVersion* and *maxVersion* attributes from the *schema* element. The default schema version, defined in the XML Schema preferences page, is used.

**Note:**
The **Change XML Schema version** action is also available in the informative panel presented at the top of the edited XML Schema. If you close this panel, it will no longer appear until you restore Oxygen XML Editor to its default options.

Oxygen XML Editor automatically uses the version set through the versioning attributes, or the default version if the versioning attributes are not declared, when proposing content completion elements and validating an XML Schema. Also, the XML instance validation against an XML Schema is aware of the versioning attributes defined in the XML Schema.

When you generate sample XML files from an XML Schema, Oxygen XML Editor takes into account the *minVersion* and *maxVersion* attributes defined in the XML Schema.

**Related Information:**
XML Schema 1.1 (on page 1030)

**Editing XQuery Documents**

XQuery is the query language for XML and is officially defined by a [W3C Recommendation document](#). Oxygen XML Editor provides support for XQuery 3.1, which is also backwards compatible with XQuery 3.0 and 1.0.

The many benefits of XQuery include:
• XQuery allows you to work in one common model no matter what type of data you are working with: relational, XML, or object data.
• XQuery is ideal for queries that must represent results as XML, to query XML stored inside or outside the database, and to span relational and XML sources.
• XQuery allows you to create many different types of XML representations of the same data.
• XQuery allows you to query both relational sources and XML sources, and create one XML result.

Related Information:
XQuery and Databases (on page 2074)

XQuery Validation

With Oxygen XML Editor, you can validate your documents before using them in your transformation scenarios. The validation uses the Saxon 10.6 PE, EE, or HE processor, or you can use some database engines (such as MarkLogic or eXist) if you installed them. Any other XQuery processor that offers an XQJ API implementation (on page 2068) can also be used. This is in conformance with the XQuery Working Draft. The processor is used in two cases: validation of the expression and execution. Although the execution implies a validation, it is faster to check the expression syntactically, without executing it. The errors that occurred in the document are presented in the messages view at the bottom of editor window, with a full description message. As with all error messages, if you click an entry, the line where the error appeared is highlighted.

Figure 350. XQuery Validation

Note:
If you choose a processor that does not support XQuery validation, Oxygen XML Editor displays a warning when trying to validate.

The Validation options button, available in the Document > Validate menu, allows quick access to the XQuery options (on page 253) in the Oxygen XML Editor preferences.
When you open an XQuery document from a connection that supports validation (for example, MarkLogic, or eXist), by default Oxygen XML Editor uses this connection for validation. If you open an XQuery file using a MarkLogic connection, the validation resolves imports better.

**Content Completion in XQuery**

Oxygen XML Editor provides content completion for keywords and all known XQuery functions and operators. The *Content Completion Assistant (on page 3217)* can be manually activated with the *(Ctrl + Space)* shortcut. The functions and operators are presented together with a description of the parameters and functionality, depending on the validation or transformation engine.

For some supported database engines such as MarkLogic and eXist, the content completion list offers the specific XQuery functions implemented by that engine. This feature is available when the XQuery file has an associated transformation scenario that uses one of these database engines or the XQuery validation engine is set to one of them via a validation scenario or in the *XQuery Preferences (on page 253)* page. For more information about the support for working with XQuery with regard to databases, see *XQuery and Databases (on page 2074)*.

The extension functions included in the Saxon engine are available on content completion if one of the following conditions are true:

- The edited file has a transformation scenario associated that uses as transformation engine Saxon 10.6 PE or Saxon 10.6 EE.
- The edited file has a validation scenario associated that use as validation engine Saxon 10.6 PE or Saxon 10.6 EE.
- The validation engine specified in *Preferences (on page 253)* is Saxon 10.6 PE or Saxon 10.6 EE.

If the Saxon namespace (http://saxon.sf.net) is mapped to a prefix, the functions are presented using this prefix. Otherwise, the default prefix for the Saxon namespace (saxon) is used.

If you want to use a function from a namespace mapped to a prefix, just type that prefix and the content completion displays all the XQuery functions from that namespace. When the default namespace is mapped to a prefix, the XQuery functions from this namespace offered by content completion are also prefixed. Otherwise, only the function name being used.

The content completion pop-up window presents all the variables and functions from both the edited XQuery file and its imports.
Syntax Highlighting in XQuery

Oxygen XML Editor supports syntax highlighting in Text mode to make it easier to read the semantics of the structured content by displaying each type of code in different colors and fonts.

To customize the colors or styles used for the syntax highlighting colors for XQuery files, follow these steps:

1. Open the Preferences dialog box (Options > Preferences) (on page 124).
2. Go to Editor > Syntax Highlight (on page 224).
3. Select and expand the XQuery/XPath section in the top pane.
4. Select the component you want to change and customize the colors or styles using the selectors to the right of the pane.

You can see the effects of your changes in the Preview pane.

Related Information:
Customize Syntax Highlight colors (on page 224)

Formatting and Indenting XQuery Documents

Editing XQuery documents may lead to large chunks of content that are not easily readable by human audience. Also, each developer may have a particular way of writing XQuery code. Oxygen XML Editor assists you in maintaining a consistent code writing style with the 🔄 Format and Indent action that is available in the Document > Source menu and also on the toolbar.

The 🔄 Format and Indent action achieves this by performing the following steps:

- Manages whitespaces, by collapsing or inserting space characters where needed.
- Formats complex expressions on multiple, more readable lines by properly indenting each of them. The amount of whitespaces that form an indent unit is controlled through one of the Indent with tabs and Indent size options from the Format Preferences page (on page 202).
Note:
These operations can be performed only if your XQuery document conforms with XQuery 1.0, 3.0, 3.1, or XQuery Update Facility 1.0 specifications. If the Format and Indent operation fails, the document is left unaltered and an error message is presented in the Results view (on page 548).

Folding in XQuery Documents

In a large XQuery document, the instructions enclosed in the '{' and '}' characters can be collapsed so that only the needed instructions remain in focus. The same folding features available for XML documents (on page 528) are also available in XQuery documents.

Figure 352. Folding in XQuery Documents

There is available the action Go to Matching Bracket Ctrl + Shift + G (Command + Shift + G on macOS) on contextual menu of XQuery editor for going to matching character when cursor is located at '{' character or '}' character. It helps for finding quickly matching character of current folding element (on page 3219).

XQuery Outline View

The XQuery document structure is presented in the Outline view. The outline tree presents the list of all the components (namespaces, imports, variables, and functions) from both the edited XQuery file and its imports and it allows quick access to components. By default, it is displayed on the left side of the editor. If the view is not displayed, it can be opened by selecting it from the Window > Show View menu.
The following actions are available in the Settings menu on the Outline view toolbar:

- **Selection update on cursor move**
  Controls the synchronization between Outline view and source document. The selection in the Outline view can be synchronized with the cursor moves or the changes performed in the XQuery editor. Selecting one of the components from the Outline view also selects the corresponding item in the source document.

- **Sort**
  Allows you to alphabetically sort the XQuery components.

**Show all components**
Displays all collected components starting from the current file. This option is set by default.

**Show only local components**
Displays the components defined in the current file only.

**Group by location/namespace/type**
Allows you to group the components by location, namespace, and type. When grouping by namespace, the main XQuery module namespace is presented first in the Outline view.
If you know the component name, you can search it in the **Outline** view by typing its name in the filter text field from the top of the view or directly on the tree structure. When you type the component name in the filter text field you can switch to the tree structure using the arrow keys of the keyboard, **Enter, Tab, Shift-Tab**. To switch from tree structure to the filter text field, you can use **Tab, Shift-Tab**.

**Tip:**
The search filter is case insensitive. The following wildcards are accepted:

- * - any string
- ? - any character
- , - patterns separator

If no wildcards are specified, the string to search is used as a partial match.

The upper part of the **Outline** view contains a filter box that allows you to focus on the relevant components. Type a text fragment in the filter box and only the components that match it are presented. For advanced usage you can use wildcard characters (such as * or ?) and separate multiple patterns with commas.

**XQuery Builder View**

The **XPath/XQuery Builder** view allows you to compose complex XQuery expressions and execute them over the currently edited XML document. You can use the `doc()` function to specify the source file that will have the expressions executed. When you connect to a database, the expressions are executed over that database. If you are using the **XPath/XQuery Builder** view and the current file is an XSLT document, Oxygen XML Editor executes the expressions over the XML document in the associated scenario.

If the view is not displayed, it can be opened by selecting it from the **Window > Show View** menu.

The upper part of the view contains the following actions:

**XPath version chooser drop-down menu**

A drop-down menu that allows you to select the type of the expression you want to execute. You can choose between:

- XPath 1.0 (Xerces-driven)
- XPath 2.0, XPath 2.0 SA, XPath 3.1, XPath 3.1 SA, Saxon-HE XQuery, Saxon-PE XQuery, or Saxon-EE XQuery (all of them are Saxon-driven)
- Custom connection to XML databases that can execute XQuery expressions
Note:
The results returned by XPath 2.0 SA and XPath 3.1 SA have a location limited to the line number of the start element (there are no column information and no end specified).

Note:
Oxygen XML Editor uses Saxon to execute XPath 3.1 expressions. Since Saxon implements a part of the 3.1 functions, when using a function that is not implemented, Oxygen XML Editor returns a compilation error.

Execute XPath button
Use this button to start the execution of the XPath or XQuery expression you are editing. The result of the execution is displayed in the Results view (on page 548).

Favorites button
Allows you to save certain expressions that you can later reuse. To add an expression as a favorite, click this button and enter a name for it. The star turns yellow to confirm that the expression was saved. Expand the drop-down menu next to the star button to see all your favorites. Oxygen XML Editor automatically groups favorites in folders named after the method of execution.

History drop-down menu
Keeps a list of the last 15 executed XPath or XQuery expressions. Use the Clear history action from the bottom of the list to remove them.

Settings drop-down menu
Contains the following three options:

Update on cursor move
When selected and you navigate through a document, the XPath expression corresponding to the XML node at the current cursor position is displayed. For JSON documents, it displays the XPath expression for the current property.

Evaluate as you type
When you select this option, the XPath expression you are composing is evaluated in real time.

Note:
This option and the automatic validation are disabled when you edit huge documents (on page 469) or when the scope is other than Current file.
Options

Opens the Preferences page of the currently selected processing engine.

XPath scope menu

Oxygen XML Editor allows you to define a scope for the XPath operation to be executed. You can choose where the XPath expression will be executed:

- **Current file** - Currently selected file only.
- **Project** - All the files in the project.
- **Selected project resources** - The files selected in the project.
- **All opened files** - All files that are opened in the application.
- **Current DITA Map hierarchy** - All resources referenced in the currently selected DITA map that is open in the DITA Maps Manager view (on page 2897).
- **Opened archive** - Files that are opened in the Archive Browser view (on page 2008).
- **Working sets** - The selected working sets (on page 3224).

At the bottom of the scope menu the following scope configuration actions are available:

- **Configure XPath working sets** - Allows you to configure and manage collections of files and folders, encapsulated in logical containers called working sets (on page 3224).
- **XPath file filter** - You can filter the files from the selected scope that will have the XPath expression executed. By default, the XPath expression will be executed only on XML or JSON files, but you can also define a set of patterns that will filter out files from the current scope. If you select the **Include archive** option, the XPath expression will be also executed on the files in any archive (including EPUB and DocX) found at the current scope.
While you edit an XPath or XQuery expression, Oxygen XML Editor assists you with the following features:

- **Content Completion Assistant** (on page 3217) - It offers context-dependent proposals and takes into account the cursor position in the document you are editing. The set of functions proposed by the Content Completion Assistant also depends on the engine version. Select the engine version from the drop-down menu available in the toolbar.

- Syntax Highlighting - Allows you to identify the components of an expression. To customize the colors of the components of the expression, open the Preferences dialog box (Options > Preferences) (on page 124) and go to Editor > Syntax Highlight (on page 224).

- Automatic validation of the expression as you type.

  **Note:**

  When you type invalid syntax, a red serrated line underlines the invalid fragments.

- Function signature and documentation balloon, when the cursor is located inside a function.
The usual edit actions (Cut, Copy, Paste, Select All, Undo, Redo) are available in the contextual menu of the top editable part of the view.

**XQuery Input View**

The structure of the source documents of an edited XQuery is displayed in a tree form in a view called the **XQuery Input** view. If the view is not displayed, it can be opened by selecting it from the **Window > Show View** menu. The tree nodes represent the elements of the documents.

You can use the **XQuery Input** view to drag and drop a node into the editing area to quickly insert XQuery expressions.

**Example:**

For the following XML documents:

```xml
<movies>
  <movie id="1">
    <title>The Green Mile</title>
    <year>1999</year>
  </movie>
  <movie id="2">
    <title>Taxi Driver</title>
    <year>1976</year>
  </movie>
</movies>

<reviews>
  <review id="100" movie-id="1">
  </review>
</reviews>
```
<review id="101" movie-id="1">
  <rating>5</rating>
  <comment>It is made after a great Stephen King book.</comment>
  <author>Paul</author>
</review>

<review id="104" movie-id="2">
  <rating>4</rating>
  <comment>Robert De Niro is my favorite actor.</comment>
  <author>Maria</author>
</review>

and the following XQuery:

```xquery
let $review := doc("reviews.xml")
for $movie in doc("movies.xml")/movies/movie
let $movie-id := $movie/@id
return
<movie id="{$movie/@id}">
  {$movie/title}
  {$movie/year}
  <maxRating>
    ...
  </maxRating>
</movie>
```

If you drag the **review** element and drop it between the braces, the following pop-up menu is displayed:

```xquery
let $review := doc("reviews.xml")
for $movie in doc("movies.xml")/movies/movie
let $movie-id := $movie/@id
return
<movie id="{$movie/@id}">
  {$movie/title}
  {$movie/year}
  <maxRating>
    ...
  </maxRating>
</movie>
```

Select **FLWOR review**, the resulting document will look like this:
Generating HTML Documentation for an XQuery Document

To generate HTML documentation for an XQuery document, use the XQuery Documentation dialog box. It is opened with the XQuery Documentation action that is available from the Tools > Generate Documentation menu or from the Generate Documentation submenu in the contextual menu of the Project view (on page 402). You can also open the tool by using the Generate Documentation toolbar button.

The dialog box allows you to configure a set of parameters for the process of generating the HTML documentation.

The following options are available:

- **Input** - The full path to the XQuery file must be specified in one of the two fields in this section:
  - **URL** - The URL of the file to be used for generating the documentation.
  - **Folder** - The directory that contains the files to be used for generating the documentation. You can also specify the XQuery file extensions to be searched for in the specified directory.
• Default function namespace - Optional URI for the default namespace for the submitted XQuery.

• Predefined function namespaces - Optional, engine-dependent, predefined namespaces that the submitted XQuery refers to. They allow the conversion to generate annotation information to support the presentation component hypertext linking (only if the predefined modules have been loaded into the local qxDoc XML repository).

• Open in Browser/System Application - Select this option if you want the result to be opened in the system application associated with that file type.

Note: To set the browser or system application that will be used, open the Preferences dialog box (Options > Preferences) (on page 124), go to Global, and set it in the Default Internet browser field. This will take precedence over the default system application settings.

• Output - Allows you to specify where the generated documentation is saved on disk.

Transforming XML Documents Using XQuery

XQuery is similar to XSL stylesheets, both being capable of transforming an XML input into another format. You specify the input URL when you define the transformation scenario (on page 1464). The result can be saved and opened in the associated application. You can even run a FO processor (on page 1526) on the output of an XQuery. The transformation scenarios may be shared between many XQuery files, are exported (on page 323) together with the XSLT scenarios and can be managed in the Configure Transformation Scenario dialog box (on page 1548), or in the Scenarios view (on page 1555). The transformation can be performed on the XML document specified in the XML URL field, or, if this field is empty, the documents referenced from the query expression. The parameters of XQuery transforms must be set in the Parameters dialog box (on page 1464). Parameters that are in a namespace must be specified using the qualified name (for example, a param parameter in the http://www.oxygenxml.com/ns namespace must be set with the name {http://www.oxygenxml.com/ns}param).

The transformation uses one of the Saxon 10.6 HE, Saxon 10.6 PE, Saxon 10.6 EE processors, a database connection (details can be found in the Working with Databases (on page 2015) chapter - in the XQuery transformation (on page 2075) section) or any XQuery processor that provides an XQJ API implementation.

The Saxon 10.6 EE processor also supports XQuery 3.1 transformations.

Related Information:
XQuery and Databases (on page 2074)

Display XQuery Result in Sequence View

The result of an XQuery executed on a database can be very large and sometimes only a part of the full result is needed. To avoid the long time necessary for fetching the full result, select the Present as a sequence option (on page 1486) in the Output tab of the Edit scenario dialog box. This option fetches only the first
chunk of the result. Clicking the **More results available** label that is displayed at the bottom of the **Sequence** view fetches the next chunk of results.

The size of a chunk can be set with the **Size limit of Sequence view** option (on page 253). The **XQuery options** button from the **More results available** label provides a quick access to this option by opening the **XQuery preferences page** (on page 253) where the option can be modified.

**Figure 357. XQuery transformation result displayed in Sequence view**

![Edit scenario](image1)

A chunk of the XQuery transformation result is displayed in the **Sequence** view.

**Figure 358. XQuery transformation result displayed in Sequence view**

![Sequence view](image2)
Tip:
You can right-click the results in the Sequence view and if the item is an XML element, the Go to definition action will open the XML file from where the queried node was obtained.

Advanced Saxon HE/PE/EE XQuery Transformation Options

The XQuery transformation scenario allows you to configure advanced options that are specific for the Saxon HE (Home Edition), PE (Professional Edition), and EE (Enterprise Edition) engines. They are the same options as those in the Saxon HE/PE/EE preferences page (on page 254) but they are configured as a specific set of transformation options for each transformation scenario, while the values set in the preferences page apply as global options. The advanced options configured in a transformation scenario override the global options (on page 3219) defined in the preferences page.

Saxon-HE/PE/EE Options

The advanced options for Saxon 10.6 Home Edition (HE), Professional Edition (PE), and Enterprise Edition (EE) are as follows:

Use a configuration file ("-config")
Sets a Saxon 10.6 configuration file that is used for XQuery transformation and validation scenarios.

Enable Optimizations ("-opt")
This option is selected by default, which means that optimization is enabled. If not selected, the optimization is suppressed, which is helpful when reducing the compiling time is important, optimization conflicts with debugging, or optimization causes extension functions with side-effects to behave unpredictably.

Use linked tree model ("-tree:linked")
This option activates the linked tree model.

Strip whitespaces ("-strip")
Specifies how the strip whitespaces operation is handled. You can choose one of the following values:

- **All ("all")** - Strips all whitespace text nodes from source documents before any further processing, regardless of any @xml:space attributes in the source document.
- **Ignore ("ignorable")** - Strips all ignorable whitespace text nodes from source documents before any further processing, regardless of any @xml:space attributes in the source document. Whitespace text nodes are ignorable if they appear in elements defined in the DTD or schema as having element-only content.
- **None ("none")** - Strips no whitespace before further processing.

Enable profiling ("-TP")
If selected, profiling of the execution time in a query is enabled. The corresponding text field is used to specify the path to the output file where the profiling information will be saved. As long as the option is selected, and the output file specified, it will gather timed tracing information and create a profile report to the specified file.

Note:
The profiling support works only if the Present as a sequence transformation option (on page 1486) is not set.

Saxon-PE/EE Options

The following advanced options are specific for Saxon 10.6 Professional Edition (PE) and Enterprise Edition (EE) only:

Allow calls on extension functions ("-ext")

If selected, calls on external functions are allowed. Selecting this option is not recommended in an environment where untrusted stylesheets may be executed. It also disables user-defined extension elements and the writing of multiple output files, both of which carry similar security risks.

Saxon-EE Options

The advanced options that are specific for Saxon 10.6 Enterprise Edition (EE) are as follows:

Validation of the source file ("-val")

Requests schema-based validation of the source file and of any files read using document() or similar functions. It can have the following values:

- **Schema validation ("strict")** - This mode requires an XML Schema and allows for parsing the source documents with strict schema-validation enabled.
- **Lax schema validation ("lax")** - If an XML Schema is provided, this mode allows for parsing the source documents with schema-validation enabled but the validation will not fail if, for example, element declarations are not found.
- **Disable schema validation** - This specifies that the source documents should be parsed with schema-validation disabled.

Validation errors in the result tree treated as warnings ("-outval")

Normally, if validation of result documents is requested, a validation error is fatal. Selecting this option causes such validation failures to be treated as warnings.

Write comments for non-fatal validation errors of the result document

The validation messages for non-fatal errors are written (wherever possible) as a comment in the result document itself.

Enable XQuery update ("-update:(on|off)")
This option controls whether or not XQuery update syntax is accepted. The default value is off.

**Backup files updated by XQuery ("-backup:(on|off)")**

If selected, backup versions for any XML files updated with an XQuery Update are generated. This option is available when the Enable XQuery update option is selected.

**Other Options**

**Initializer class**

Equivalent to the `-init` Saxon command-line argument. The value is the name of a user-supplied class that implements the `net.sf.saxon.lib.Initializer` interface. This initializer is called during the initialization process, and may be used to set any options required on the configuration programmatically. It is particularly useful for tasks such as registering extension functions, collations, or external object models, especially in Saxon-HE where the option cannot be set via a configuration file. Saxon only calls the initializer when running from the command line, but the same code may be invoked to perform initialization when running user application code.

**Updating XML Documents using XQuery Update 1.0**

Using the bundled Saxon 10.6 EE XQuery processor Oxygen XML Editor offers support for XQuery Update 1.0. The XQuery Update Facility provides expressions that can be used to make persistent changes to instances of the XQuery 1.0 and XPath 2.0 Data Model. Thus, besides querying XML documents, you can modify them using the various insert/delete/modify/create methods available in the XQuery Update 1.0 standard.

Choose Saxon 10.6 EE as a transformer in the scenario associated with the XQuery files containing update statements and Oxygen XML Editor will notify you if the update was successful.

**Example: Using XQuery Update to modify a tag name in an XML file**

```xml
rename node doc("books.xml")//publisher[1]//book[1] as "firstBook"
```

**XQuery Unit Test (XSpec)**

XSpec is a behavior driven development (BDD) framework for XSLT, XQuery, and Schematron. XSpec consists of syntax for describing the behavior of your XSLT, XQuery, or Schematron code, and some code that enables you to test your code against those descriptions.

**Creating an XQuery Unit Test**

To create a XQuery Unit Test, go to **File > New > XQuery Unit Test**. This is simple document template to help you get started.
Running an XQuery Unit Test

To run a Unit Test, open the XSpec file in an editor and click on the main toolbar. This will run the built-in Run XSpec Test transformation scenario that is defined in the XSpec framework (on page 3219).

Testing an XQuery file

An XSpec file contains one or more test scenarios.

**Example:**

Suppose you have this XQuery function that takes a string as its argument. The first character of the string passed to the function is capitalized and concatenated to the rest of the string. Finally, the new string with the first character capitalized is returned to the calling method.

```xml
module namespace functx = "http://www.functx.com";

declare function functx:capitalize-first ($arg as xs:string?) as xs:string? {
  concat(upper-case(substring($arg, 1, 1)),
           substring($arg, 2))
};
```

The XSpec test invokes the XQuery function and passes the string `hello` as a parameter. The expected value that should be returned by the function (the string `Hello`) is contained in the @select attribute of the `x:expect` element.

```xml
<x:scenario label="Calling function capitalize-first">
  <x:call function="functx:capitalize-first">
    <x:param select="'hello'"/>
  </x:call>
  <x:expect label="should capitalize the first character of the string" select="'Hello'"/>
</x:scenario>
```

For more details about how to write XQuery tests and various samples, see https://github.com/xspec/xspec/wiki/Getting-Started-with-XSpec-and-XQuery.

Editing WSDL Documents

WSDL is an XML format for describing network services as a set of endpoints operating on messages containing either document-oriented or procedure-oriented information. The operations and messages are described abstractly, and then bound to a concrete network protocol and message format to define an endpoint. Related concrete endpoints are combined into abstract endpoints (services).
Oxygen XML Editor provides a special type of editor dedicated to WSDL documents. The WSDL editor offers support for validation, a specialized Content Completion Assistant (on page 3217), a component oriented Outline view (on page 1055), searching and refactoring operations, and support to generate documentation.

Both WSDL version 1.1 and 2.0 are supported and SOAP versions 1.1 and 1.2. That means that in the location where a SOAP extension can be inserted the Content Completion Assistant offers elements from both SOAP 1.1 and SOAP 1.2. Validation of SOAP requests is executed first against a SOAP 1.1 schema and then against a SOAP 1.2 schema. In addition to validation against the XSD schemas, Oxygen XML Editor also checks if the WSDL file conforms with the WSDL specification (available only for WSDL 1.1 and SOAP 1.1).

In the following example you can see how the errors are reported.

For more information about the WSDL editing support in Oxygen XML Editor, watch our video demonstration:

https://www.youtube.com/embed/OS5Ucm9b8sY

Related Information:

Editing XML Documents in Text Mode (on page 517)

Modular Contextual WSDL Editing Using 'Main Files' Support

Smaller interrelated modules that define a complex WSDL structure cannot be correctly edited or validated individually, due to their interdependency with other modules. Oxygen XML Editor provides the support for defining the main module (or modules), allowing you to edit any of the imported/included files in the context of the larger WSDL structure.

You can set a main WSDL document either using the main files support from the Project view (on page 418), or using a validation scenario.

To set a main file using a validation scenario, add validation units that point to the main modules. Oxygen XML Editor warns you if the current module is not part of the dependencies graph computed for the main WSDL document. In this case, it considers the current module as the main WSDL document.

The advantages of editing in the context of a main file (on page 3220) include:
• Correct validation of a module in the context of a larger WSDL structure.
• *Content Completion Assistant (on page 3217)* displays all components valid in the current context.
• The *Outline view (on page 1055)* displays the components collected from the entire WSDL structure.

**Note:**
When you edit an XML schema document that has a WSDL document set as the main file, the validation operation is performed over the main WSDL document.

For more information about editing WSDL documents in the *main files* context, watch our video demonstration:

https://www.youtube.com/embed/gn_YPD5xDCo

**Validating WSDL Documents**

By default, WSDL files are validated as you type. To change this, open the *Preferences* dialog box *(Options > Preferences) (on page 124)*, go to Editor > Document Checking, and deselect the *Enable automatic validation* option *(on page 226)*.

To validate a WSDL document manually, select the ✅ Validate action from the ✅ Validation toolbar drop-down menu or the Document > Validate menu. The validation problems are highlighted directly in the editor, making it easy to locate and fix any issues.

**Content Completion Assistance in WSDL Documents**

The *Content Completion Assistant (on page 3217)* is a powerful feature that enhances the editing of WSDL documents. It helps you define WSDL components by proposing context-sensitive element names. It can be manually activated with the Ctrl + Space (Command + Space on macOS) shortcut.

Another important capability of the *Content Completion Assistant* is to propose references to the defined components when you edit attribute values. For example, when you edit the @type attribute of a `<binding>` element, the *Content Completion Assistant* proposes all the defined port types. Each proposal that the *Content Completion Assistant* offers is accompanied by a documentation hint.

**Note:**
XML schema-specific elements and attributes are offered when the current editing context is the internal XML schema of a WSDL document.
Note:
If you are using the concept of main files (on page 3220) to import/include modules, the Content Completion Assistant collects its components starting from the main files. The main files can be defined in the project or in the associated validation scenario. For more information about the Main Files support in Oxygen XML Editor, see Defining Main Files at Project Level (on page 418).

Namespace prefixes in the scope of the current context are presented at the top of the content completion assistance window to speed up the insertion into the document of prefixed elements.

For the common namespaces, such as XML Schema namespace (http://www.w3.org/2001/XMLSchema) or SOAP namespace (http://schemas.xmlsoap.org/wsdl/soap/), Oxygen XML Editor provides an easy mode to declare them by proposing a prefix for these namespaces.

**WSDL Syntax Highlighting**

Oxygen XML Editor supports syntax highlighting in Text mode to make it easier to read the semantics of the structured content by displaying each type of code in different colors and fonts.

To customize the colors or styles used for the syntax highlighting colors for WSDL files, follow these steps:

1. Open the Preferences dialog box (Options > Preferences) (on page 124).
2. Go to Editor > Syntax Highlight (on page 224).
3. Select and expand the XML section in the top pane.
4. Select the component you want to change and customize the colors or styles using the selectors to the right of the pane.
5. Select the XML tab in the Preview pane to see the effects of your changes.

**Tip:**
Oxygen XML Editor also allows you to specify syntax highlighting colors for specific XML elements and attributes with specific namespace prefixes. This can be done in the Editor > Syntax Highlight > Elements/Attributes by Prefix preferences page (on page 225).

Related Information:
Customize Syntax Highlight colors (on page 224)

### WSDL Outline View

The Outline view for WSDL documents displays the list of all the components (services, bindings, port types and so on) of the currently open WSDL document along with the components of its imports.

If you use the Main Files support (on page 418), the Outline view collects the components of a WSDL document starting from the main files of the current document.

By default, it is displayed on the left side of the editor. If the view is not displayed, it can be opened by selecting it from the Window > Show View menu.

**Figure 362. WSDL Outline View**
The **Outline** view can display both the components of the current document and its XML structure, organized in a tree-like fashion. You can switch between the display modes by using the ![Show XML structure] and ![Show components] actions in the ![Settings] menu on the **Outline** view toolbar. The following actions are available:

**Filter returns exact matches**

The text filter of the **Outline** view returns only exact matches.

**Selection update on cursor move**

Controls the synchronization between the **Outline** view and the current document. The selection in the **Outline** view can be synchronized with the cursor moves or the changes in the WSDL editor. Selecting one of the components from the **Outline** view also selects the corresponding item in the current document.

When the ![Show components] option is selected, the following actions are available:

- ![Show XML structure]
  
  Displays the XML structure of the current document in a tree-like manner.

- ![Sort]
  
  Sorts the components in the **Outline** view alphabetically.

**Show all components**

Displays all the components that were collected starting from current document or from the main document, if it is defined.

**Show referable components**

Displays all the components that you can reference from the current document.

**Show only local components**

Displays the components defined in the current file only.

**Group by location**

Groups the WSDL components by their location.

**Group by type**

Groups the WSDL components by their type.

**Group by namespace**

Groups the WSDL components by their namespace.

**Note:**

By default, all the three grouping criteria are active.

When the ![Show XML structure] option is selected, the following actions are available:
Show components

Switches the Outline view to the components display mode.

Flat presentation mode of the filtered results

When active, the application flattens the filtered result elements to a single level.

Show comments and processing instructions

Show/hide comments and processing instructions in the Outline view.

Show element name

Show/hide element name.

Show text

Show/hide additional text content for the displayed elements.

Show attributes

Show/hide attribute values for the displayed elements. The displayed attribute values can be changed from the Outline preferences panel (on page 306).

Configure displayed attributes

Displays the XML Structured Outline preferences page (on page 306).

The following contextual menu actions are available in the Outline view when the Show components option is selected in the Settings menu:

Edit Attributes

Opens a dialog box that allows you to edit the attributes of the currently selected component.

Cut

Cuts the currently selected component.

Copy

Copies the currently selected component.

Delete

Deletes the currently selected component.

Search references

Searches for the references of the currently selected component.

Search references in

Searches for the references of the currently selected component in the context of a scope that you define.

Component dependencies
Opens the **Component Dependencies view (on page 1062)** that displays the dependencies of the currently selected component.

**Resource Hierarchy**

Opens the **Resource Hierarchy/Dependencies view (on page 1059)** that displays the hierarchy for the currently selected resource.

**Resource Dependencies**

Opens the **Resource Hierarchy/Dependencies view (on page 1059)** that displays the dependencies of the currently selected resource.

**Rename Component in**

Renames the currently selected component in the context of a scope that you define.

The following contextual menu actions are available in the **Outline** view when the **Show XML structure** option is selected in the **Settings** menu:

**Append Child**

Displays a list of elements that you can insert as children of the current element.

**Insert Before**

Displays a list of elements that you can insert as siblings of the current element, before the current element.

**Insert After**

Displays a list of elements that you can insert as siblings of the current element, after the current element.

**Edit Attributes**

Opens a dialog box that allows you to edit the attributes of the currently selected component.

**Toggle Comment**

Comments/uncomments the currently selected element.

**Search references**

Searches for the references of the currently selected component.

**Search references in**

Searches for the references of the currently selected component in the context of a scope that you define.

**Component dependencies**

Opens the **Component Dependencies view (on page 1062)** that displays the dependencies of the currently selected component.

**Rename Component in**
 Renames the currently selected component in the context of a scope that you define.

**Cut**

Cuts the currently selected component.

**Copy**

Copies the currently selected component.

**Delete**

Deletes the currently selected component.

**Expand More**

Expands the structure of a component in the **Outline** view.

**Collapse All**

Collapses the structure of all the component in the **Outline** view.

To switch from the tree structure to the text filter, use **Tab** and **Shift-Tab**.

---

**Tip:**

The search filter is case insensitive. The following wildcards are accepted:

- * - any string
- ? - any character
- , - patterns separator

If no wildcards are specified, the string to search is used as a partial match.

The content of the **Outline** view and the editing area are synchronized. When you select a component in the **Outline** view, its definition is highlighted in the editing area.

**WSDL Resource Hierarchy/Dependencies View in WSDL Documents**

The **Resource Hierarchy/Dependencies** view displays the hierarchy or dependencies for a WSDL resource. If the view is not displayed, it can be opened by selecting it from the **Window > Show View** menu.

**Note:**

The hierarchy of a WSDL resource includes the hierarchy of imported XML Schema resources. The dependencies of an XML Schema resource present the WSDL documents that import the schema.

To view the hierarchy or dependencies of a WSDL document, select the document in the **Project view (on page 402)** and choose **Resource Hierarchy** or **Resource Dependencies** from the contextual menu.
The following actions are available on the toolbar of the Resource Hierarchy/Dependencies view:

- **Refresh**
  Refreshes the Hierarchy/Dependencies structure.

- **Stop**
  Stops the hierarchy/dependencies computing.

- **Show Hierarchy**
  Allows you to choose a resource to compute the hierarchy structure.

- **Show Dependencies**
  Allows you to choose a resource to compute the dependencies structure.

- **Configure dependencies search scope**
  Allows you to configure a scope to compute the dependencies structure. There is also an option for automatically using the defined scope for future operations.

- **History**
  Provides access to the list of previously computed dependencies. Use the **Clear history** button to remove all items from this list.

The contextual menu for a resource listed in the Resource Hierarchy/Dependencies view contains the following actions:

- **Open**
Opens the resource. You can also double-click a resource in the Hierarchy/Dependencies structure to open it.

**Go to reference**

Opens the source document where the resource is referenced.

**Copy location**

Copies the location of the resource.

**Move resource**

Moves the selected resource.

**Rename resource**

Renames the selected resource.

**Resource Hierarchy**

Shows the hierarchy for the selected resource.

**Resource Dependencies**

Shows the dependencies for the selected resource.

**Add to Main Files**

Adds the currently selected resource in the Main Files directory (on page 418).

**Expand More**

Expands more of the children of the selected resource from the Hierarchy/Dependencies structure.

**Collapse All**

Collapses all children of the selected resource from the Hierarchy/Dependencies structure.

**Tip:**

When a recursive reference is encountered in the Hierarchy view, the reference is marked with a special icon ☹.

**Note:**

The Move resource or Rename resource actions give you the option to update the references to the resource (on page 1062).

**Related Information:**

Modular Contextual XML Editing Using 'Main Files' Support (on page 829)

Search and Refactor Operations Scope (on page 831)
Moving/Renaming WSDL Resources

You can move and rename a resource presented in the Resource/Hierarchy Dependencies view, using the Rename resource and Move resource refactoring actions from the contextual menu.

When you select the Rename action in the contextual menu of the Resource/Hierarchy Dependencies view, the Rename resource dialog box is displayed. The following fields are available:

- **New name** - Presents the current name of the edited resource and allows you to modify it.
- **Update references of the renamed resource(s)** - Select this option to update the references to the resource you are renaming. A Preview option is available that allows you to see what will be updated before selecting Rename to process the operation.

When you select the Move action from the contextual menu of the Resource/Hierarchy Dependencies view, the Move resource dialog box is displayed. The following fields are available:

- **Destination** - Presents the path to the current location of the resource you want to move and gives you the option to introduce a new location.
- **New name** - Presents the current name of the moved resource and gives you the option to change it.
- **Update references of the moved resource(s)** - Select this option to update the references to the resource you are moving, in accordance with the new location and name. A Preview option is available that allows you to see what will be updated before selecting Move to process the operation.

WSDL Component Dependencies View

The Component Dependencies view allows you to see the dependencies for a selected component. This is helpful if you want to see where components are used in the entire hierarchy. For example, if you want to find all the references where a given component is used.

If the view is not displayed, it can be opened by selecting it from the Window > Show View menu.

To see the dependencies of a WSDL component:

1. Right-click the desired component in the editor or Outline view.
2. Select the Component Dependencies action from the contextual menu.

This action is available for all WSDL components (messages, port types, operations, bindings, and so on).

If a component contains multiple references, a small table is displayed at the bottom of the view that contains all the references. When a recursive reference is encountered, it is marked with a special icon.
The **Component Dependencies** view includes the following toolbar actions:

- **Refresh**
  
  Refreshes the dependencies structure.

- **Stop**
  
  Stops the dependency computation.

- **Configure**
  
  Allows you to choose the search scope for computing the dependencies structure. This is helpful for making sure all imported/included resources are computed.

- **History**
  
  Allows you to select from a list of the most recently used dependency computations.

In addition, the following actions are available in the contextual menu:

- **Go to First Reference**
  
  Selects the first reference of the currently selected component in the dependencies tree.

- **Go to Component**
  
  Shows the definition of the currently selected component in the dependencies tree.

**Related Information:**

- [Searching and Refactoring Operations Scope in WSDL Documents](on page 1066)

**Highlight Component Occurrences in WSDL Documents**

When you position your mouse cursor over a component in a WSDL document, Oxygen XML Editor searches for the component declaration and all its references and highlights them automatically.
Customizable colors are used: one for the component definition and another one for component references. Occurrences are displayed until another component is selected.

To change the default behavior of Highlight Component Occurrences, open the Preferences dialog box (Options > Preferences) (on page 124) and go to Editor > Mark Occurrences. You can also trigger a search using the Search > Search Occurrences in File () action from contextual menu. Matches are displayed in separate tabs of the Results view (on page 548).

Searching and Refactoring Operations in WSDL Documents

Search Actions

The following search actions are available from the Search submenu in the contextual menu of the current editor or from the Document > References menu:

**Search References**

Searches all references of the item found at current cursor position in the defined scope, if any. If a scope is defined, but the currently edited resource is not part of the range of resources determined by this, a warning dialog box is displayed and you have the possibility to define another search scope.

**Search References in**

Searches all references of the item found at current cursor position in the file or files that you specify when define a scope in the Search References dialog box.

**Search Declarations**

Searches all declarations of the item found at current cursor position in the defined scope if any. If a scope is defined, but the currently edited resource is not part of the range of resources determined by this, a warning dialog box will be displayed and you have the possibility to define another search scope.

**Search Declarations in**

Searches all declarations of the item found at current cursor position in the file or files that you specify when you define a scope for the search operation.

**Search Occurrences in File**

Searches all occurrences of the item at the cursor position in the currently edited file.

The following action is available from the Document > Schema menu:

**Go to Definition**

Takes you to the location of the definition of the current item.
Note:
You can also use the Ctrl + Single-Click (Command + Single-Click on macOS) shortcut on a reference to display its definition.

Refactoring Actions

The following refactoring actions are available from the Refactoring submenu from the Document > Refactoring menu or in the contextual menu of the current editor:

Rename Component

Allows you to rename the current component (in-place). The component and all its references in the document are highlighted with a thin border and the changes you make to the component at the cursor position are updated in real time to all occurrences of the component. To exit the in-place editing, press the Esc or Enter key on your keyboard.

Rename Component in

Opens a dialog box that allows you to rename the selected component by specifying the new component name and the files to be affected by the modification. If you click the Preview button, you can view the files to be affected by the action.

Figure 365. Rename Identity Constraint Dialog Box
Searching and Refactoring Operations Scope in WSDL Documents

The **scope** is a collection of documents that define the context of a search and refactor operation. To control it you can use the 📐 **Change scope** operation, available in the **Quick Assist** action set or on the **Resource Hierarchy/Dependency View** toolbar. You can restrict the scope to the current project or to one or multiple **working sets (on page 3224)**. The **Use only Main Files, if enabled** checkbox allows you to restrict the scope of the search and refactor operations to the resources from the **Main Files** directory. Click **read more** for details about the **Main Files support (on page 418)**.

![Figure 366. Change Scope Dialog Box](image)

The scope you define is applied to all future search and refactor operations until you modify it. Contextual menu actions allow you to add or delete files, folders, and other resources to the **working set (on page 3224)** structure.

**Quick Assist Support in WSDL Documents**

The **Quick Assist feature (on page 3223)** is activated automatically when the cursor is positioned over the name of a component. It is accessible via a yellow bulb icon (💡) placed at the current line in the stripe on the left side of the editor. Also, you can invoke the **quick assist** menu by using the **Alt + 1** (**Meta + Alt + 1** on macOS) keyboard shortcuts.
The *Quick Assist* support offers direct access to the following actions:

- **Rename Component in**
  Renames the component and all its dependencies.

- **Search Declarations**
  Searches the declaration of the component in a predefined scope. It is available only when the context represents a component name reference.

- **Search References**
  Searches all references of the component in a predefined scope.

- **Component Dependencies**
  Searches the component dependencies in a predefined scope.

- **Change Scope**
  Configures the scope that will be used for future search or refactor operations.

- **Rename Component**
  Allows you to rename the current component in-place.

- **Search Occurrences**
  Searches all occurrences of the component within the current file.

### Generating Documentation for WSDL Documents

You can use Oxygen XML Editor to generate detailed documentation for the components of a WSDL document in HTML format. You can select the WSDL components to include in your output and the level of details to present for each of them. Also, the components are hyperlinked. You can also generate the documentation in a custom output format *(on page 1072)* by using a custom stylesheet.

**Note:**

The WSDL documentation includes the XML Schema components that belong to the internal or imported XML schemas.
To generate documentation for a WSDL document, select **WSDL Documentation** from the Tools > Generate Documentation menu or from the Generate Documentation submenu in the contextual menu of the Project view (on page 402). You can also open the tool by using the Generate Documentation toolbar button.

Figure 368. WSDL Documentation Dialog Box

![WSDL Documentation Dialog Box](image)

The **Input URL** field of the dialog box must contain the full path to the WSDL document that you want to generate documentation for. The WSDL document may be a local or a remote file. You can specify the path to the WSDL file by entering it in the text field, or by using the Insert Editor Variables button or the options in the Browse drop-down menu.

**Output Tab**

The following options are available in the **Output** tab:

- **Format** - Allows you to choose between the following formats:
  - **HTML** - The documentation is generated in HTML output format (on page 1071).
  - **Custom** - The documentation is generated in a custom output format (on page 1072), allowing you to control the output. Click the Options button to open a Custom format options dialog box where you can specify a custom stylesheet for creating the output. There is also an option to Copy additional resources to the output folder and you can select the path to the additional Resources that you want to copy. You can also choose to keep the intermediate XML files created during the documentation process by deselecting the Delete intermediate XML file option.
• **Output file** - You can specify the path of the output file by entering it in the text field, or by using the *Insert Editor Variables* button or the options in the *Browse* drop-down menu.

• **Split output into multiple files** - Instructs the application to split the output into multiple files. For large WSDL documents, choosing a different split criterion may generate smaller output files providing a faster documentation browsing. You can choose to split them by namespace, location, or component name.

• **Open in Browser/System Application** - Opens the result in the system application associated with the output file type.

---

**Note:**

To set the browser or system application that will be used, open the *Preferences* dialog box (*Options > Preferences*) (on page 124), go to *Global*, and set it in the *Default Internet browser* field. This will take precedence over the default system application settings.

---

• **Keep only the annotations with xml:lang set to** - The generated output will contain only the annotations with the `@xml:lang` attribute set to the selected language. If you choose a primary language code (for example, `en` for English), this includes all its possible variations (`en-us`, `en-uk`, etc.).

---

**Setting Tab**

When you generate documentation for a WSDL document, you can choose what components to include in the output and the details to be included in the documentation.

**Figure 369. Settings Tab of the WSDL Documentation Dialog Box**
The **Settings** tab allows you to choose whether or not to include the following:

- **Components**
  - **Services** - Specifies whether or not the generated documentation includes the WSDL services.
  - **Bindings** - Specifies whether or not the generated documentation includes the WSDL bindings.
  - **Port Types** - Specifies whether or not the generated documentation includes the WSDL port types.
  - **Messages** - Specifies whether or not the generated documentation includes the WSDL messages.
  - **XML Schema Components** - Specifies whether or not the generated documentation includes the XML Schema components.
  - **Only global elements and types** - Specifies whether or not the generated documentation includes only global elements and types.

- **Component Details**
  - **Namespace** - Presents the namespace information for WSDL or XML Schema components.
  - **Location** - Presents the location information for each WSDL or XML Schema component.
  - **Used by** - Presents the list of components that reference the current one.
  - **Documentation** - Presents the component documentation. If you choose **Escape XML Content**, the XML tags are presented in the documentation.
  - **Source** - Presents the XML fragment that defines the current component.
  - **Instance** - Generates a sample XML instance for the current component.

  **Note:**
  This option applies to the XML Schema components only.

- **XML Schema Diagram** - Displays the diagram for each XML Schema component. You can choose the image format (JPEG, PNG, GIF, SVG) to use for the diagram section.
- **Diagram annotations** - Specifies whether or not the annotations of the components presented in the diagram sections are included.

- **Generate index** - Displays an index with the components included in the documentation.
  - **Include local elements and attributes** - If selected, local elements and attributes are included in the documentation index.
  - **Include resource hierarchy** - Specifies whether or not the resource hierarchy for an XML Schema documentation is generated. It is deselected by default.

**Export settings** - Save the current settings in a settings file for further use (for example, if you need the exported settings file for generating the documentation from the command-line interface).

**Import settings** - Reloads the settings from the exported file.

**Generate** - Use this button to generate the WSDL documentation.
Tip:
This function can be executed from an automated command-line script, for more details, see Scripting Oxygen (on page 3196).

Generating WSDL Documentation in HTML Format

The WSDL documentation generated in HTML format is presented in a visual diagram style with various sections, hyperlinks, and options.

Figure 370. WSDL Documentation in HTML Format

The documentation of each component is presented in a separate section. The title of the section is composed of the component type and the component name. The component information (namespace, documentation, etc.) is presented in a tabular form.

If you choose to split the output into multiple files, the table of contents is displayed in the left frame and is divided in two tabs: **Components** and **Resource Hierarchy**.

The **Components** tab allows you to group the contents by namespace, location, or component type. The WSDL components from each group are sorted alphabetically. The **Resource Hierarchy** tab displays the dependencies between WSDL and XML Schema modules in a tree-like fashion. The root of the tree is the WSDL document that you generate documentation for.

After the documentation is generated, you can collapse or expand details for some WSDL components by using the **Showing** options or the **Collapse** or **Expand** buttons.
Generating WSDL Documentation in a Custom Format

To obtain the default HTML documentation output from a WSDL document, Oxygen XML Editor uses an intermediary XML document to which it applies an XSLT stylesheet. To create a custom output from your WSDL document, edit the `wsdlDocHtml.xsl` XSLT stylesheet or create your own.

**Note:**
The `wsdlDocHtml.xsl` stylesheet that is used to obtain the HTML documentation is located in the `{OXYGEN_INSTALL_DIR}/frameworks/wsdl_documentation/xsl` folder.

**Note:**
The intermediary XML document complies with the `wsdlDocSchema.xsd` XML Schema. This schema is located in the `{OXYGEN_INSTALL_DIR}/frameworks/wsdl_documentation` folder.

When using a custom format, you can also copy additional resources into the output folder or choose to keep the intermediate XML files created during the documentation process.

**WSDL SOAP Analyzer Tool**

**WSDL SOAP Analyzer** is a tool that helps you test if the messages defined in a Web Service Descriptor (WSDL) are accepted by a Web Services server.
After you edit and validate your Web service descriptor against a mix of the XML Schemas for WSDL and SOAP, it is easy to check if the defined SOAP messages are accepted by the remote Web Services server by using the integrated WSDL SOAP Analyzer tool (available from the toolbar or Tools menu).

Oxygen XML Editor provides two ways of testing, one for the currently edited WSDL document and another for the remote WSDL documents that are published on a web server. To open the WSDL SOAP Analyzer tool for the currently edited WSDL document do one of the following:

- Click the WSDL SOAP Analyzer toolbar button.
- Use the WSDL SOAP Analyzer action from the Tools menu.
- Go to Open with > WSDL SOAP Analyzer in the contextual menu of the Project (on page 402) view.
This tool contains a SOAP analyzer and sender for Web Services Description Language file types. The analyzer fields are as follows:

- **Services** - The list of services defined by the WSDL file.
- **Ports** - The ports for the selected service.
- **Operations** - The list of available operations for the selected service.
- **Action URL** - The script that serves the operation.
- **SOAP Action** - Identifies the action performed by the script.
- **Version** - Choose between 1.1 and 1.2. The SOAP version is selected automatically depending on the selected port.
• **Request Editor** - It allows you to compose the web service request. When an action is selected, Oxygen XML Editor tries to generate as much content as possible for the SOAP request. The envelope of the SOAP request has the correct namespace for the selected SOAP version, that is `http://schemas.xmlsoap.org/soap/envelope/` for SOAP 1.1 or `http://www.w3.org/2003/05/soap-envelope` for SOAP 1.2. Usually you just have to change a few values for the request to be valid. The [Content Completion Assistant](on page 3217) is available for this editor and is driven by the schema that defines the type of the current message. While selecting various operations, Oxygen XML Editor remembers the modified request for each one. You can click the **Regenerate** button to overwrite your modifications for the current request with the initial generated content.

• **Attachments List** - You can define a list of file URLs to be attached to the request.

• **Response Area** - Initially it displays an auto generated server sample response so you can have an idea about how the response looks like. After pressing the **Send** button, it presents the message received from the server in response to the Web Service request. It may show also error messages. If the response message contains attachments, Oxygen XML Editor prompts you to save them, then tries to open them with the associated system application.

• **Errors List** - There may be situations where the WSDL file is respecting the WSDL XML Schema, but it fails to be valid (for example, in the case of a message that is defined by means of an element that is not found in the types section of the WSDL). In such a case, the errors are listed here. This list is presented only when there are errors.

• **Send Button** - Executes the request. A status dialog box is displayed when Oxygen XML Editor is connecting to the server.

The testing of a WSDL file is straight-forward. Click the WSDL analysis button, then select the service, the port, and the operation. The editor generates the skeleton for the SOAP request. You can edit the request, eventually attach files to it and send it to the server. Watch the server response in the response area. You can find more details in the [Testing Remote WSDL Files](on page 1075) section.

**Note:**

SOAP requests and responses are automatically validated in the **WSDL SOAP Analyzer** using the XML Schemas specified in the WSDL file.

Once defined, a request derived from a Web Service descriptor can be saved with the **Save** button to a Web Service SOAP Call (WSSC) file for later reuse. In this way, you save time in configuring the URLs and parameters.

You can open the result of a Web Service call in an editor panel using the **Open** button.

**Testing Remote WSDL Files**

To open and test a remote WSDL file the steps are the following:

1. Go to **Tools > WSDL SOAP Analyzer**.
2. On the **WSDL File** tab enter the URL of the remote WSDL file.
3. Click the OK button.

This will open the WSDL SOAP Analyzer tool (on page 1072). In the Saved SOAP Request tab you can open directly a previously saved Web Service SOAP Call (WSSC) file, thus skipping the analysis phase.

Editing CSS Stylesheets

Oxygen XML Editor includes a built-in editor for CSS stylesheets. This section presents the features of the CSS editor and how these features should be used. The features of the CSS editor include:

- **Create new CSS files and templates** - You can use the built-in new file wizards to create new CSS documents or templates (on page 368).
- **Open and Edit CSS files** - CSS files can be opened and edited in a source editing mode.
- **Validation** - Presents validation errors in CSS files.
- **Content completion** - Offers proposals for properties and the values that are available for each property.
- **Syntax highlighting** - The syntax highlighting in Oxygen XML Editor makes CSS files more readable.
- **Shortcut to open resources** - You can use Ctrl + Single-Click (Command + Single-Click on macOS) to open imported stylesheets or other resources (such as images) in the default system application for the particular type of resource.

Related Information:

CSS Support in Author Mode (on page 2307)
Supported CSS Selectors (on page 2313)
Supported CSS Properties (on page 2320)
CSS Extensions (on page 2333)

Validating CSS Stylesheets

Oxygen XML Editor includes a built-in CSS Validator, integrated with general validation support. This makes the usual validation features (on page 775) for presenting errors also available for CSS stylesheets.

When you edit a CSS document, you can access the CSS validator options (on page 234) by selecting Validation options from the Document > Validate menu.

The CSS properties accepted by the validator are those included in the current CSS profile that is selected in the CSS validation preferences (on page 234). The CSS 3 with Oxygen extensions profile includes all the CSS 3 standard properties plus the CSS extensions specific for Oxygen (on page 2333) that can be used in Author mode (on page 354). That means all Oxygen-specific extensions are accepted in a CSS stylesheet by the built-in CSS validator (on page 1076) when this profile is selected.

Specify Custom CSS Properties

To specify custom CSS properties, follow these steps:
1. Create a file named `CustomProperties.xml` that has the following structure:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<css_keywords
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.oxygenxml.com/ns/css
    http://www.oxygenxml.com/ns/css/CssProperties.xsd"
    xmlns="http://www.oxygenxml.com/ns/css">
  <property name="custom">
    <summary>Description for custom property.</summary>
    <value name="customValue"/>
    <value name="anotherCustomValue"/>
  </property>
</css_keywords>
```

2. Go to your desktop and create the `builtin/css-validator/` folder structure.

3. Press and hold `Shift` and right-click anywhere on your desktop. From the contextual menu, select **Open Command Window Here**.

4. In the command line, run the `jar cvf custom_props.jar builtin/` command. The `custom_props.jar` file is created.

5. Go to `[OXYGEN_INSTALL_DIR]/lib` and create the `endorsed` folder. Copy the `custom_props.jar` file to `[OXYGEN_INSTALL_DIR]/lib/endorsed`.

### Content Completion in CSS Stylesheets

A *Content Completion Assistant (on page 3217)*, similar to the one available for XML documents *(on page 532)* offers the CSS properties and the values available for each property. It can be manually activated with the `Ctrl + Space (Command + Space on macOS)` shortcut and is context-sensitive when invoked for the value of a property. The *Content Completion Assistant* also includes code templates that can be used to quickly insert code fragments *(on page 536)* into CSS stylesheets. The code templates that are proposed include form controls, actions, and **Author** mode operations.

**Figure 374. Content Completion in CSS Stylesheets**

The properties and values available are dependent on the CSS Profile selected in the *CSS preferences (on page 234)*. The CSS 2.1 set of properties and property values is used for most of the profiles. However, with CSS 1 and CSS 3 specific proposal sets are used.
The profile CSS 3 with Oxygen extensions includes all the CSS 3 standard properties plus the CSS extensions specific for Oxygen XML Editor (on page 2333) that can be used in Author mode (on page 354).

Proposals for CSS Selectors - After inserting a CSS selector, the content completion assistance will propose a list of pseudo-elements and pseudo-classes that are available for the selected CSS profile.

Proposals for @media and @import Rules - After inserting @media or @import <url> rules, the content completion assistance will propose a list of supported media types.

Related Information:
Specify Custom CSS Properties (on page 1076)

Syntax Highlighting in CSS Files

Oxygen XML Editor supports syntax highlighting in Text mode to make it easier to read the semantics of the structured content by displaying each type of code in different colors and fonts.

To customize the colors or styles used for the syntax highlighting colors for CSS files, follow these steps:

1. Open the Preferences dialog box (Options > Preferences) (on page 124).
2. Go to Editor > Syntax Highlight (on page 224).
3. Select and expand the CSS section in the top pane.
4. Select the component you want to change and customize the colors or styles using the selectors to the right of the pane.

You can see the effects of your changes in the Preview pane.

Related Information:
Syntax Highlight Preferences (on page 224)

CSS Outline View

The Outline view for CSS stylesheets presents the import declarations for other CSS stylesheet files and all the selectors defined in the current CSS document. The selector entries can be presented as follows:

- In the order they appear in the document.
- Sorted by the element name used in the selector.
- Sorted by the entire selector string representation.

You can synchronize the selection in the Outline view with the cursor moves or changes you make in the stylesheet document. When you select an entry from the Outline view, Oxygen XML Editor highlights the corresponding import or selector in the CSS editor.

By default, it is displayed on the left side of the editor. If the view is not displayed, it can be opened by selecting it from the Window > Show View menu.
Figure 375. CSS Outline View

The selectors presented in this view can be found quickly using the key search field. When you press a sequence of character keys while the focus is in the view, the first selector that starts with that sequence is selected automatically.

Folding in CSS Stylesheets

In a large CSS stylesheet document, some styles can be collapsed so that only the styles that are needed remain in focus. The same folding features available for XML documents (on page 528) are also available in CSS stylesheets.

Note:

To enhance your editing experience, you can select entire blocks (parts of text delimited by brackets) by double-clicking somewhere inside the brackets.

Formatting and Indenting CSS Stylesheets (Pretty Print)

If the edited CSS stylesheet becomes unreadable because of the bad alignment of the text lines, the format and indent operation available for XML documents (on page 555) is also available for CSS stylesheets. It works in the same way as for XML documents and is available as the same menu and toolbar action.

Minifying CSS Stylesheets

Minification (or compression) of a CSS document is the practice of removing unnecessary code without affecting the functionality of the stylesheet.
To minify a CSS, invoke the contextual menu anywhere in the edited document and choose the **Minify CSS** action. Oxygen XML Editor opens a dialog box that allows you to:

- Set the location of the resulting CSS.
- Place each style rule on a new line.

After pressing **OK**, Oxygen XML Editor performs the following actions:

- All spaces are normalized (all leading and trailing spaces are removed, while sequences of white spaces are replaced with single space characters).
- All comments are removed.

**Note:**
The CSS minifier relies heavily upon the W3C CSS specification. If the content of the CSS file you are trying to minify does not conform with the specifications, an error dialog box will be displayed, listing all errors encountered during the processing.

The resulting CSS stylesheet gains a lot in terms of execution performance, but loses in terms of readability. The source CSS document is left unaffected.

**Note:**
To restore the readability of a minified CSS, invoke the **Format and Indent** action from the **Document > Source** menu, the **Source** submenu from the contextual menu, or **Source** toolbar. However, this action will not recover any of the deleted comments.

### Editing LESS Stylesheets

Oxygen XML Editor provides support for stylesheets coded with the LESS dynamic stylesheet language. LESS extends the CSS language by adding features that allow mechanisms such as *variables, nesting, mixins, operators, and functions*. Oxygen XML Editor offers additional LESS-editing features that include:

- **Create new LESS files and templates** - You can use the built-in new file wizards to create new LESS documents or templates *(on page 368).*
- **Open and Edit LESS files** - LESS files can be opened and edited in a source editing mode.
- **Validation** - Presents validation errors in LESS files.
- **Content completion** - Offers proposals for properties and the values that are available for each property.
- **Compile to CSS** - Options are available to compile LESS files to CSS.
- **Syntax highlighting** - Oxygen XML Editor supports syntax highlighting in LESS files, although there may be some limitations in supporting all the LESS constructs.
- **Shortcut to open resources** - While editing LESS files, you can use **Ctrl + Single-Click** (Command + Single-Click on macOS) to open imported stylesheets or other resources (such as images) in the default system application for the particular type of resource.
For more information about LESS go to: http://lesscss.org/.

Validating LESS Stylesheets

Oxygen XML Editor includes a built-in LESS CSS Validator, integrated with general validation support. The usual validation features (on page 775) for presenting errors also available for LESS stylesheets.

Oxygen XML Editor provides three validation methods:

- Automatic validation as you type - marks validation errors in the document as you are editing.
- Validation upon request, by pressing the Validate button from the Validation toolbar drop-down menu. An error list is presented in the message panel at the bottom of the editor.
- Validation scenarios, by selecting Configure Validation Scenario(s) from the Validation toolbar drop-down menu. Errors are presented in the message panel at the bottom of the editor. This is useful when you need to validate the current file as part of a larger LESS import hierarchy (for instance, you may change the URL of the file to validate to the root of the hierarchy).

Content Completion in LESS Stylesheets

A Content Completion Assistant (on page 3217) offers the LESS properties and the values available for each property. It can be manually activated with the Ctrl + Space (Command + Space on macOS) shortcut and is context-sensitive when invoked for the value of a property in a LESS file. The Content Completion Assistant also includes code templates that can be used to quickly insert code fragments (on page 536) into LESS stylesheets. The code templates that are proposed include form controls, actions, and Author mode operations.

The properties and values available are dependent on the CSS Profile selected in the CSS preferences (on page 234).
Syntax Highlighting in LESS Files

Oxygen XML Editor supports syntax highlighting in Text mode to make it easier to read the semantics of the structured content by displaying each type of code in different colors and fonts.

To customize the colors or styles used for the syntax highlighting colors for LESS files, follow these steps:

1. Open the Preferences dialog box (Options > Preferences) (on page 124).
2. Go to Editor > Syntax Highlight (on page 224).
3. Select and expand the LESS section in the top pane.
4. Select the component you want to change and customize the colors or styles using the selectors to the right of the pane.

You can see the effects of your changes in the Preview pane.

Related Information:
Syntax Highlight Preferences (on page 224)

Compiling LESS Stylesheets to CSS

When editing LESS files, you can compile the files into CSS. Oxygen XML Editor provides both manual and automatic options to compile LESS stylesheets into CSS.

⚠️ Important:
The LESS processor works well only with files having the UTF-8 encoding. Thus, it is highly recommended that you always use the utf-8 encoding when working with LESS files or the files they import (other LESS or CSS files). You can use the following directive at the beginning of your files:

```css
@charset "utf-8";
```

You have two options for compiling LESS files to CSS:

1. Use the contextual menu in a LESS file and select Compile to CSS (Ctrl + Shift + C (Command + Shift + C on macOS)).
2. Select the Automatically compile LESS to CSS when saving option (on page 202) (in the Save preferences page). If selected, when you save a LESS file it will automatically be compiled to CSS (this option is deselected by default).

⚠️ Important:
If this option is selected, when you save a LESS file, the CSS file that has the same name as the LESS file is overwritten without warning. Make sure all your changes are made in the LESS file. Do not edit the CSS file directly, as your changes might be lost.
Editing Relax NG Schemas

An XML Schema describes the structure of an XML document and is used to validate XML document instances against it, to check that the XML instances conform to the specified requirements. If an XML instance conforms to the schema then it is said to be valid. Otherwise, it is invalid.

Oxygen XML Editor offers support for editing Relax NG schema files in the following editing modes:

- **Text editing mode (on page 989)** - Allows you to edit Relax NG schema files in a source editing mode, along with a schema design pane with two tabs that offer a Full Model View (on page 1084) and Logical Model View (on page 1085).
- **Grid editing mode (on page 354)** - Displays Relax NG schema files in a structured spreadsheet-like grid.
- **Author editing mode (on page 588)** - The visual Author mode is also available for Relax NG schema files, presenting the schema similar to the Relax NG compact syntax. It links to imported schemas and external references. Embedded Schematron is also supported in Relax NG schemas with XML syntax.

For information about applying and detecting schemas, see [Associating a Schema to XML Documents (on page 816)](#).

Related Information:

- [Associating a Schema to XML Documents (on page 816)](#)

Modular Contextual Relax NG Schema Editing Using 'Main Files' Support

Smaller interrelated modules that define a complex Relax NG Schema cannot be correctly edited or validated individually, due to their interdependency with other modules. For example, an element defined in a main schema document is not visible when you edit an included module. Oxygen XML Editor provides the support for defining the main module (or modules), thus allowing you to edit any of the imported/included schema files in the context of the larger schema structure.

You can set a main Relax NG document either using the [main files support from the Project view (on page 418)](#), or using a validation scenario.

To set a main file using a validation scenario, add validation units that point to the main schemas. Oxygen XML Editor warns you if the current module is not part of the dependencies graph computed for the main schema. In this case, it considers the current module as the main schema.

The main advantage of editing in the context of a main file (on page 3220) is that it provides correct validation of a module in the context of a larger schema structure.

Related Information:

- [Creating a New Validation Scenario (on page 788)](#)
- [XML Schema Outline View (on page 993)](#)
Relax NG Schema Diagram Editor

This section explains how to use the graphical diagram editor for Relax NG schemas.

Introduction to Relax NG Schema Diagram Editor

Oxygen XML Editor provides a simple, expressive, and easy-to-read schema diagram editor for Relax NG schemas.

With this new feature you can easily develop complex schemas, print them on multiple pages or save them as JPEG, PNG, or BMP images. It helps both schema authors in developing the schema and content authors who are using the schema to understand it.

Oxygen XML Editor provides a side-by-side source and diagram presentation with real-time synchronization:

• The changes you make in the editor are immediately visible in the Diagram (no background parsing).
• Changing the selected element in the diagram selects the underlying code in the source editor.

Full Model View

When you create a new schema document or open an existing one, the editor panel is divided in two sections: one containing the schema diagram and the second the source code. The schema diagram editor has two tabs that offer a Full Model View and Logical Model View (on page 1085).
The following references can be expanded in place: patterns, includes, and external references. This expansion mechanism, coupled with the synchronization support, makes the schema navigation easy.

All the element and attribute names are editable by double-clicking the names.

**Logical Model View**

The **Logical Model View** presents the compiled schema in the form of a single pattern. The patterns that form the element content are defined as top-level patterns with generated names. These names are generated depending of the elements name class.
Symbols Used in the Schema Diagram

The views in the schema diagram editor renders all the Relax NG schema patterns with the following intuitive symbols:

- **name** - define pattern with the `@name` attribute set to the value shown inside the rectangle (in this example `name`).
- **attlist.person** - define pattern with the `@combine` attribute set to `interleave` and the `@name` attribute set to the value shown inside the rectangle (in this example `attlist.person`).
- **attlist.person** - define pattern with the `@combine` attribute set to `choice` and the `@name` attribute set to the value shown inside the rectangle (in this example `attlist.person`).
- **name** - element pattern with the `@name` attribute set to the value shown inside the rectangle (in this example `name`).
- **note** - attribute pattern with the `@name` attribute set to the value shown inside the rectangle (in this case `note`).
- **family** - ref pattern with the `@name` attribute set to the value shown inside the rectangle (in this case `family`).
- `1..` - oneOrMore pattern.
- `0..` - zeroOrMore pattern.
- `0..1` - optional pattern.
- `|` - choice pattern.
- `true` - value pattern (for example, used inside a choice pattern).
- `*` - group pattern.
- `Doc` - A pattern from the Relax NG Annotations namespace (http://relaxng.org/ns/compatibility/annotations/1.0) that is treated as a documentation element in a Relax NG schema.
- `text` - text pattern.
- `∅` - empty pattern.

### Actions Available in the Schema Diagram Editor

When editing Relax NG schemas in **Full Model View (on page 1084)**, the contextual menu offers the following actions:

- **Go to definition (Available for imported components)**
  
  This action is available for imported components from other RNG files, and it shows where that component is defined.

- **Append child**
  
  Appends a child to the selected component.

- **Insert Before**
  
  Inserts a component before the selected component.

- **Insert After**
  
  Inserts a component after the selected component.

- **Edit attributes**
  
  Edits the attributes of the selected component.

- **Remove**
  
  Removes the selected component.

- **Show only the selected component**
Depending on its state (selected/not selected), either the selected component or all the diagram components are shown.

**Show Annotations**

Depending on its state (selected/not selected), the documentation nodes are shown or hidden.

**Auto expand to references**

This option controls how the schema diagram is automatically expanded. If you select it and then edit a top-level element or you make a refresh, the diagram is expanded until it reaches referenced components. If this option is left unchecked, only the first level of the diagram is expanded, showing the top-level elements. For large schemas, the editor disables this option automatically.

**Collapse Children**

Collapses the children of the selected view.

**Expand Children**

Expands the children of the selected view.

**Print Selection**

Prints the selected view.

**Save as Image**

Saves the current selection as JPEG, BMP, SVG or PNG image.

**Refresh**

Refreshes the schema diagram according to the changes in your code. They represent changes in your imported documents or changes that are not reflected automatically in the compiled schema.

If the schema is not valid, you see only an error message in the Logical Model View (on page 1085) instead of the diagram.

**Validating Relax NG Schema Documents**

By default, Relax NG schema files are validated as you type. To change this, open the Preferences dialog box (Options > Preferences) (on page 124), go to Editor > Document Checking, and deselect the Enable automatic validation option (on page 226).

To validate a Relax NG schema document manually, select the Validate action from the Validation toolbar drop-down menu or the Document > Validate menu. When Oxygen XML Editor validates a Relax NG schema file, it expands all the included modules so the entire schema hierarchy is validated. The validation problems are highlighted directly in the editor, making it easy to locate and fix any issues.
The intelligent *Content Completion Assistant* allows you to quickly identify and insert elements, attributes, and attribute values that are valid in the current editing context. All available proposals are listed in a pop-up menu displayed at the current cursor position.

The *Content Completion Assistant* is enabled by default. To disable it, open the Preferences dialog box (Options > Preferences), go to Editor > Content Completion, and deselect the Enable content completion option.

When active, the *Content Completion Assistant* displays a list of context-sensitive proposals valid at the current cursor position. It can be manually activated with the Ctrl + Space (Command + Space on macOS) shortcut. You can navigate through the list of proposals by using the Up and Down keys on your keyboard. For each selected item in the list, the *Content Completion Assistant* displays a documentation window. You can customize the size of the documentation window by dragging its top, right, and bottom borders.

To insert the selected proposal in Text mode, do one of the following:

- Press Enter or Tab to insert both the start and end tags and position the cursor inside the start tag in a position suitable for inserting attributes.
- Press Ctrl + Enter (Command + Enter on macOS) to insert both the start and end tags and positions the cursor between the tags in a position where you can start typing content.

If you are using the concept of main files to import/include modules, the *Content Completion Assistant* collects its components starting from the main files. The main files can be defined in the project or in the associated validation scenario. For more information about the Main Files support in Oxygen XML Editor, see Defining Main Files at Project Level.

The *Content Completion Assistant* also offers additional information for the element and attribute proposals in the form of schema annotations that is displayed in a tooltip.
Syntax Highlighting in Relax NG Schemas

Oxygen XML Editor supports syntax highlighting in Text mode to make it easier to read the semantics of the structured content by displaying each type of code in different colors and fonts.

To customize the colors or styles used for the syntax highlighting colors for Relax NG schemas, follow these steps:

1. Open the Preferences dialog box (Options > Preferences) (on page 124).
2. Go to Editor > Syntax Highlight (on page 224).
3. Select and expand the XML section in the top pane (for RELAX NG Compact Syntax schemas, select and expand the RNC section).
4. Select the component you want to change and customize the colors or styles using the selectors to the right of the pane.
5. Select the XML tab in the Preview pane to see the effects of your changes (for RELAX NG Compact Syntax schemas, the tab is RNC).

Tip:
Oxygen XML Editor also allows you to specify syntax highlighting colors for specific XML elements and attributes with specific namespace prefixes. This can be done in the Editor > Syntax Highlight > Elements/Attributes by Prefix preferences page (on page 225).

Related Information:
Syntax Highlight Preferences (on page 224)

Quick Fixes for DTD, XSD, and Relax NG Errors

Oxygen XML Editor offers Quick Fixes (on page 3223) for common errors that appear in XML documents that are validated against DTD, XSD, or Relax NG schemas.
Note:
For XML documents validated against XSD schemas, the Quick Fixes are only available if you use the default Xerces validation engine.

Quick Fixes are available in Text mode and Author mode.

Oxygen XML Editor provides Quick Fixes for numerous types of problems, including the following:

<table>
<thead>
<tr>
<th>Problem Type</th>
<th>Available Quick Fixes</th>
</tr>
</thead>
<tbody>
<tr>
<td>A specific element is required in the current context</td>
<td>Insert the required element</td>
</tr>
<tr>
<td>An element is invalid in the current context</td>
<td>Remove the invalid element</td>
</tr>
<tr>
<td>The content of the element should be empty</td>
<td>Remove the element content</td>
</tr>
<tr>
<td>An element is not allowed to have child elements</td>
<td>Remove all child elements</td>
</tr>
<tr>
<td>Text is not allowed in the current element</td>
<td>Remove the text content</td>
</tr>
<tr>
<td>A required attribute is missing</td>
<td>Insert the required attribute</td>
</tr>
<tr>
<td>An attribute is not allowed to be set for the current element</td>
<td>Remove the attribute</td>
</tr>
<tr>
<td>The attribute value is invalid</td>
<td>Propose the correct attribute values</td>
</tr>
<tr>
<td>ID value is already defined</td>
<td>Generate a unique ID value</td>
</tr>
<tr>
<td>References to an invalid ID</td>
<td>Change the reference to an already defined ID</td>
</tr>
</tbody>
</table>

Related Information:
Schematron Quick Fixes (SQF) (on page 815)
Ignoring/Unignoring Validation Problems (on page 812)

Relax NG Outline View
The Outline view for Relax NG schemas presents a list with the patterns that appear in the diagram in both the Full Model View (on page 1084) and Logical Model View (on page 1085) cases and it allows for quick access to a component by name. By default, it is displayed on the left side of the editor. If the view is not displayed, it can be opened by selecting it from the Window > Show View menu.
This view has two modes, with the tree showing either the XML structure or the defined pattern (components) collected from the current document. By default, the Outline view presents the components.

When the Show components option is selected in the Settings menu on the Outline view toolbar, the following option is available:

Show XML structure

Shows the XML structure of the current document in a tree-like manner.

The following actions are available in the Settings menu on the Outline view toolbar when the Show XML structure option is selected:

Filter returns exact matches

The text filter of the Outline view returns only exact matches.

Selection update on cursor move

Allows a synchronization between Outline view and schema diagram. The selected view from the diagram will be also selected in the Outline view.

Show components
Shows the defined pattern collected from the current document.

Flat presentation mode of the filtered results
When active, the application flattens the filtered result elements to a single level.

Show comments and processing instructions
Show/hide comments and processing instructions in the Outline view.

Show element name
Show/hide element name.

Show text
Show/hide additional text content for the displayed elements.

Show attributes
Show/hide attribute values for the displayed elements. The displayed attribute values can be changed from the Outline preferences panel (on page 306).

Configure displayed attributes
Displays the XML Structured Outline preferences page (on page 306).

The following contextual menu actions are also available in the Outline view when the Show XML structure option is selected in the Settings menu:

Append Child
Displays a list of elements that you can insert as children of the current element.

Insert Before
Displays a list of elements that you can insert as siblings of the current element, before the current element.

Insert After
Displays a list of elements that you can insert as siblings of the current element, after the current element.

Edit Attributes
Opens a dialog box that allows you to edit the attributes of the currently selected component.

Toggle Comment
Comments/uncomments the currently selected element.

Search references
Searches for the references of the currently selected component.

Search references in
Searches for the references of the currently selected component in the context of a scope that you define.
Component dependencies

Opens the Component Dependencies view (on page 1097) that displays the dependencies of the currently selected component.

Rename Component in

Renames the currently selected component in the context of a scope that you define.

Cut

Cuts the currently selected component.

Copy

Copies the currently selected component.

Delete

Deletes the currently selected component.

Expand More

Expands the structure of a component in the Outline view.

Collapse All

Collapses the structure of all the component in the Outline view.

The upper part of the Outline view contains a filter box that allows you to focus on the relevant components. Type a text fragment in the filter box and only the components that match it are presented. For advanced usage you can use wildcard characters (such as * or ?) and separate multiple patterns with commas.

RNG Resource Hierarchy/Dependencies View

The Resource Hierarchy/Dependencies view displays the hierarchy or dependencies for resources included in an RNG schema. If the view is not displayed, it can be opened by selecting it from the Window > Show View menu.

If you want to see the hierarchy or dependencies of an RNG schema, select the desired schema in the Project view (on page 402) and choose Resource Hierarchy or Resource Dependencies from the contextual menu.
The following actions are available on the toolbar of the **Resource Hierarchy/Dependencies** view:

- **Refresh**
  
  Refreshes the Hierarchy/Dependencies structure.

- **Stop**

  Stops the hierarchy/dependencies computing.

- **Show Hierarchy**

  Allows you to choose a resource to compute the hierarchy structure.

- **Show Dependencies**

  Allows you to choose a resource to compute the dependencies structure.

- **Configure dependencies search scope**

  Allows you to configure a scope to compute the dependencies structure. There is also an option for automatically using the defined scope for future operations.

- **History**

  Provides access to the list of previously computed dependencies. Use the **Clear history** button to remove all items from this list.

The contextual menu for a resource listed in the **Resource Hierarchy/Dependencies** view contains the following actions:

- **Open**
Opens the resource. You can also double-click a resource in the Hierarchy/Dependencies structure to open it.

**Go to reference**

Opens the source document where the resource is referenced.

**Copy location**

Copies the location of the resource.

**Move resource**

Moves the selected resource.

**Rename resource**

Renames the selected resource.

**Resource Hierarchy**

Shows the hierarchy for the selected resource.

**Resource Dependencies**

Shows the dependencies for the selected resource.

**Add to Main Files**

Adds the currently selected resource in the Main Files directory (on page 418).

**Expand More**

Expands more of the children of the selected resource from the Hierarchy/Dependencies structure.

**Collapse All**

Collapses all children of the selected resource from the Hierarchy/Dependencies structure.

**Tip:**

When a recursive reference is encountered in the Hierarchy view, the reference is marked with a special icon.

**Note:**

The Move resource or Rename resource actions give you the option to update the references to the resource (on page 1062).

**Related Information:**

Modular Contextual XML Editing Using 'Main Files' Support (on page 829)

Search and Refactor Operations Scope (on page 831)
Moving/Renaming RNG Resources

You can move and rename a resource presented in the Resource/Hierarchy Dependencies view, using the Rename resource and Move resource refactoring actions from the contextual menu.

When you select the Rename action in the contextual menu of the Resource/Hierarchy Dependencies view, the Rename resource dialog box is displayed. The following fields are available:

- **New name** - Presents the current name of the edited resource and allows you to modify it.
- **Update references of the renamed resource(s)** - Select this option to update the references to the resource you are renaming. A Preview option is available that allows you to see what will be updated before selecting Rename to process the operation.

When you select the Move action from the contextual menu of the Resource/Hierarchy Dependencies view, the Move resource dialog box is displayed. The following fields are available:

- **Destination** - Presents the path to the current location of the resource you want to move and gives you the option to introduce a new location.
- **New name** - Presents the current name of the moved resource and gives you the option to change it.
- **Update references of the moved resource(s)** - Select this option to update the references to the resource you are moving, in accordance with the new location and name. A Preview option is available that allows you to see what will be updated before selecting Move to process the operation.

**Note:**
Updating the references of a resource that is resolved through a catalog is not supported. Also, the update references operation is not supported if the path to the renamed or moved resource contains entities.

Relax NG Schema Component Dependencies View

The Component Dependencies view allows you to see the dependencies for a selected component. This is helpful if you want to see where components are used in the entire hierarchy. For example, if you want to find all the references where a given component is used.

If the view is not displayed, it can be opened by selecting it from the Window > Show View menu.

To see the dependencies of a Relax NG component:

1. Right-click the desired component in the editor or Outline view.
2. Select the Component Dependencies action from the contextual menu.

The action is available for all named defines.

If a component contains multiple references, a small table is displayed at the bottom of the view that contains all the references. When a recursive reference is encountered, it is marked with a special icon.
The **Component Dependencies** view includes the following toolbar actions:

- **Refresh**
  - Refreshes the dependencies structure.

- **Stop**
  - Stops the dependency computation.

- **Configure**
  - Allows you to choose the search scope for computing the dependencies structure. This is helpful for making sure all imported/included resources are computed.

- **History**
  - Allows you to select from a list of the most recently used dependency computations.

In addition, the following actions are available in the contextual menu:

- **Go to First Reference**
  - Selects the first reference of the currently selected component in the dependencies tree.

- **Go to Component**
  - Shows the definition of the currently selected component in the dependencies tree.

**Related Information:**

Search and Refactor Operations Scope (on page 831)

### Searching and Refactoring Actions in RNG Schemas

#### Search Actions

The following search actions can be applied on named *defines* and are available from the **Search** submenu in the contextual menu of the current editor or from the **Document > References** menu:
Search References

Searches all references of the item found at current cursor position in the defined scope, if any. If a scope is defined, but the currently edited resource is not part of the range of resources determined by this, a warning dialog box is displayed and you have the possibility to define another search scope.

Search References in

Search all references of the item found at current cursor position in the file or files that you specify when you define a scope for the search operation.

Search Declarations

Searches all declarations of the item found at current cursor position in the defined scope if any. If a scope is defined, but the currently edited resource is not part of the range of resources determined by this, a warning dialog box will be displayed and you have the possibility to define another search scope.

Search Declarations in

Searches all declarations of the item found at current cursor position in the file or files that you specify when you define a scope for the search operation.

Search Occurrences in File

Searches all occurrences of the item at the cursor position in the currently edited file.

The following action is available from the contextual menu and the Document > Schema menu:

Go to Definition

Moves the cursor to the definition of the current element in the Relax NG (full syntax) schema.

Note:

You can also use the Ctrl + Single-Click (Command + Single-Click on macOS) shortcut on a reference to display its definition.

Refactoring Actions

The following refactoring actions can be applied on named defines and are available from the Refactoring submenu in the contextual menu of the current editor or from the Document > Refactoring menu:

Rename Component

Allows you to rename the current component (in-place). The component and all its references in the document are highlighted with a thin border and the changes you make to the component at the cursor position are updated in real time to all occurrences of the component. To exit the in-place editing, press the Esc or Enter key on your keyboard.

Rename Component in
Opens a dialog box that allows you to rename the selected component by specifying the new component name and the files to be affected by the modification. If you click the **Preview** button, you can view the files to be affected by the action.

**Figure 383. Rename Identity Constraint Dialog Box**

![Rename Identity Constraint Dialog Box](image)

**RNG Quick Assist Support**

The *Quick Assist support* (on page 3223) improves the development work flow, offering fast access to the most commonly used actions when you edit schema documents.

The *Quick Assist feature* (on page 3223) is activated automatically when the cursor is positioned over the name of a component. It is accessible via a yellow bulb icon (☉) placed at the current line in the stripe on the left side of the editor. Also, you can invoke the *quick assist* menu by using the **Alt + 1** (**Meta + Alt + 1** on macOS) keyboard shortcuts.
The *Quick Assist* support offers direct access to the following actions:

- **Rename Component in**
  Renames the component and all its dependencies.

- **Search Declarations**
  Searches the declaration of the component in a predefined scope. It is available only when the context represents a component name reference.

- **Search References**
  Searches all references of the component in a predefined scope.

- **Component Dependencies**
  Searches the component dependencies in a predefined scope.

- **Change Scope**
  Configures the scope that will be used for future search or refactor operations.

- **Rename Component**
  Allows you to rename the current component in-place.

- **Search Occurrences**
  Searches all occurrences of the component within the current file.

**Related Information:**

- Component Dependencies View *(on page 1097)*
- Resource Hierarchy/Dependencies View *(on page 1094)*
- Searching and Refactoring Actions *(on page 1098)*
- Search and Refactor Operations Scope *(on page 831)*
Configuring a Custom Datatype Library for a RELAX NG Schema

A RELAX NG schema can declare a custom datatype library for the values of elements found in XML document instances. The datatype library must be developed in Java and it must implement the interface specified on the www.thaiopensource.com website.

The JAR (on page 3220) file containing the custom library and any other dependent JAR file must be added to the classpath of the application, that is the JAR files must be added to the folder \OXYGEN_INSTALL_DIR\lib.

To load the custom library, restart Oxygen XML Editor.

Editing NVDL Schemas

Some complex XML documents are composed by combining elements and attributes from namespaces. Furthermore, the schemas that define these namespaces are not even developed in the same schema language. In such cases, it is difficult to specify in the document all the schemas that must be taken into account for validation of the XML document or for content completion. An NVDL (Namespace Validation Definition Language) schema can be used. This schema allows the application to combine and interleave multiple schemas of different types (W3C XML Schema, RELAX NG schema, Schematron schema) in the same XML document.

Oxygen XML Editor offers support for editing NVDL schema files in the following editing modes:

- **Text editing mode (on page 989)** - Allows you to edit NVDL schema files in a source editing mode, along with a schema design pane with two tabs that offer a Full Model View (on page 1103) and Logical Model View (on page 1104).
- **Grid editing mode (on page 354)** - Displays NVDL schema files in a structured spreadsheet-like grid.
- **Author editing mode** - The visual Author mode is also available for Relax NG schema files, presenting them in a compact and easy to understand representation.

For information about applying and detecting schemas, see Associating a Schema to XML Documents (on page 816).

Related Information:

- Associating a Schema to XML Documents (on page 816)

### NVDL Schema Diagram

This section explains how to use the graphical diagram of a NVDL schema.

### Introduction to NVDL Schema Diagram Editor

Oxygen XML Editor provides a simple, expressive, and easy-to-read schema diagram editor for NVDL schemas.
With this new feature you can easily develop complex schemas, print them on multiple pages or save them as JPEG, PNG, and BMP images. It helps both schema authors in developing the schema and content authors that are using the schema to understand it.

Oxygen XML Editor provides a side-by-side source and diagram presentation with real-time synchronization:

- The changes you make in the editor are immediately visible in the Diagram (no background parsing).
- Changing the selected element in the diagram selects the underlying code in the source editor.

**Full Model View**

When you create a schema document or open an existing one, the editor panel is divided in two sections: one containing the schema diagram and the second the source code. The diagram view has two tabbed panes offering a **Full Model View** and a **Logical Model View** (on page 1104).

![Figure 385. NVDL Schema Editor - Full Model View](image)

The **Full Model View** renders all the NVDL elements with intuitive icons. This representation coupled with the synchronization support makes the schema navigation easy.

Double-click any diagram component to edit its properties.
Logical Model View

The Logical Model View presents the compiled schema in the form of a single pattern. The patterns that form the element content are defined as top-level patterns with generated names. These names are generated depending on the elements name class.

Figure 386. Logical Model View for an NVDL Schema

Actions Available in the Diagram Editor

The contextual menu offers the following actions:

**Show only the selected component**

Depending on its state (selected/not selected), either the selected component or all the diagram components are shown.

**Show Annotations**

Depending on its state (selected/not selected), the documentation nodes are shown or hidden.

**Auto expand to references**

This option controls how the schema diagram is automatically expanded. For instance, if you select it and then edit a top-level element or you trigger a diagram refresh, the diagram will be expanded until it reaches the referenced components. If this option is left unchecked, only the first level of the diagram is expanded, showing the top-level elements. For large schemas, the editor disables this option automatically.

**Collapse Children**

Collapses the children of the selected view.

**Expand Children**

Expands the children of the selected view.
Print Selection

Prints the selected view.

Save as Image

Saves the current selection as image, in JPEG, BMP, SVG or PNG format.

Refresh

Refreshes the schema diagram according to the changes in your code (changes in your imported documents or those that are not reflected automatically in the compiled schema).

If the schema is not valid, you see only an error message in the Logical Model View (on page 1104) instead of the diagram.

Validating NVDL Schema Documents

By default, NVDL schema files are validated as you type. To change this, open the Preferences dialog box (Options > Preferences) (on page 124), go to Editor > Document Checking, and deselect the Enable automatic validation option (on page 226).

To validate an NVDL schema document manually, select the ✓ Validate action from the ✓ Validation toolbar drop-down menu or the Document > Validate menu. When Oxygen XML Editor validates an NVDL schema file, it expands all the included modules so the entire schema hierarchy is validated. The validation problems are highlighted directly in the editor, making it easy to locate and fix any issues.

Related Information:

Validating XML Documents Against a Schema (on page 775)
Presenting Validation Errors in Text Mode (on page 777)

Content Completion in NVDL Schemas

The intelligent Content Completion Assistant (on page 3217) allows you to quickly identify and insert elements, attributes, and attribute values that are valid in the current editing context. All available proposals are listed in a pop-up menu displayed at the current cursor position.

The Content Completion Assistant is enabled by default. To disable it, open the Preferences dialog box (Options > Preferences) (on page 124), go to Editor > Content Completion, and deselect the Enable content completion option (on page 211).

When active, the Content Completion Assistant displays a list of context-sensitive proposals valid at the current cursor position. It can be manually activated with the Ctrl + Space (Command + Space on macOS) shortcut. You can navigate through the list of proposals by using the Up and Down keys on your keyboard. For each selected item in the list, the Content Completion Assistant displays a documentation window. You can customize the size of the documentation window by dragging its top, right, and bottom borders.

To insert the selected proposal in Text mode, do one of the following:
• Press Enter or Tab to insert both the start and end tags and position the cursor inside the start tag in a position suitable for inserting attributes.
• Press Ctrl + Enter (Command + Enter on macOS) to insert both the start and end tags and positions the cursor between the tags in a position where you can start typing content.

If you are using the concept of main files (on page 3220) to import/include modules, the Content Completion Assistant collects its components starting from the main files. The main files can be defined in the project or in the associated validation scenario. For more information about the Main Files support in Oxygen XML Editor, see Defining Main Files at Project Level (on page 418).

The Content Completion Assistant also offers additional information for the element and attribute proposals in the form of schema annotations that is displayed in a tooltip.

Figure 387. NVDL Content Completion Assistant

Syntax Highlighting in NVDL Schemas

Oxygen XML Editor supports syntax highlighting in Text mode to make it easier to read the semantics of the structured content by displaying each type of code in different colors and fonts.

To customize the colors or styles used for the syntax highlighting colors for NVDL schemas, follow these steps:

1. Open the Preferences dialog box (Options > Preferences) (on page 124).
2. Go to Editor > Syntax Highlight (on page 224).
3. Select and expand the XML section in the top pane.
4. Select the component you want to change and customize the colors or styles using the selectors to the right of the pane.
5. Select the XML tab in the Preview pane to see the effects of your changes.

Tip:
Oxygen XML Editor also allows you to specify syntax highlighting colors for specific XML elements and attributes with specific namespace prefixes. This can be done in the Editor > Syntax Highlight > Elements/Attributes by Prefix preferences page (on page 225).
NVDL Outline View

The Outline view for NVDL schemas presents a list with the named or anonymous rules that appear in the diagram and it allows for quick access to a rule by name. By default, it is displayed on the left side of the editor. If the view is not displayed, it can be opened by selecting it from the Window > Show View menu.

NVDL Schema Component Dependencies View

The Component Dependencies view allows you to see the dependencies for a selected component. This is helpful if you want to see where components are used in the entire hierarchy. For example, if you want to find all the references where a given component is used.

If the view is not displayed, it can be opened by selecting it from the Window > Show View menu.

To see the dependencies of an NVDL component:

1. Right-click the desired component in the editor or Outline view.
2. Select the Component Dependencies action from the contextual menu.

The action is available for all named modes.

If a component contains multiple references, a small table is displayed at the bottom of the view that contains all the references. When a recursive reference is encountered, it is marked with a special icon.

![Component Dependencies View](image)

The Component Dependencies view includes the following toolbar actions:

- **Refresh**
  - Refreshes the dependencies structure.
- **Stop**
  - Stops the dependency computation.
- **Configure**
Allows you to choose the search scope for computing the dependencies structure. This is helpful for making sure all imported/included resources are computed.

History

Allows you to select from a list of the most recently used dependency computations.

In addition, the following actions are available in the contextual menu:

Go to First Reference

Selects the first reference of the currently selected component in the dependencies tree.

Go to Component

Shows the definition of the currently selected component in the dependencies tree.

Searching and Refactoring Actions in NVDL Schemas

Search Actions

The following search actions can be applied on @name, @useMode, and @startMode attributes and are available from the Search submenu in the contextual menu of the current editor or from the Document > References menu:

Search References

Searches all references of the item found at current cursor position in the defined scope, if any. If a scope is defined, but the currently edited resource is not part of the range of resources determined by this, a warning dialog box is displayed and you have the possibility to define another search scope.

Search References in

Searches all references of the item found at current cursor position in the file or files that you specify when define a scope in the Search References dialog box.

Search Declarations

Searches all declarations of the item found at current cursor position in the defined scope if any. If a scope is defined, but the currently edited resource is not part of the range of resources determined by this, a warning dialog box will be displayed and you have the possibility to define another search scope.

Search Declarations in

Searches all declarations of the item found at current cursor position in the file or files that you specify when you define a scope for the search operation.

Search Occurrences in File

Searches all occurrences of the item at the cursor position in the currently edited file.

The following action is available from the contextual menu and the Document > Schema menu:

Go to Definition
Moves the cursor to its definition in the schema used by the NVDL to validate it.

Note:
You can also use the Ctrl + Single-Click (Command + Single-Click on macOS) shortcut on a reference to display its definition.

Refactoring Actions

The following refactoring actions can be applied on @name, @useMode, and @startMode attributes and are available from the Refactoring submenu in the contextual menu of the current editor or from the Document > Refactoring menu:

Rename Component

Allows you to rename the current component (in-place). The component and all its references in the document are highlighted with a thin border and the changes you make to the component at the cursor position are updated in real time to all occurrences of the component. To exit the in-place editing, press the Esc or Enter key on your keyboard.

Rename Component in

Opens a dialog box that allows you to rename the selected component by specifying the new component name and the files to be affected by the modification. If you click the Preview button, you can view the files to be affected by the action.
Editing JSON Documents

This section explains the features of the Oxygen XML Editor JSON Editor and how to use them.

Resources

For more information about the JSON support in Oxygen XML Editor, see the following resources:

- Webinar: JSON and JSON Schema Support in Oxygen
- Video: JSON Editing
- Video: JSON Tools in Oxygen

JSON Editor

Oxygen XML Editor includes a specialized JSON editor with various editing features for files that have the `json` file extension. It also includes a document template to help you get started with JSON documents. The template is called `JSON` and it can be found in the New Document folder in the New document wizard (on page 368).
Tip:
You can experiment with a sample of a JSON file available at: [OXYGEN-INSTALL-DIR]/samples/json/personal.json.

Text Mode Editor

When editing JSON documents in the Text editing mode, the usual text editing actions (on page 521) are available, along with other editor-specific actions, including:

- Search and Find/Replace (on page 422)
- Drag and Drop (on page 529)
- Validation (on page 1113)
- Format and Indent (Pretty Print) (on page 555)

Note:
You can run XPath expressions on open JSON documents, but in Text mode the XPath results cannot be mapped in the document. However, they can be mapped in the Grid editing mode. You can use the Grid button at the bottom of the editor panel to switch to that editing mode.

Grid Mode Editor

Oxygen XML Editor allows you to view and edit the JSON documents in the Grid mode (on page 354). The JSON is represented in Grid mode as a compound layout of nested tables and the JSON data and structure can be easily manipulated with table-specific operations or drag and drop operations on the grid components.

Figure 390. JSON Editor Grid Mode

You can also use the following JSON-specific contextual actions:

Array
Useful when you want to convert a JSON value to array.

**Insert value before**

Inserts a value before the currently selected one.

**Insert value after**

Inserts a value after the currently selected one.

**Append value as child**

Appends a value as a child of the currently selected value.

You can customize the JSON grid appearance (on page 174) according to your needs. For instance, you can change the font, the cell background, foreground, or even the colors from the table header gradients. The default width of the columns can also be changed.

**Author Visual Editor**

You can edit JSON files in the visual Author editing mode and you have access to the various features and actions that are available when editing XML documents in Author mode (on page 588). When a JSON document is opened in Author mode, it is automatically converted to proper XML structure using the built-in JSON to XML Converter (on page 1131). Additionally, for Boolean, Number, and Null types, an oxy_Type="[symple_type]" attribute structure is added in the XML to preserve the type of the value from the JSON document.

![Figure 391. JSON Editor Author Mode](image-url)
You can also create your own custom JSON framework, similar to the process for creating custom XML frameworks (on page 2133). For example, to create a document type association (framework) for JSON documents, you could:

- Add a rule to match the "JSON" as the root local name.
- Add a rule to match the topProperties attribute that contains a value that is the name of the properties from the first level of the JSON document.
- Add a rule to match the schema attribute that contains a value that is the associated schema from the $schema property.

Note:
The default JSON framework has the Lowest priority in the Document Type Association preferences page (on page 138). If you create a custom JSON framework, you need to set it to a higher priority. Otherwise, the Author mode rendering will revert to the default JSON framework.

Tip:
You can experiment with some samples of custom JSON frameworks available in the [OXYGEN-INSTALL-DIR]/samples/json/author/ directory. There is a sample application form called residentCardForm.json and a sample travel guide called travel-guide.json, along with referenced resources in the images folder.

For more information about the visual editing support for JSON, watch our video demonstration:

https://www.youtube.com/embed/_C2dVpbGANQ

Navigating References in JSON Documents

When editing JSON documents (or JSON Schema), you can easily navigate JSON Pointer references and hyperlinks by using the CTRL + Click shortcut. Holding the CTRL key while hovering over a JSON Pointer references or hyperlink will change the reference to a clickable link.

Validating JSON Documents

Oxygen XML Editor includes a built-in JSON validator that is used to validate JSON documents against JSON Schemas, as well as a built-in JSON Well-Formedness validator (based on the free JAVA source code available at www.json.org). A built-in JSON Schematron Validator engine is also provided to validate JSON documents against a specified Schematron schema. Validation for JSON documents works in both Text mode and Author mode.

For more information, see the following video demonstration:

https://www.youtube.com/embed/3JEL6nFUozQ
Checking Well-Formedness in JSON Documents

A *Well-formed* JSON document is a sequence of Unicode code points that strictly conforms to the JSON grammar defined by the *JSON Data Interchange Syntax specification*. By default, Oxygen XML Editor automatically checks the document for *Well-formedness* as you type.

**Check for Well-Formedness Manually**

To manually check documents for *Well-Formedness*:

- Select the **Check Well-Formedness** (Ctrl + Shift + W (Command + Shift + W on macOS)) action from the **Validation** drop-down menu on the toolbar or from the Document > Validate menu.
- A selection of files can be checked for well-formedness by selecting the **Check Well-Formedness** action from the Validate submenu when invoking the contextual menu in the Project view (on page 402).

**Result:** If any errors are found, the result is displayed in the message panel at the bottom of the editor. Each error is displayed as one record in the result list and is accompanied by an error message. Clicking the record will open the document containing the error and highlight its approximate location.

**Example: A non* Well-formed JSON Document**

```json
{  "person": {  "name": "John Doe"  }
```

This would result in the following error:

```
Expected a ',,' or '}'
```

To resolve the error, click the record in the result list and it will locate and highlight the approximate position of the error. In this case, you would need to identify where the missing end bracket needs to be placed.

**Validating JSON Documents Against JSON Schema or Schematron**

A *valid* JSON document is a *well-formed* document that also conforms to the rules of a JSON Schema that defines the legal syntax of a JSON document. The purpose of the JSON schema is to define the legal properties and values of a JSON document.

This section contains topics that explain the automatic and manual validation possibilities in Oxygen XML Editor, how validation errors are presented, and information about built-in validation scenarios.

Oxygen XML Editor also includes a built-in JSON Schematron Validator engine to validate JSON documents against a Schematron schema specified in a custom validation scenario or using the **Validate with** action (on page 1115).
Tip:
Inside the samples folder, there are a few files you can use to see how Schematron validation can be done with JSON files. The path of the folder containing these sample files is: 

```
[OXYGEN_INSTALL_DIR]/samples/json/schematron/.
```

For information about how to associate a schema for the purposes of validation, see Associating a Schema Through a Validation Scenario (on page 1122).

### Automatic Validation

By default, Oxygen XML Editor is configured to automatically mark validation errors in the JSON document as you are editing. The Enable automatic validation option (on page 226) in the Document Checking preferences page (on page 226) controls whether or not all validation errors and warnings will automatically be highlighted in the editor pane.

The automatic validation starts parsing the document and marking the errors after a configurable delay (on page 226) from the last typed key. Errors are highlighted with underline markers in the main editor pane and small rectangles on the right side ruler. Hovering over a validation error presents a tooltip message with more details about the error.

If the error message is too long to be displayed completely in the error line at the bottom of the editing area, double-clicking the error icon at the left of the error line, or on the error line itself, displays an information dialog box with the full error message. You can use the arrow buttons in this dialog box to navigate through the errors issued by the automatic validation feature.

**Related Information:**
- Manual Validation Actions (on page 1115)
- Presenting Validation Errors in JSON Documents (on page 1116)

### Manual Validation Actions

You can choose to validate JSON documents at any time by using the manual validation actions that are available in Oxygen XML Editor.

#### Manual Validation Actions

To manually validate the currently edited document, use one of the following actions:

- **Validate (Ctrl + Shift + V (Command + Shift + V on macOS))**
  
  Available from the Validation drop-down menu on the toolbar, the Document > Validate menu, or from the Validate submenu when invoking the contextual menu on one or more JSON documents in the Project view (on page 402).
Available from the Validation drop-down menu on the toolbar or the Document > Validate menu. This action opens a dialog box that allows you to specify a schema for validating the current document (on page 1123).

**Note:**

The Validate with action does not work for files loaded through a custom protocol plugin (on page 2427) developed independently and added to Oxygen XML Editor after installation.

### Validate with Schema

Available from the Validate submenu when invoking the contextual menu on one or more JSON documents in the Project view (on page 402). This action opens a dialog box that allows you to specify a JSON or Schematron schema to be used for the validation.

### Other Validation Options

**Tip:**

If a large number of validation errors are detected and the validation process takes too long, you can limit the maximum number of reported errors in the Document Checking preferences page (on page 226).

Related Information:

- [Automatic Validation](on page 1115)
- [Presenting Validation Errors in JSON Documents](on page 1116)

### Presenting Validation Errors in JSON Documents

Validation errors and warnings in JSON documents are presented in various locations within the interface.

#### Validation Marker Locations

Validation issues are marked in the following locations:

- In the main editing pane, with the issue underlined in a color according to the type of issue.
- In the right-side vertical stripe, with a marker that is colored according to the type of issue.
- In the Outline view, with an icon that is colored according to the type of issue.

#### Validation Marker Colors

The colors for each type of issue are as follows:
• **Validation Errors** [Red] - By default, the underline in the editing pane, the marker in the right vertical stripe, and the foreground color of the attribute in the Attributes view are colored in red.

• **Validation Warnings** [Yellow] - By default, the underline in the editing pane, the marker in the right vertical stripe, and the foreground color of the attribute in the Attributes view are colored in yellow.

• **Validation Info** [Blue] - By default, the underline in the editing pane, the marker in the right vertical stripe, and the foreground color of the attribute in the Attributes view are colored in blue.

You can configure the color for each type in the **Document Checking preferences page** *(on page 226)*.

### Validation Markers in the Right-Side Stripe

Also, the stripe on the right side of the editor panel is designed to display the issues found during the validation process and to help you locate them in the document. The stripe contains the following:

#### Upper Part of the Stripe

A success indicator square will turn green if the validation is successful or only info messages are found, red if validation errors are found, or yellow if only validation warnings are found. More details about the issues are displayed in a tooltip when you hover over indicator square. If there are numerous problems, only the first three are presented in the tooltip.

#### Middle Part of the Stripe

Errors are presented with red markers, warnings with yellow markers, and info message with blue markers. If you want to limit the number of markers that are displayed, open the **Preferences dialog box** *(Options > Preferences)* *(on page 124)*, go to Editor > Document checking, and specify the desired limit in the **Maximum number of validation highlights** option *(on page 226)*.

Clicking a marker will highlight the corresponding text area in the editor. The validation message is also displayed both in a tooltip (when hovering over the marker) and in the message area on the bottom of the editor panel (clicking the **Document checking options** button opens the **Document Checking preferences page** *(on page 226)*).

#### Bottom Part of the Stripe

Two navigation arrows (分子) can be used to jump to the next or previous issue. The same actions can be triggered from **Document > Automatic validation > Next Error/Highlight** *(Ctrl + Period (Command + Period on macOS))* and **Document > Automatic validation > Previous Error/Highlight** *(Ctrl + Comma (Command + Comma on macOS))*). Also, the **Remove All** button can be used to clear all the validation markers.

### Hovering Over Validation Issues

Hovering over a validation issue presents a tooltip message with more details about the problem. Also, when hovering over an issue, pressing **F2** will change the focus to the tooltip.
Details About Validation Issues

- Information about the issue is also displayed in the message area on the bottom of the editor panel (clicking the Document checking options button opens the Document Checking preferences page (on page 226) where you can configure some validation options (such as the colors used to present the validation issues). Some validation messages have an icon and clicking it opens a dialog box with additional information and a link to specifications.

- If you want to see all the validation messages grouped in the Results view (on page 548), use the Validate action from the toolbar or Document > Validate menu. To see more information about a validation message, right-click the item in the Results view and select Show message. Some validation messages have an icon in the Info column and clicking it opens a dialog box with additional information and a link to specifications.

Creating a JSON Validation Scenario

Validation scenarios can be used to associate one or more JSON Schemas with a JSON document (on page 1122). Oxygen XML Editor also includes a built-in JSON Schematron Validator engine that can be specified in the validation scenario to validate JSON documents against a specified Schematron schema.

Creating a JSON Validation Scenario

To create a validation scenario, follow these steps:

1. Select the Configure Validation Scenario(s) action in one of the following ways:
   - From the Validation toolbar drop-down menu.
   - From the Document > Validate menu.
   - From the Validate submenu, when invoking the contextual menu on a file in the Project view (on page 402).

   **Step Result:** The Configure Validation Scenario(s) dialog box is displayed.

2. Click the New button.

   **Step Result:** A validation scenario configuration dialog box is displayed.
This scenario configuration dialog box allows you to configure the following information and options:

**Name**

The name of the validation scenario.

**Storage**

You can choose between storing the scenario in the [Project Options](on page 3222) or [Global Options](on page 3219).

**URL of the file to validate**

The URL of the main module that includes the current module. It is also the entry module of the validation process when the current one is validated. To edit the URL, double-click its cell and specify the URL of the main module by doing one of the following:

- Enter the URL in the text field or select it from the drop-down list.
- Use the ![Browse](on page) drop-down button to browse for a local, remote, or archived file.
- Use the ![Insert Editor Variable](on page) button to insert an editor variable (on page 323) or a custom editor variable (on page 332).
Figure 393. Insert an Editor Variable

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>${Desktop}</td>
<td>My Desktop</td>
</tr>
<tr>
<td>${start-dir}</td>
<td>Start directory of custom validator</td>
</tr>
<tr>
<td>${standard-params}</td>
<td>List of standard params for command line</td>
</tr>
<tr>
<td>${fn}</td>
<td>The current file name without extension</td>
</tr>
<tr>
<td>${currentFileURL}</td>
<td>The path of the currently edited file (URL)</td>
</tr>
<tr>
<td>${foud}</td>
<td>The path of current file directory (URL)</td>
</tr>
<tr>
<td>${frameworks}</td>
<td>Oxygen frameworks directory (URL)</td>
</tr>
<tr>
<td>${pdu}</td>
<td>Project directory (URL)</td>
</tr>
<tr>
<td>${oxygenHome}</td>
<td>Oxygen installation directory (URL)</td>
</tr>
<tr>
<td>${home}</td>
<td>The path to user home directory (URL)</td>
</tr>
<tr>
<td>${pj}</td>
<td>Project name</td>
</tr>
<tr>
<td>${env(VAR_NAME)}</td>
<td>Value of environment variable VAR_NAME</td>
</tr>
<tr>
<td>${system(var.name)}</td>
<td>Value of system variable var.name</td>
</tr>
</tbody>
</table>

**File type**

The type of the document that is validated in the current validation unit. Oxygen XML Editor automatically selects the file type depending on the value of the URL of the file to validate field.

**Validation engine**

You can choose between the following types of validation engines for validating JSON documents:

- **Default engine** - The built-in JSON Validator will be used. For JSON Schema documents, this type should not be chosen unless the document has a schema version specified.
- **JSON Schema Validator** - This type is for JSON Schema documents only. It will use the version specified in the JSON Schema, or if a version is not specified, the JSON Schema draft-04 will be used.
- **JSON Schematron Validator** - The built-in JSON Schematron Validator will be used to validate JSON documents against a specified Schematron schema.

---

**Note:**

For proper error localization, the root element of the Schematron schema should include the `@queryBinding` attribute with the value of `xslt2` after the Schematron namespace declaration:

```xml
<sch:schema xmlns:sch="http://purl.oclc.org/dsdl/schematron" query Binding="xslt2"/>
```

**Automatic validation**
If this option is selected, the validation operation defined by this row is also applied by the automatic validation feature (on page 775). If the Automatic validation feature is disabled in the Document Checking preferences page (on page 226), then this option is ignored, as the preference setting has a higher priority.

**Schema**

Displays the specified schema.

**Specify Schema**

Opens the Specify Schema dialog box that allows you to set a schema to be used for validating JSON documents.

**Move Up**

Moves the selected scenario up one spot in the list.

**Move Down**

Moves the selected scenario down one spot in the list.

**Add**

Adds a new validation unit to the list.

**Remove**

Removes an existing validation unit from the list.

3. Configure any of the existing validation units according to the information above. You can use the buttons at the bottom of the table to add, remove, or move validation units.

4. Click OK.

**Result:** The newly created validation scenario will now be included in the list of scenarios in the Configure Validation Scenario(s) dialog box. You can select the scenario in this dialog box to associate it with the current document and click the Apply associated button to run the validation scenario.

**Sharing JSON Validation Scenarios**

The validation scenarios and their settings can be shared with other users by saving them at project level (on page 3222) or by exporting them to a specialized scenarios file (on page 323) that can then be imported. When you create a new validation scenario or edit an existing one, there is a Storage option to control whether the scenarios are stored in Project Options (on page 3222) or Global Options (on page 3219).

Selecting Project Options (on page 3222) stores the scenario in the project file and can be shared with other users that have access to the project. If your project is saved on a source versioning/sharing system (CVS, SVN, Source Safe, etc.) then your team will have access to the scenarios that you define. When you create a scenario at the project level, the URLs from the scenario become relative to the project URL.
Oxygen XML Editor 24.1 | 8 - Editing Documents | 1122
Selecting Global Options (on page 3219) stores the scenario in the global options that are stored in the user
home directory.
You can also change the storage options of existing validation scenarios by using the Change storage action
from the contextual menu of the list of scenarios in the Configure Validation Scenario(s) dialog box.

Resolving References with an XML Catalog
If a reference to a remote JSON schema must be used but a local copy of the schema should actually be
preferred for performance reasons, the reference can be resolved to the local copy with an XML Catalog (on
page 3225).
For example, if the JSON schema contains a reference to a remote schema such as:
{"$ref": "http://json-schema.org/example/geo.json"}

the reference can be resolved to a local copy of the schema by inserting the following catalog entry:
<uri name="http://json-schema.org/example/geo.json" uri="schemas/geo.json"/>

Related Information:
Working with XML Catalogs (on page 826)

Associating a Schema to JSON Documents
To provide as-you-type validation and to compute valid proposals for the Content Completion Assistant (on
page 3217), Oxygen XML Editor requires a schema to be associated with the JSON document. The schema
specifies how the internal structure is defined.

Detecting the Schema(s) for Validation and Content Completion
For validation, Oxygen XML Editor tries to detect the JSON Schema by searching in the following order:
1. The schema referenced in validation stages from the validation scenario(s) (on page 1122) associated
with the current JSON document.
2. If a schema is not detected, then it falls back to the schema associated directly in the JSON document
(on page 1125).
Tip:
To quickly open the schema used for validating the current document, select the

Open

Associated Schema action from the toolbar (or Document > Schema menu).

Associating a Schema Through a Validation Scenario
Oxygen XML Editor uses the rules defined in the detected schema to report errors and warnings during
automatic and manual validations that help maintain the structural integrity of your JSON documents. Oxygen


XML Editor includes built-in validation engines for validating JSON documents against a JSON Schema or Schematron schema. There are several methods that can be used to validate JSON document with a schema.

**Configure a Validation Scenario and Specify the Schema**

You can specify the schema to be used for validation directly in the JSON validation scenario (on page 1118). To associate a schema to a validation scenario to be used whenever the scenario is invoked, follow these steps:

1. Select the **Configure Validation Scenario(s)** from the **Validation** toolbar drop-down menu, from the **Document > Validate** menu, or from the **Validate** submenu when invoking the contextual menu on a JSON file in the Project view (on page 402).
2. Click the **New** button to create a new validation scenario (on page 1118) or the **Edit** button to modify an existing one.
3. Add or configure validation units according to your needs. For details about all of the configuration options, see Creating a JSON Validation Scenario (on page 1118).
4. Click the **Specify Schema** button to select the schema to be associated with the validation unit.
5. Click **OK** on both dialog boxes.

**Result:** The schema is now associated with that validation scenario whenever it is invoked.

**Use the Validate with Action to Specify a Schema for Validating the Current Document**

To validate the current document using a specified schema, follow these steps:

1. Select the **Validation with** action from the **Validation** drop-down menu on the toolbar (or **Document > Validate** menu).

**Step Result:** The **Validate with** dialog box is displayed:

*Figure 394. Validate with Dialog Box*

This dialog box contains the following options:

- **URL** - Allows you to specify or select a URL for the schema. It also keeps a history of the last used schemas. The URL must point to the schema file that can be loaded from the local disk or from a remote server through HTTP(S), FTP(S) or a custom protocol (on page 2444). You can
specify the URL by using the text field, the history drop-down, the Insert Editor Variables (on page 323) button, or the browsing actions in the Browse drop-down list.

- **Schema type** - You can select one of the following two types (other types of schema will not work with JSON documents):
  - JSON - Used for validating JSON documents against a specified JSON Schema.
  - Schematron - Used for validating JSON documents against a specified Schematron schema. You can also select a Schematron phase that you want to use for the validation.

Note:
For proper error localization, the root element of the Schematron schema should include the @queryBinding attribute with the value of xsIt2 after the Schematron namespace declaration:

```xml
<sch:schema xmlns:sch="http://purl.oclc.org/dsdl/schematron" queryBinding="xslt2">
```

2. Select the schema to be associated with the manual validation.
3. Click OK.

Result: The current document is validated using the schema you specified.

**Use the Validate with Schema Action to Specify a Schema for Validating all Selected JSON Documents**

To validate multiple JSON documents using a specified schema, follow these steps:

1. Select all the JSON documents you want to validate in the Project view.
2. Invoke the contextual menu (right-click) and select the Validate with Schema action from the Validate submenu.

**Step Result:** The Validate with dialog box is displayed:

**Figure 395. Validate with Dialog Box**

This dialog box contains the following options:
URL - Allows you to specify or select a URL for the schema. It also keeps a history of the last used schemas. The URL must point to the schema file that can be loaded from the local disk or from a remote server through HTTP(S), FTP(S) or a custom protocol (on page 2444). You can specify the URL by using the text field, the history drop-down, the Insert Editor Variables (on page 323) button, or the browsing actions in the Browse drop-down list.

Schema type - You can select one of the following two types (other types of schema will not work with JSON documents):

- JSON - Used for validating JSON documents against a specified JSON Schema.
- Schematron - Used for validating JSON documents against a specified Schematron schema. You can also select a Schematron phase that you want to use for the validation.

Note:
For proper error localization, the root element of the Schematron schema should include the \@queryBinding attribute with the value of xsIt2 after the Schematron namespace declaration:

<sch:schema xmlns:sch="http://purl.oclc.org/dsdl/schematron" queryBinding="xsIt2">

3. Select the JSON schema that you want to use to validate all selected JSON documents.
4. Click OK.

Result: The selected JSON documents are validated using the JSON schema you specified.

Associating a Schema Directly in JSON Documents

Associate Schema Action

The schema used by the Content Completion Assistant (on page 3217) and document validation engine can be associated with the current document by using the Associate Schema action. The association can specify a relative file path or a URL of the schema.

To associate a JSON Schema to the current JSON document, follow these steps:

1. Select the Associate Schema action from the toolbar (or Document > Schema menu).

   Step Result: The Associate Schema dialog box is displayed:
This dialog box contains the following options for JSON documents:

- **URL** - Allows you to specify or select a URL for the schema. It also keeps a history of the last used schemas. The URL must point to the schema file that can be loaded from the local disk or from a remote server through HTTP(S), FTP(S) or a custom protocol (on page 2444).

- **Use path relative to file location** - Select this option if the JSON instance document and the associated schema contain relative paths. The location of the schema file is inserted in the JSON instance document as a relative file path. This practice allows you, for example, to share these documents with other users without running into problems caused by multiple project locations on physical disk.

1. Select the JSON Schema that will be associated with the JSON document.
2. Click **OK**.

**Result:** A `$schema` property is added at the beginning of the document with its value set to the specified URL. If the document already contained a schema association, the old association is replaced with the new one.

**Tip:**
To quickly open the schema used for validating the current document, select the **Open Associated Schema** action from the toolbar (or **Document > Schema** menu).

### Content Completion Assistant in JSON

Oxygen XML Editor includes an intelligent **Content Completion Assistant (on page 3217)** that offers proposals for inserting JSON structures that are valid at the current editing location.

The **Content Completion Assistant** is enabled by default. To disable it, open the **Preferences** dialog box (Options > Preferences) (on page 124), go to Editor > Content Completion, and deselect the **Enable content completion** option (on page 211).

**Figure 397. Content Completion Assistant in JSON**

![Content Completion Assistant in JSON](image)

### Content Completion and the Associated Schema

The **Content Completion Assistant** feature is schema-driven and the list of proposals in the **Content Completion Assistant (on page 3217)** depend on the associated JSON Schema. For information about ways to associate a schema to a JSON document, see the **Associating a Schema to JSON Documents (on page 1122)** section.
Using the Content Completion Assistant in JSON

The feature is activated in **Text** mode for JSON documents by:

- Typing a quote symbol (') to insert a property or value.
- Pressing **Ctrl + Space** (Command + Space on macOS) or **Alt + ForwardSlash** (Command + Alt + ForwardSlash on macOS).

The feature is activated in **Author** mode by using the **Enter** key.

You can navigate through the list of proposals by using the **Up** and **Down** keys on your keyboard. In some cases, the **Content Completion Assistant** displays a documentation window with information about the particular proposal (on page 1127). You can also change the size of the documentation window by dragging its top, right, and bottom borders.

To insert the selected proposal, press **Enter** or **Tab**.

**Types of Proposals Listed in the Content Completion Assistant for JSON**

The proposals that populate the **Content Completion Assistant** for JSON documents depend on the structure defined in the associated JSON Schema. The types of structure proposed in the content completion window include:

- JSON properties
- JSON values
- JSON arrays
- JSON objects

The number and type of proposals displayed by the **Content Completion Assistant** is dependent on the cursor's current position in the JSON document and the child items displayed within a given context are defined by the structure of the specified JSON Schema.

**Code Templates in the Content Completion**

Oxygen XML Editor includes a set of built-in code templates for JSON documents that can be selected from the **Content Completion Assistant**. The code templates are displayed with a 🔄 symbol in the content completion list. You can also define your own code templates and share them with others. For more information, see **Code Templates** (on page 536).

**Schema Annotations in JSON Content Completion**

A schema annotation is a documentation snippet that appears in the **Content Completion Assistant** (on page 3217) offering more information about the current proposal.

This feature is enabled by default, but you can disable it by deseleting the **Show annotations in Content Completion Assistant** (on page 216) option in the **Annotations** preferences page.
Collecting Annotations from the JSON Schema

In a JSON Schema, the annotations are specified in the value of the `title` and `description` properties like this:

```json
"idType": {
  "title": "The 'id' property",
  "description": "Specifies a required ID for this person.",
  "type": "string",
  "maxLength": 20
}
```

Syntax Highlighting in JSON Documents

Oxygen XML Editor supports syntax highlighting in Text mode to make it easier to read the semantics of the structured content by displaying each type of code in different colors and fonts.

To customize the colors or styles used for the syntax highlighting colors for JSON files, follow these steps:

1. Open the Preferences dialog box (Options > Preferences) (on page 124).
2. Go to Editor > Syntax Highlight (on page 224).
3. Select and expand the JSON section in the top pane.
4. Select the component you want to change and customize the colors or styles using the selectors to the right of the pane.

You can see the effects of your changes in the Preview pane.

Related Information:
Syntax Highlight Preferences (on page 224)

Folding in JSON

In a large JSON document, the data enclosed in the curly bracket characters `{}` can be collapsed so that only the needed data remains in focus. The folding features available for XML documents (on page 528) are also available in JSON documents.

JSON Outline View

The Outline view for JSON documents displays the list of all the components of the JSON document you are editing. By default, it is displayed on the left side of the editor. If the view is not displayed, it can be opened by selecting it from the Window > Show View menu.
Outline View Features

The **Outline** view allows you to:

- Quickly navigate through the document by selecting nodes in the **Outline** tree.
- Move elements by dragging them to a new position in the tree structure.
- Highlight elements in the editor area. It is synchronized with the editor area, so when you make a selection in the editor area, the corresponding nodes are highlighted in the **Outline** view, and vice versa.
- View document errors, as they are highlighted in the **Outline** view. A tooltip also provides more information about the nature of the error when you hover over the faulted element.

Outline View Interface

By default, it is displayed on the left side of the editor. If the view is not displayed, it can be opened by selecting it from the **Window > Show View** menu.

The upper part of the **Outline** view contains a filter box that allows you to focus on the relevant components. Type a text fragment in the filter box and only the components that match it are presented. For advanced usage you can use wildcard characters (such as `*` or `?`) and separate multiple patterns with commas.

It also includes a **Settings** menu in the top-right corner that presents the following options to help you filter the view even further.

- **Filter returns exact matches**
  
  The text filter of the **Outline** view returns only exact matches.

- **Selection update on cursor move**
Controls the synchronization between Outline view and source document. The selection in the Outline view can be synchronized with the cursor moves or the changes in the editor. Selecting one of the components from the Outline view also selects the corresponding item in the source document.

Flat presentation mode of the filtered results

When active, the application flattens the filtered result elements to a single level.

Drag and Drop Actions in the Outline View

Entire JSON properties, objects, and arrays can be moved or copied in the edited document using only the mouse in the Outline view with drag-and-drop operations. Several drag and drop actions are possible:

- If you drag a JSON node in the Outline view and drop it on another node, then the dragged node will be moved after the drop target.
- If you hold the mouse pointer over the drop target for a short time before the drop then the drop target node will be expanded first and the dragged node will be moved inside the drop target.
- You can also drop a node before or after another node if you hold the mouse pointer towards the upper or lower part of the target. A marker will indicate whether the drop will be performed before or after the target node.
- If you hold down the Ctrl (Command on macOS) key after dragging, a copy operation will be performed instead of a move.

Contextual Menu Actions

The following actions are available in the contextual menu of the JSON Outline view:

- Cut
  Cuts the currently selected component.
- Copy
  Copies the currently selected component.
- Paste
  Pastes the copied component.
- Delete
  Deletes the currently selected component.
- Expand More
  Expands the structure of a component in the Outline view.
- Collapse All
  Collapses the structure of all the component in the Outline view.
JSON to XML Converter

Online JSON to XML Converter

Attention:
For a simple ONLINE tool for converting a single JSON file to XML, or vice versa, go to: https://www.oxygenxml.com/xml_json_converter.html.

Converting JSON to XML in Oxygen

Oxygen XML Editor includes a useful and simple tool for converting JSON files to XML. The JSON to XML action for invoking the tool can be found in the Tools > JSON Tools menu.

To convert a JSON document to XML, follow these steps:

1. Select the JSON to XML action from the Tools > JSON Tools menu.

   The JSON to XML dialog box is displayed:

   ![Figure 399. JSON to XML Dialog Box](image)

2. Choose or enter the Input URL of the JSON document.
3. Choose the path of the Output file that will contain the resulting XML document.
4. Select the Open in Editor option to open the resulting XML document in the main editing pane.
5. Click the Convert button.

Result: The original JSON document is now converted to an XML document.
Conversion Details

- If the JSON document has more than one property on the first level, the converted XML document will have an additional root element called `<JSON>`.

For example, the following JSON document:

```json
{
   "personnel": {
      "person": [
      {
         "name": "Boss"
      },
      {
         "name": "Worker"
      }
   },
   "id": "personnel-id"
}
```

it is converted to:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<JSON>
  <personnel>
    <person>
      <name>Boss</name>
    </person>
  </personnel>
</JSON>
```
• If the JSON document is an array, the converted XML document will have a root element called `<array>` and for each item within the array, another `<array>` is created.

```json
[
  {"name": "Boss"},
  {"name": "Worker"}
]
```

it is converted to:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<array>
  <array>
    <name>Boss</name>
  </array>
  <array>
    <name>Worker</name>
  </array>
</array>
```

• If the name of a JSON property contains characters that are not valid in XML element names (for example, `$`), then the invalid characters will be escaped as its hexadecimal equivalent in the converted XML.

```json
{"$id": "personnel-id"}
```

is converted to:

```xml
<_X24_id>personnel-id</_X24_id>
```

Related Information:

XML to JSON Converter (on page 1134)
XML to JSON Converter

Online XML to JSON Converter

**Attention:**

For a simple ONLINE tool for converting a single XML file to JSON, or vice versa, go to: [https://www.oxygenxml.com/xml_json_converter.html](https://www.oxygenxml.com/xml_json_converter.html).

Converting XML to JSON in Oxygen

Oxygen XML Editor includes a useful and simple tool for converting XML files to JSON. The XML to JSON action for invoking the tool can be found in the Tools > JSON Tools menu.

To convert an XML document to JSON, follow these steps:

1. Select the XML to JSON action from the Tools > JSON Tools menu.

   **Step Result:** The XML to JSON dialog box is displayed:

   ![XML to JSON Dialog Box](image)

2. Choose or enter the Input URL of the XML document.
3. Choose the path of the Output file that will contain the resulting JSON document.
4. Select how you want empty elements to be converted (default is object).
5. Select the Open in Editor option to open the resulting JSON document in the main editing pane.
6. Click the Convert button.

**Result:** The original XML document is now converted to a JSON document.
Conversion Details

- Some XML components are ignored (e.g. comments and processing instructions).

- If any elements contain attributes in the XML document, the attributes are converted to properties in the converted JSON document. If the XML document contains more than one element with the same name, they will be converted into an array of object in the converted JSON document.

For example, the following XML document:

```xml
<personnel>
  <person id="person.one">
    <name>Boss</name>
  </person>
  <person id="person.two">
    <name>Worker</name>
  </person>
</personnel>
```

it is converted to:

```json
{
  "personnel": {
    "person": [
      {
        "id": "harris.anderson",
        "photo": "personal-images/harris.anderson.jpg",
        "name": {
          "given": "Harris",
          "family": "Anderson"
        },
        "email": "harris.anderson@example.com",
        "link": [{"subordinate": "robert.taylor", "url": "http://www.example.com/na/harr"}]
      },
      {
        "id": "robert.taylor",
        "photo": "personal-images/robert.taylor.jpg",
        "name": {
          "given": "Robert",
          "family": "Taylor"
        },
        "email": "robert.taylor@example.com",
        "link": [{"manager": "harris.anderson"},
                  "url": "http://www.example.com/na/tobe"
        ]
      },
      {
        "id": "helen.jackson",
        "photo": "personal-images/helen.jackson.jpg",
        "name": {
          "given": "Helen",
          "family": "Jackson"
        },
        "email": "helen.jackson@example.com",
        "link": [{"supervisor": "harris.anderson"},
                  "url": "http://www.example.com/na/hel"
        ]
      }
    ]
  }
}
```
"id": "person.one",
"name": "Boss"
],
[
"id": "person.two",
"name": "Worker"
]
}

- If the XML document contains elements with mixed content (text plus elements), the converted JSON document will contain a `#text` property with its value set as the text content. If there are multiple text nodes, the subsequent `#text` properties will contain a number (e.g. `#text1`, `#text2`). If there are multiple elements with the same name, the first property will have the element name and the subsequent properties will contain a number (e.g. b, b#1, b#2).

```xml
<p>This <b>is</b> an <b>example</b>!</p>
```

is converted to:

```json
{
  "p": {
    "#text": "This",
    "b": "is",
"#text1": " an",
    "b#1": "example",
"#text2": "!"
  }
}
```

- If the XML document contains element names that contain hexadecimal codes (for example, if they were escaped during a JSON to XML conversion (on page 1131)), it will be converted to the normal character value in the converted JSON document.

```xml
<_X24_id>personnel-id</_X24_id>
```

is converted to:

```json
{"$id": "personnel-id"}
```

Related Information:

JSON to XML Converter (on page 1131)
JSON to YAML Converter

Converting JSON to YAML in Oxygen

Oxygen XML Editor includes a useful and simple tool for converting JSON files to YAML. The JSON to YAML action for invoking the tool can be found in the Tools > JSON Tools menu.

To convert a JSON document to YAML, follow these steps:

1. Select the JSON to YAML action from the Tools > JSON Tools menu.

   The JSON to YAML dialog box is displayed:

   ![JSON to YAML Dialog Box](image)

2. Choose or enter the JSON URL for the document you want to convert.

3. Choose the path of the Output file that will contain the resulting YAML document.

4. [Optional] Select the Open in Editor option to open the resulting YAML document in the main editing pane.

5. Click the Convert button.

   Result: The original JSON document is now converted to a YAML document.

Related Information:
YAML to JSON Converter (on page 1137)

YAML to JSON Converter

Converting YAML to JSON in Oxygen

Oxygen XML Editor includes a useful and simple tool for converting YAML files to JSON. It even works on files that consist of multiple YAML documents, each separated by three dashes (---), in which case the conversion creates multiple JSON files with a number in the name.

The YAML to JSON action for invoking the tool can be found in the Tools > JSON Tools menu.

To convert a YAML document to JSON, follow these steps:
1. Select the **YAML to JSON** action from the **Tools > JSON Tools** menu.

   The **YAML to JSON** dialog box is displayed:

   ![YAML to JSON Dialog Box](image)

2. Choose or enter the **YAML URL** for the document you want to convert.
3. Choose the path of the **Output file** that will contain the resulting JSON document.
4. **[Optional]** Select the **Open in Editor** option to open the resulting JSON document in the main editing pane.
5. Click the **Convert** button.

**Result:** The original YAML document is now converted to a JSON document.

**Related Information:**

- **JSON to YAML Converter (on page 1137)**

### Contextual Menu Actions in JSON Documents

When editing JSON documents, Oxygen XML Editor provides the following actions in the contextual menu:

- **Cut, Copy, Paste**
  
  Executes the typical editing actions on the currently selected content.

- **Copy JSON Pointer**
  
  Creates a **JSON Pointer** at the current cursor location and copies the expression that denotes the JSON pointer to the system clipboard.

- **Copy XPath**
  
  Copies the XPath expression of the current property from the current editor to the clipboard.

- **Go to Matching Bracket (Ctrl + Shift + G (Command + Shift + G on macOS))**
  
  Moves the cursor to the end bracket that matches the start bracket, or vice versa.

**Source submenu**

This submenu includes the following actions:
**To Lower Case**

Converts the content selection to lower case characters. This works with contiguous and multiple selections.

**To Upper Case**

Converts the selected content to upper case characters. This works with contiguous and multiple selections.

**Capitalize Lines**

It capitalizes the first letter found on every new line that is selected. Only the first letter is affected, the rest of the line remains the same. If the first character on the new line is not a letter then no changes are made.

**Convert Hexadecimal Sequence to Character (Ctrl + Shift + X (Command + Shift + X on macOS))**

Converts a sequence of hexadecimal characters to the corresponding Unicode character (on page 463). The action can be invoked if there is a selection containing a valid hexadecimal sequence or if the cursor is placed at the right side of a valid hexadecimal sequence. A valid hexadecimal sequence can be composed of 2 to 4 hexadecimal characters and may or may not be preceded by the \texttt{0x} or \texttt{0X} prefix. Examples of valid sequences and the characters they will be converted to:

- \texttt{0x0045} will be converted to \texttt{E}
- \texttt{0X0125} to \texttt{ĥ}
- \texttt{265} to \texttt{ɥ}
- \texttt{2190} to \texttt{←}

**Note:**

For more information about finding the hexadecimal value of a character, see Finding the Decimal, Hexadecimal, or Character Entity Equivalent (on page 466).

**Base64 Encode/Decode submenu**

This submenu include the following actions for encoding or decoding base 64 schemes:

**Import File to Encode and Insert**

Encodes a file and then inserts the encoded content into the current document at the cursor position.

**Decode Selection and Export to File**
Decodes a selection of text from the current document and then exports (saves) the result to another file.

**Encode Selection**

Replaces a selection of text with the result of encoding that selection. By default, a dialog box is displayed that allows you to select the encoding to use. There is an option to choose to not show this dialog box in the future. In this case, the encoding that is specified in the *Encoding for Base64, Base32, Hex conversions option in the Encoding preferences page (on page 169)* will be used. Likewise, the same is true if the *Show the dialog box for choosing the encoding for Base64, Base32, Hex conversions* option is not selected in the *Messages preference page (on page 308)*.

**Decode Selection**

Replaces a selection of text with the result of decoding that selection. By default, a dialog box is displayed that allows you to select the encoding to use. There is an option to choose to not show this dialog box in the future. In this case, the encoding that is specified in the *Encoding for Base64, Base32, Hex conversions option in the Encoding preferences page (on page 169)* will be used. Likewise, the same is true if the *Show the dialog box for choosing the encoding for Base64, Base32, Hex conversions* option is not selected in the *Messages preference page (on page 308)*.

**Modify All Matches**

Use this option to modify (in-place) all the occurrences of the selected content (or the contiguous fragment where the cursor is located). When you use this option, a thin rectangle replaces the highlights and allows you to start editing. If matches with different letter cases are found, a dialog box is displayed that allows you select whether you want to modify only matches with the same letter case or all matches.

**Base32 Encode/Decode submenu**

This submenu include the following actions for encoding or decoding base32 schemes:

**Import File to Encode and Insert**

Encodes a file and then inserts the encoded content into the current document at the cursor position.

**Decode Selection and Export to File**
Decodes a selection of text from the current document and then exports (saves) the result to another file.

**Encode Selection**

Replaces a selection of text with the result of encoding that selection. By default, a dialog box is displayed that allows you to select the encoding to use. There is an option to choose to not show this dialog box in the future. In this case, the encoding that is specified in the [Encoding for Base64, Base32, Hex conversions option in the Encoding preferences page](on page 169) will be used. Likewise, the same is true if the [Show the dialog box for choosing the encoding for Base64, Base32, Hex conversions option](on page 308).

**Decode Selection**

Replaces a selection of text with the result of decoding that selection. By default, a dialog box is displayed that allows you to select the encoding to use. There is an option to choose to not show this dialog box in the future. In this case, the encoding that is specified in the [Encoding for Base64, Base32, Hex conversions option in the Encoding preferences page](on page 169) will be used. Likewise, the same is true if the [Show the dialog box for choosing the encoding for Base64, Base32, Hex conversions option](on page 308) is not selected in the Messages preference page.

**Modify All Matches**

Use this option to modify (in-place) all the occurrences of the selected content (or the contiguous fragment where the cursor is located). When you use this option, a thin rectangle replaces the highlights and allows you to start editing. If matches with different letter cases are found, a dialog box is displayed that allows you select whether you want to modify only matches with the same letter case or all matches.

**Hex Encode/Decode submenu**

This submenu include the following actions for encoding or decoding hex schemes:

**Import File to Encode and Insert**

Encodes a file and then inserts the encoded content into the current document at the cursor position.

**Decode Selection and Export to File**
Decodes a selection of text from the current document and then exports (saves) the result to another file.

**Encode Selection**

Replaces a selection of text with the result of encoding that selection. By default, a dialog box is displayed that allows you to select the encoding to use. There is an option to choose to not show this dialog box in the future. In this case, the encoding that is specified in the **Encoding for Base64, Base32, Hex conversions** option in the **Encoding preferences page (on page 169)** will be used. Likewise, the same is true if the Show the dialog box for choosing the encoding for Base64, Base32, Hex conversions option is not selected in the **Messages preference page (on page 308)**.

**Decode Selection**

Replaces a selection of text with the result of decoding that selection. By default, a dialog box is displayed that allows you to select the encoding to use. There is an option to choose to not show this dialog box in the future. In this case, the encoding that is specified in the **Encoding for Base64, Base32, Hex conversions** option in the **Encoding preferences page (on page 169)** will be used. Likewise, the same is true if the Show the dialog box for choosing the encoding for Base64, Base32, Hex conversions option is not selected in the **Messages preference page (on page 308)**.

**Modify All Matches**

Use this option to modify (in-place) all the occurrences of the selected content (or the contiguous fragment where the cursor is located). When you use this option, a thin rectangle replaces the highlights and allows you to start editing. If matches with different letter cases are found, a dialog box is displayed that allows you select whether you want to modify only matches with the same letter case or all matches.

**Join and Normalize Lines (Ctrl + J (Command + J on macOS))**

For the current selection, this action joins the lines by replacing the *line separator* with a single space character. It also normalizes the whitespaces by replacing a sequence of such characters with a single space.

**Insert new line after (Ctrl + Alt + Enter (Command + Alt + Enter on macOS))**

This action has the same result as moving the cursor to the end of the current line and pressing the **ENTER** key.

**Modify All Matches**
Use this option to modify (in-place) all the occurrences of the selected content (or the contiguous fragment where the cursor is located). When you use this option, a thin rectangle replaces the highlights and allows you to start editing. If matches with different letter cases are found, a dialog box is displayed that allows you select whether you want to modify only matches with the same letter case or all matches.

**Go to Definition**

Navigates to the definition of the current property.

**Open submenu**

The following actions are available in this submenu:

**Open File at Cursor**

Opens the file at the cursor position in a new panel. If the file path represents a directory path, it will be opened in system file browser. If the file at the specified location does not exist, an error dialog box is displayed and it includes a Create new file button that starts the New document wizard. This allows you to choose the type or the template for the file. If the action succeeds, the file is created with the referenced location and name and is opened in a new editor panel. If the file is an image file, it will be opened in the Image Preview pane (on page 469).

**Open File at Cursor in System Application**

Opens the file (identified by its link) or web page (identified by a web link) found at the cursor position. The target is opened in the default system application associated with that file type.

**Compare**

Opens the current file in the Compare Files tool (on page 474).

**Transforming and Querying JSON Documents**

Oxygen XML Editor provides the ability to transform JSON documents to XML or HTML through XSLT or XQuery processing. You also have access to some powerful tools for querying JSON through XPath expressions or XQuery.

For more information about transforming and querying in JSON, watch our video demonstration:

https://www.youtube.com/embed/1LHoMhEFagA

**Transforming JSON Documents with XSLT**

It is possible to transform JSON documents through XSLT processing. To do so, follow these steps:
1. Create an XSLT 3.0 stylesheet that has the `xsl:initial-template`. You can use one of the following two templates available in the New Document Wizard.
   - **XSLT Stylesheet for JSON** - Processes a JSON document by using a `json-doc()` function and matches the JSON properties from the JSON map.
   - **XSLT Stylesheet for JSON to XML** - Processes a JSON document by using a `json-to-xml()` function and matches the converted XML content.

2. Create a new **XSLT transformation** scenario for your stylesheet.

3. Reference the JSON document that you want to transform using one of these two methods:
   - In the transformation scenario, click the **Parameters** button in the XSLT tab and add a parameter that specifies the URL to your JSON document in its value. For example, if you are transforming one of the built-in templates mentioned above, the `input` parameter is added by default and you could specify the URL in its value.
   - Specify the URL to your JSON document in the stylesheet you created. For example, if you use one of the built-in templates mentioned above, you would specify the URL in the value of the `input` parameter (in the `xsl:param` element).

4. Run the transformation.

---

**Tip:**

There are some sample files in the `[OXYGEN_INSTALL_DIR]/samples/json/transform` folder that can be used to transform a JSON document to XML or HTML.

---

**Related Information:**

- Blog: Transforming JSON
- XSLT Functions on JSON Data

### Transforming JSON Documents with XQuery

It is possible to transform JSON documents through XQuery processing. To do so, follow these steps:

2. Create a new **XQuery transformation** scenario for your XQuery file.
3. Reference the JSON document that you want to transform using one of these two methods:
   - In the transformation scenario, click the **Parameters** button in the XQuery tab and add a parameter that specifies the URL to your JSON document in its value.
   - Specify the URL to your JSON document in the XQuery file you created.

4. Run the transformation.

---

**Tip:**

There is a sample XQuery file in the `[OXYGEN_INSTALL_DIR]/samples/json/transform` folder that can be used to transform a JSON document.
Querying JSON Documents with XPath or XQuery

Oxygen XML Editor provides an XPath toolbar that makes it easy to quickly query JSON documents using XPath expressions. You can also use the dedicated XPath/XQuery Builder view that allows you to compose more complex XPath or XQuery expressions and execute them over JSON documents in Text or Author mode.

XPath Toolbar

When an XPath expression is run over a JSON document, the document is converted to XML and the XPath is executed over the converted XML document. For more information about this toolbar, see XPath Toolbar (on page 2000).

Figure 405. XPath Toolbar for JSON

XPath/XQuery Builder View

You can also use the XPath/XQuery view to run XPath and XQuery expressions over a JSON document. For XQuery, you need to reference the JSON document in your XQuery content. For more information about this view, see XPath Builder View (on page 2002).
Details About Querying JSON Documents Using XPath Expressions

To execute XPath expressions over a JSON document, the document is converted to XML and the XPath is executed over the converted XML document. For this conversion, Oxygen XML Editor uses the built-in JSON to XML Converter tool. The results are mapped back to the original JSON document.

For example, if you have the following JSON document:

```json
{
    "personnel": {
        "person": [
            {"name": "Boss"},
            {"name": "Worker"}
        ],
        "id": "personnel-id"
    }
}
```

and you want to match the name of the second person, the XPath expression would look like this:

```
/JSON/personnel/person[2]/name
```

The reason why the first element is JSON is because if the JSON document contains more than one property on the first level, the converted XML document will have an additional root element called <JSON>. For more information, see JSON to XML Conversion Details (on page 1132).

The [2] in the expression represents the index of the person in the array and in this case, it matches the second person because the index counting starts with 1.

Editing JSON Schema Documents

Oxygen XML Editor offers powerful tools that allow you to design, develop, and edit JSON Schemas. These tools include:
Text editing mode (on page 1147) (packed full of editing helpers)

Schema Design mode (on page 1149) (a visual, intuitive schema diagram editor)

Author editing mode (on page 1148) (a visual authoring mode with form controls)

Grid mode (on page 1148) (a compact layout of nested tables)

JSON Schema documentation tool (on page 1167) for producing high quality output

JSON Schema instance generator (on page 1165) for producing JSON Schema documents from a JSON file

XSD to JSON Schema converter tool (on page 1172) for producing a JSON Schema from an XML schema

This section describes the features included in Oxygen XML Editor for editing JSON Schema documents and how to use them.

Resources

For more information about the JSON support in Oxygen XML Editor, see the following resources:

- Video: Introducing JSON Schema Design Mode.
- Video: JSON Editing.
- Video: JSON Tools in Oxygen

JSON Schema Editor

Oxygen XML Editor includes a specialized JSON Schema editor with various editing features for files that have the .json file extension, or for files that have .jschema file extension and includes a meta-schema URL (on page 1176) in the "$schema" key. The purpose of the JSON schema is to define the legal properties and values of a JSON document to keep it valid and well-formed.

New Document Templates

Oxygen XML Editor includes a new document template to help you get started creating a JSON Schema document. The template is called JSON Schema and it can be found in the New Document folder in the New Document wizard (on page 368). You can also customize your own JSON Schema templates (on page 377) and specify other versions (draft 04, 06, or 07).

Tip:

You can experiment with a sample of a JSON Schema file available at: [OXYGEN-INSTALL-DIR]/samples/json/personal-schema.json.

Text Mode Editor

When editing JSON Schema documents in Text editing mode, the usual text editing actions (on page 521) are available, along with various other actions, including:
• JSON Outline View (on page 1128)
• JSON-specific Syntax Highlighting (on page 1177)
• Search and Find/Replace (on page 422)
• Drag and Drop (on page 529)
• Validation (on page 1113)
• Format and Indent (Pretty Print) (on page 555)

Grid Mode Editor

Oxygen XML Editor allows you to view and edit the JSON Schema documents in the **Grid mode (on page 354)**. The JSON Schema is represented in **Grid mode** as a compound layout of nested tables and the JSON data and structure can be easily manipulated with table-specific operations or drag and drop operations on the grid components. For more details, see **JSON Grid Mode Editor (on page 1111)**.

Author Visual Editor

It is also possible to edit JSON Schema files in the visual **Author** editing mode. Oxygen XML Editor provides a JSON Schema framework that uses a specific CSS for rendering JSON Schema documents in **Author mode** and you have access to the various features and actions that are available when editing XML documents in **Author mode (on page 588)**.

Figure 407. JSON Schema in Author Mode
Design Mode Editor

Oxygen XML Editor provides a powerful, expressive visual schema diagram editor (Design mode) for editing JSON Schemas. It is helpful for both content authors who want to visualize or understand a schema and schema designers who develop complex schemas. For all the details, see JSON Schema Design Mode (on page 1149).

JSON Schema Design Mode (JSON Schema Diagram Editor)

Oxygen XML Editor provides a powerful, expressive visual schema diagram editor (Design mode) for editing JSON Schemas. The structure of the diagram editor is designed to be intuitive and easy to use. The Design mode was created to help both content authors who want to visualize or understand a schema and schema designers who develop complex schemas.

The JSON Schema Design mode includes various navigation features (on page 1150), contextual menu actions (on page 1152), automatic validation (on page 1164), and you can move and edit components directly within the diagram (on page 1152).

To switch to the JSON Schema diagram editing mode, select Design at the bottom of the editing area.

The diagram font can be increased using the usual Oxygen XML Editor shortcuts: (Ctrl + "+" (Meta + "+" on macOS)), (Ctrl + "-" (Meta + "-" on macOS)), (Ctrl + 0 (Meta + 0 on macOS)) or (Ctrl + mouse wheel (Meta...
The whole diagram can also be zoomed with one of the predefined factors available in the Schema preferences panel (on page 198). The same zoom factor is applied for the print and save actions.

Resources

For more information about the JSON Schema Design mode, watch our video demonstration:

https://www.youtube.com/embed/IEzaPRPHmGc

Also, watch our webinar to see a more in-depth look at Design mode and other features for editing JSON Schema documents: Webinar: The New JSON Schema Diagram Editor.

Navigation in the JSON Schema Design Mode

The following editing and navigation features work for all types of schema components in the JSON Schema Design mode:

- Select consecutive components on the diagram (components from the same level) using the **Shift** key. You can also make discontinuous selections in the schema diagram using the **Ctrl** (Meta on macOS) key. To deselect one of the components, use **Ctrl + Single-Click** (Command + Single-Click on macOS).
- Use the arrow keys to navigate the diagram vertically and horizontally.
- To expand a component, click the **widget** to the right of the component.
- Use **Home/End** keys to jump to the first/last component from the same level. Use **Ctrl + Home** (Command + Home on macOS) key combination to go to the diagram root and **Ctrl + End** (Command + End on macOS) to go to the last child of the selected component.
- You can easily go back to a previously visited component while moving from left to right. The path will be preserved only if you use the left arrow key or right arrow key. For example, if the current selection is on the second property from a properties parent and you press the left arrow key to jump to the properties parent, when you press the right arrow key, then the selection will be moved to the second property.
- Go back and forward between components viewed or edited in the diagram by selecting them in the Outline view (on page 993):
  - **Back** (go to previous schema component).
  - **Forward** (go to next schema component).
  - **Go to Last Modification** (go to last modified schema component).
- Go to the definition of a property by clicking on the Go to Definition widget ( ). You can then use the **Back** or **Forward** toolbar buttons to go back and forth between the properties.
- Search in the diagram using the Find/Replace dialog box (on page 432) or the Quick find toolbar (on page 445). You can find components only in the current file scope.
JSON Schema Palette View (Available in Design Mode)

The Palette view is designed to offer quick access to JSON schema components and to improve the usability of the JSON schema diagram builder. You can use the Palette to drag and drop components into the Design mode. If the view is not displayed, it can be opened by selecting it from the Window > Show View menu.

Figure 409. JSON Palette View

Components are organized functionally into 4 collapsible categories:

- **Groups**: properties, pattern properties, definitions, items, items array.
- **Components**: property, pattern property, definition, additional properties, additional items.
- **Schema composition**: anyOf, allOf, oneOf, not.
- **Conditionals**: if, then, else.

**Note:**

Two of the palette components presented above (items array and pattern property), have no match in JSON schema keywords. They were introduced only as a design convenience and their use does not generate code that would affect the validity of the edited schema.

To add a component to the edited schema:

- Click and hold a graphic symbol from the Palette view, then drag the component into the Design view.
- A line dynamically connects the component with the XML schema structure.
- Release the component into a valid position.
Note:
You cannot drop a component into an invalid position. When you hover the component into an invalid position, the mouse cursor changes its shape into 🗝️. Also, the connector line changes its color from the usual dark gray to the color defined in the Validation error highlight color option (on page 226) (default color is red).

For more information about the Schema palette, watch our video demonstration:

https://www.youtube.com/embed/r5SLk3XLOUs

Editing Actions in JSON Schema Design Mode

The JSON Schema Design mode includes various editing features, including:

- You can edit a JSON Schema using various contextual menu actions (on page 1152).
- You can copy/paste or drag/drop to move existing components to other locations in an JSON Schema. You can also use the Move Up/Down actions to change the order of the components in a parent.
- You can edit schema components directly in the diagram. For these components, you can edit the name and the additional properties presented in the diagram by double-clicking the value you want to edit. If you want to edit the name of a selected component, you can also press Enter. The list of properties that can be displayed for each component can be customized in the JSON Schema Properties preferences page (on page 199).
- When switching between editing modes, the cursor position is synchronized. For example, if you switch from Design to Text mode, the cursor will be in the same position within the document in Text mode as it was in Design mode, and vice versa.
- The content completion assistant can be used for in-place editing within the JSON schema Design mode. For example, when editing properties, you can use Ctrl+Space to invoke the content completion window and the proposals are offered based on the defined JSON schema, according to the version used.

Contextual Menu Actions in the JSON Schema Design Mode

The contextual menu of the Design mode includes the following actions:

- **Go to Definition [Ctrl + Shift + Enter]**
  Navigates to the referenced schema component. This action is also available by clicking the arrow displayed in its bottom right corner.

- **Edit Properties**
  Allows you to edit the properties of the selected component in a in-place editor.
Properties that have set values are rendered in bold while unset properties are rendered with a gray foreground. You can edit any property (set or unset) by double-clicking or by pressing Ctrl + Enter (Command + Enter on macOS).

You can delete any property already set by pressing Delete. This operation does not mean the selected property is deleted from the table. It means the property is unset (rendered with gray foreground). The Edit and Remove actions are also available on the contextual menu in the table.

If the type property changes as a result of an editing/removal action, then the list of properties presented in the table is updated according to the new schema type.

**Note:**
When filling in string values they should not be enclosed in quotation marks, these are added automatically.

**Note:**
For array values simply fill in the items that will constitute the array, separated by commas.

**Edit Annotations**

Allows you to edit the annotations for the selected schema component in the Edit Annotations dialog box. Annotations are not required, but they are encouraged as a good practice and can make the schema "self-documenting".

The title and description must be strings. A title is preferably short, whereas a description provides a more lengthy explanation about the data described by the schema. The default keyword specifies a default schema value that can be anything. The examples keyword is meant to provide an array of examples that validate against the schema. Its items must be separated by a comma.

Annotations that have set values are rendered in bold while unset annotations are rendered with a gray foreground. You can unset an annotation by using the Delete key or the Remove action that is available in the contextual menu of the table.

By default, annotations are rendered under the graphical representation of the component. To edit the annotations, use the Edit Annotations action from the contextual menu or simply double-click the annotations area (if any).

**Append child**

Offers a list of valid components, depending on the context, and appends your selection as a child of the currently selected component. You can set a name for a named component after it has been added in the diagram.

**Insert before**
Offers a list of valid components, depending on the context, and inserts your selection before the
selected component, as a sibling. You can set a name for a named component after it has been
added in the diagram.

**Insert after**

Offers a list of valid components, depending on the context, and inserts your selection after the
selected component, as a sibling. You can set a name for a named component after it has been
added in the diagram.

**Cut** [Ctrl + X (Command + X on macOS)]

Cuts the selected component(s).

**Copy** [Ctrl + C (Command + C on macOS)]

Copies the selected component(s) to the clipboard.

**Paste** [Ctrl + V (Command + V on macOS)]

Pastes the component(s) from the clipboard as children of the selected component.

**Remove Delete**

Removes the selected component(s).

**Make Required**

Marks the selected property as being required in the parent object. By default, the defined
properties are not required in the JSON schema. You can set a list of required properties in
the `required` keyword. By invoking the action, the name of the property is added in the parent
object's `required` keyword.

**Make Optional**

Marks the selected property as being optional in the parent object. By default, the defined
properties are optional in the JSON schema. You can set a list of required properties in the
`required` keyword. By invoking the action, the name of the property is deleted from the parent
object's `required` keyword.

**Refactoring > Extract definition**

Allows you to extract a definition as a global definition and reference it. It can be used on a
property to extract its definition (in case you want to reuse it) or on a local definition to extract it
as global one. Note that this action is not available for global definitions.

**Refactoring > Convert property to boolean 'any' type**

Converts the selected property's `type` into an `any` type with the value `true` or `false`. You can set
`true` value to represent a schema that matches anything, or `false` for a schema that matches
nothing.
Converts the selected definition's type into an any type definition with the value true or false. Setting the value to true represents a schema that matches anything, while false is for a schema that matches nothing.

Refactoring > Convert conditional to boolean 'any' type

Converts the selected conditional's type into an any type with the value true or false. Setting the value to true represents a schema that matches anything, while false is for a schema that matches nothing.

Refactoring > Convert property to standard type

Converts the selected property's any type into a standard type property.

Refactoring > Convert definition to standard type

Converts the selected definition's any type into a standard type property.

Refactoring > Convert conditional to standard type

Converts the selected conditional's any type into a standard type conditional.

Move Up [Alt + UpArrow]

Moves a component up in its parent

Move Down [Alt + DownArrow]

Moves a component down in its parent

Expand All

Recursively expands all sub-components of the selected component.

Collapse All

Recursively collapses all sub-components of the selected component.

Print Selection

Prints the selected component diagram.

Save as Image

Saves the selected component diagram as image, in JPEG, BMP, SVG or PNG format.

Options

Opens the JSON Schema preferences page (on page 199) where you can control which properties to display for JSON Schema components in the JSON Schema Design mode.

JSON Schema Design Mode Components and Properties

A schema diagram contains a series of interconnected components. In the JSON Schema Design mode, each component is displayed with distinguishable graphics and the thickness of the lines that connect the components identify whether the connected component is required or optional (a thick line denotes a required component while a thin line denotes an optional component.)
This section provide details about the available components and their corresponding graphics, as well as details about component properties.

**JSON Schema Components**

**Schema**

**Description:** Defines the root element of a JSON schema. A JSON schema document contains all the steps that are necessary to be performed to validate JSON documents and it contains a collection of JSON schema components.

**Properties**

**Description:** An object is valid against the properties keyword if every property that is present in both the object and the value of this keyword validates against the corresponding schema. The value must be an
object, where properties must contain valid JSON schemas (object or boolean). Only the property names that are present in both the object and the keyword value are checked.

**Note:**
Properties with a boolean value are presented in diagram as components. You can change the boolean value by modifying the `any type` property value. You can also convert a boolean `any type` property into a standard `type` property using the **Convert property to standard type** contextual menu action.

### Definitions

**Figure 414. The definitions Component**

**Description:** The optional definitions keyword does not directly validate data, but it contains a map of validation schemas. The value can be anything.

**Note:**
Definitions with a boolean value are presented in diagram as components. You can change the boolean value by modifying the definition's `any type` property value. You can also convert a definition's boolean `any type` property into a standard `type` property using the **Convert definition to standard type** contextual menu action.

### Pattern Properties

**Figure 415. The patternProperties Component**

**Description:** An object is valid against the patternProperties keyword if every property where a property name (key) matches a regular expression from the value of this keyword is also valid against the corresponding schema. The value must be an object where the keys must be valid regular expressions and the corresponding values must be valid JSON schemas (object or boolean).

### Items

**Figure 416. The items Component**

**Description:** An array is valid against the items keyword if the items are valid against the corresponding schemas provided by the keyword value. The value can be:
• A valid JSON schema (object or boolean). Every item must be valid against this schema.
• An array of valid JSON schemas. Each item must be valid against the schema defined at the same position (index). Items that do not have a corresponding position (e.g. an array contains 5 items but this keyword only has 3) will be considered valid unless the **additionalItems** keyword is present, which will decide the validity.

**Additional Items**

*Figure 417. The additionalItems Component*

![Additional Items Component](image)

**Description:** An array is valid against the **additionalItems** keyword if all *unchecked* items are valid against the schema defined by the keyword value. An item is considered *unchecked* if the **items** keyword or **prefixItems** keyword (starting with draft 2020-12) contains an array of schemas and does not have a corresponding position (index). The value must be a valid JSON schema (object or boolean).

**Property**

*Figure 418. Example of a property Component*

![Property Component](image)

**Description:** Each **key**:**value** pair is known as a **property** of the object. The value can be any JSON data type.

**Definition**

*Figure 419. Example of a definition Component*

![Definition Component](image)

**Description:** The definition of a component of a schema. It can be referenced from a property to define its specification.

**Pattern Property**

*Figure 420. Example of a Pattern property Component*

![Pattern Property Component](image)
**Description:** Maps regular expressions to schemas. If a property name matches the given regular expression, the property value must validate against the corresponding schema.

### Additional Properties

**Figure 421. The additionalProperties Component**

<table>
<thead>
<tr>
<th>additionalProperties</th>
</tr>
</thead>
<tbody>
<tr>
<td>type integer</td>
</tr>
</tbody>
</table>

**Description:**

An object is valid against the `additionalProperties` keyword if all `unchecked` properties are valid against the schema defined by the value of this keyword. **Unchecked** properties are the properties not checked by the `properties` and `patternProperties` keywords (if a property name is not present in the `properties` keyword and does not match any regular expression defined by the `patternProperties` keyword, then it is considered `unchecked`). The value must be a valid JSON schema (object or boolean).

To be more concise, if there are `unchecked` properties:

- If the value of the `additionalProperties` keyword is `true`, it is always valid.
- If the value is `false`, it is never valid.
- If the value contains an object (schema), every property must be valid against that schema.

### AllOf

**Figure 422. Example of an allOf Component**

<table>
<thead>
<tr>
<th>allOf / 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>type object</td>
</tr>
</tbody>
</table>

**allOf / 1 |
| type object |

**Description:** An instance is valid against the `allOf` keyword if it is valid against all schemas defined by the value of this keyword. The value of this keyword must be an array of valid JSON schemas (objects or boolean).

### AnyOf

**Figure 423. Example of an anyOf Component**

<table>
<thead>
<tr>
<th>anyOf / 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>type object</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>anyOf / 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>type object</td>
</tr>
</tbody>
</table>
Description: An instance is valid against the **anyOf** keyword if it is valid against at least one schema defined by the value of this keyword. The value must be an array of valid JSON schemas (objects or boolean).

**OneOf**

![Figure 424. Example of an oneOf Component](image)

Description: An instance is valid against the **oneOf** keyword if it is valid against exactly one schema defined by the value of this keyword. The value must be an array of valid JSON schemas (object or boolean).

**Not**

![Figure 425. Example of a not Component](image)

Description: An instance is valid against the **not** keyword if it is not valid against the schema defined by the value of this keyword. The value must be a valid JSON schema (object or boolean).

**If/Then/Else**

![Figure 426. Examples of an if, then, and else Components](image)

Description: A conditional structure that contains three keywords: **if**, **then**, and **else**. Every keyword value must be a valid JSON schema (object or boolean). When the **if** keyword is not present, the **then** and **else** keywords are ignored. When the **if** keyword is present, at least one of the **then** or **else** keywords should also be present (or both). The instance is valid against this keyword in one of the following cases:

- The **if** keyword validates the instance and the **then** keyword also validates it.
- The **if** keyword does not validate the instance but the **else** keyword validates it.

Related information

* JSON Schema Component Properties (on page 1161)
# JSON Schema Component Properties

## Table 37. JSON Schema Diagram Component Properties

<table>
<thead>
<tr>
<th>Group</th>
<th>Property Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Common</td>
<td>type</td>
<td>Specifies the type of data that the schema is expecting to validate. This keyword is not mandatory and the value must be a string representing a valid data type, or an array of strings representing a valid list of data types. When specifying multiple types, their order is irrelevant to the validation process, but make sure that a data type is specified only once.</td>
</tr>
<tr>
<td>$ref</td>
<td></td>
<td>An instance is valid against this keyword if it is valid against the schema that points to the location indicated in its value. The value must be a string representing a URI, URI reference, URI template, or JSON pointer.</td>
</tr>
<tr>
<td>$id</td>
<td></td>
<td>Specifies a unique ID for a document or a document sub-schema. The value must be a string representing a URI. All sub-schema IDs are resolved relative to the document ID. It is not a required keyword, but it is considered best practice to use it.</td>
</tr>
<tr>
<td>enum</td>
<td></td>
<td>An instance validates against this keyword if its value can be found in the items defined by its value. The value must be an array that contains anything (an empty array is not allowed).</td>
</tr>
<tr>
<td>const</td>
<td></td>
<td>An instance validates against this keyword if its value equals the value of this keyword. The value can be anything.</td>
</tr>
<tr>
<td>$comment</td>
<td></td>
<td>Contains an observation about the schema. The value must be a string.</td>
</tr>
<tr>
<td>readOnly</td>
<td></td>
<td>Used to mark specific properties as <code>read-only</code>.</td>
</tr>
<tr>
<td>writeOnly</td>
<td></td>
<td>Used to mark specific properties as <code>write-only</code>.</td>
</tr>
<tr>
<td>Object</td>
<td>additional-Properties</td>
<td>An object is valid against this keyword if all <code>unchecked</code> properties are valid against the schema defined by its value. <code>Unchecked</code> properties are the properties not checked by the <code>properties</code> and <code>patternProperties</code> keywords (if a property name is not present in the <code>properties</code> keyword and does not match any regular expression defined by the <code>patternProperties</code> keyword, then it is considered <code>unchecked</code>). The value must be a valid JSON schema (object or boolean).</td>
</tr>
</tbody>
</table>

To be more concise, if we have `unchecked` properties:
### Dependencies
An object is valid against this keyword if it meets all dependencies specified by its value.

### Max Properties
An object is valid against this keyword if the number of properties it contains is lower than or equal to its value. The value must be a non-negative integer. Using 0 as a value means that the object must be empty (no properties).

### Min Properties
An object is valid against this keyword if the number of properties it contains is greater than or equal to its value. The value of this keyword must be a non-negative integer. Using 0 as a value has no effect.

### Pattern Properties
An object is valid against this keyword if every property where a property name (key) matches a regular expression from its value and is also valid against the corresponding schema. The value must be an object where the keys must be valid regular expressions and the corresponding values must be valid JSON schemas (object or boolean).

### Property Names
An object is valid against this keyword if every property name (key) is valid against its value. The value must be a valid JSON schema (an object or a boolean).

### Required
An object is valid against this keyword if it contains all property names (keys) specified by its value. The value must be a non-empty array of strings that represent property names.

### Additional Items
An array is valid against this keyword if all unchecked items are valid against the schema defined by its value.

### Contains
An array is valid against this keyword if at least one item is valid against the schema defined by its value. The value must be a valid JSON schema (object or boolean).

### Items
An array is valid against this keyword if the items are valid against the corresponding schemas provided by its value. The value of this keyword can be:
Table 37. JSON Schema Diagram Component Properties (continued)

<table>
<thead>
<tr>
<th>Properties Group</th>
<th>Property Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-</td>
<td>maxItems</td>
<td>An array is valid against this keyword if the number of items it contains is lower than or equal to its value. The value must be a non-negative integer.</td>
</tr>
<tr>
<td>-</td>
<td>minItems</td>
<td>An array is valid against this keyword if the number of items it contains is greater than or equal to its value. The value must be a non-negative integer.</td>
</tr>
<tr>
<td>-</td>
<td>uniqueItems</td>
<td>An array is valid against this keyword if an item cannot be found more than once in the array. The value must be boolean. If set to false, the keyword validation will be ignored.</td>
</tr>
<tr>
<td>Number/Integer Properties</td>
<td>exclusiveMaximum</td>
<td>A number is valid against this keyword if it is strictly lower than its value. The value must be a number (integer or float) or boolean.</td>
</tr>
<tr>
<td>-</td>
<td>exclusiveMinimum</td>
<td>A number is valid against this keyword if it is strictly greater than its value. The value must be a number (integer or float) or boolean.</td>
</tr>
<tr>
<td>-</td>
<td>maximum</td>
<td>A number is valid against this keyword if it is lower than or equal to its value. The value must be a number (integer or float).</td>
</tr>
<tr>
<td>-</td>
<td>minimum</td>
<td>A number is valid against this keyword if it is greater than or equal to its value. The value must be a number (integer or float).</td>
</tr>
<tr>
<td>-</td>
<td>multipleOf</td>
<td>A number is valid against this keyword if the division between the number and its value results in an integer. The value must be a strictly positive number (zero is not allowed).</td>
</tr>
<tr>
<td>String Properties</td>
<td>contentEncoding</td>
<td>A string is valid against this keyword if it is encoded using the method indicated by its value. The value must be a string.</td>
</tr>
</tbody>
</table>
### JSON Schema Diagram Component Properties (continued)

<table>
<thead>
<tr>
<th>Properties Group</th>
<th>Property Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>content-MediaType</td>
<td>A string is valid against this keyword if its content has the media type (MIME type) indicated by its value. If the <strong>contentEncoding</strong> keyword is also specified, the decoded content must have the indicated media type. The value must be a string.</td>
<td></td>
</tr>
<tr>
<td>format</td>
<td>Performs a semantic validation on data. The value must be a string that represents a format. The keyword behavior depends on the data type, meaning that the same <strong>format</strong> name for a string behaves differently on a number, or is missing, because not all data types must implement a format and usually differing data types have different formats.</td>
<td></td>
</tr>
<tr>
<td>maxLength</td>
<td>A string is valid against this keyword if its length is lower than or equal to its value. The value must be a non-negative integer.</td>
<td></td>
</tr>
<tr>
<td>minLength</td>
<td>A string is valid against this keyword if its length is greater than or equal to its value. The value must be a non-negative integer.</td>
<td></td>
</tr>
<tr>
<td>pattern</td>
<td>A string is valid against this keyword if it matches the regular expression specified by its value. The value must be a string that represents a valid regular expression.</td>
<td></td>
</tr>
</tbody>
</table>

### Related information

**JSON Schema Components** *(on page 1156)*

### JSON Schema Design Mode Validation

Validation for the **Design** mode is seamlessly integrated in the Oxygen XML Editor **JSON Schema validation support** *(on page 1176)*. You can ensure that the JSON Schemas you develop comply with JSON standards by using the built-in validation engine. You can also configure a validation scenario to use an external JSON Schema validation engine. A validation scenario can also be configured to define a main module of a complex JSON Schema to validate modules in the context of the larger schema structure.
Visual Error Markers

A schema validation error is presented by highlighting the invalid component in the following locations:

- The component is surrounded with a red or yellow border within the diagram (you can customize these colors in the Document Validation preferences page (on page 226)).
- A tooltip contains information about the error when hovering over the error within the diagram.
- The vertical stripe at the right side of the editor displays error markers.
- In the message area at the bottom of the editor (denoted with a red icon 🟢).

If you validate the entire schema using the ✅ Validate action from the ✅ Validation toolbar drop-down menu, all validation errors will be presented in the Results pane at the bottom of the application. To resolve an error, just click it and the corresponding schema component will be displayed as the diagram root so that you can easily correct the error.

Generating JSON Schema from a JSON File

Oxygen XML Editor includes a tool for generating a sample JSON Schema from a JSON file. To generate a sample JSON Schema, select Generate JSON Schema from the Tools > JSON Tools menu. The action opens a dialog box where you can configure some options for generating the JSON Schema.
The **Generate JSON Schema** dialog box includes the following fields and options:

**JSON Document URL**

The URL of the JSON file. You can specify the path by using the text field, the history drop-down menu, or the browsing actions in the **Browse** drop-down list.

**Output JSON Schema**

The path to the folder where the generated JSON Schema will be saved.

**Open in Editor**

If selected, the generated JSON Schema is opened in the editor.

**JSON Schema version**

The version of the resulting JSON schema. The possible choices are: **Draft 4**, **Draft 6**, **Draft 7**, and **Draft 2019-09**.

**Extract matching format for strings**

If selected, the generator will attempt to find a format that matches the string values from the JSON Document.

**Restrict additional content**

If selected, **additionalProperties** (for objects) and **additionalItems** (for arrays) will be set to **false** in the resulting schema. By default, these keys are not in the schema, meaning that providing additional content (according to the schema) is allowed.

**Add default values for simple types**

If selected, the **default** values (0 for number, "" for string, **false** for boolean) and **examples** for strings will be added.
Make all properties required

If selected, the generator will mark all the properties as required in the resulting schema.

You can click **Generate** at any point to generate the JSON Schema.

### Generating JSON Schema Documentation

Oxygen XML Editor includes a tool for generating documentation for a JSON Schema file in HTML format. To generate JSON Schema documentation, select **JSON Schema Documentation** from the **Tools > Generate Documentation** menu. You can also open the tool by using the **Generate Documentation** toolbar button. This opens a dialog box where you can specify the location of the JSON Schema file and HTML output file.

This tool requires an additional add-on to be installed, so the first time you invoke the action, Oxygen XML Editor presents a dialog box asking if you want to install it. Once installed, you need to restart Oxygen XML Editor and the **JSON Schema Documentation** action will invoke the tool.

### Quick Installation

You can drag the following **Install** button and drop it into the main editor in **Oxygen** (version 24.1 or newer) to quickly initiate the installation process:

![Install Button](Install)

### Manual Installation

To manually install it the add-on, follow these instructions:

1. Go to **Help > Install new add-ons** to open an add-on selection dialog box.
2. Enter or paste `https://www.oxygenxml.com/InstData/Addons/default/updateSite.xml` in the **Show add-ons from** field or select it from the drop-down menu.
3. Select the **JSON Schema Documentation** add-on and click **Next**.
4. Read the end-user license agreement. Then select the **I accept all terms of the end-user license agreement** option and click **Finish**.
5. Restart the application.

**Result:** The **JSON Schema Documentation** dialog box is now available and can be selected from the **Tools > Generate Documentation** menu.
The JSON Schema Documentation dialog box includes the following fields and options:

**JSON Schema URL**

The URL of the JSON Schema file. You can specify the path by using the text field, the history drop-down menu, or the browsing actions in the Browse drop-down list.

**Output file**

The path to the folder where the generated HTML file will be saved.

**Split output into multiple files**

If selected, the application splits the output into multiple files. You can choose between splitting them by component name or location.

**Open in Browser/System Application**

If selected, the generated result is opened in the system application associated with the output file type (HTML).

**Included component details**

This section can be used to specify whether or not the following components are shown in the generated documentation:

- **Annotations** - Displays the annotations (title, description) for each component (property or definition).
- **Constraints** - Displays the schema constraints for each component, according to its type.
- **Properties** - Displays the `properties` of an Object Schema.
- **Pattern Properties** - Displays the `patternProperties` of an Object Schema.
- **Enumerations** - Displays the enumerated values, if specified in the schema.
- **Source** - Displays the text schema source for each component.
- **Used By** - Displays the list of all the components that reference the current definition.
- **Composition** - Displays the `oneOf`, `anyOf`, and `allOf` compositors that are used for combining schemas.
- **Diagram** - Displays the diagram image for each component. The diagrams are generated according to the options specified in the **Schema Design preferences page (on page 198)**. Diagrams are also generated for components within schemas from referenced files.
- **Image Map** - Diagrams will include hyperlinks that navigate to each particular component.
- **Location** - Displays the schema location for each component.

You can click **Generate** at any point to generate the JSON Schema documentation.

**Generated JSON Schema Documentation in HTML Format**

After generating the JSON Schema documentation, it is presented in a visual diagram style with various sections, hyperlinks, and options.
The generated documentation includes a Table of Contents on the left pane with links to particular sections in the right pane. You can collapse or expand details by using the Showing options or the Collapse or Expand buttons.

**Generating Sample JSON Files from a JSON Schema**

Oxygen XML Editor includes a tool for generating sample JSON files. To generate sample JSON files from a JSON Schema, select **Generate Sample JSON Files** from the Tools > JSON Tools menu. The action opens a dialog box where you can configure a variety of options for generating the files.
The Generate Sample JSON Files dialog box includes the following fields and options:

**Schema URL**

The URL of the Schema location. You can specify the path by using the text field, the history drop-down menu, or the browsing actions in the Browse drop-down list.

**Associate schema in the document**

If enabled, the specified schema will be associated with the generated files.

**Output folder**

Path to the folder where the generated JSON instances will be saved.

**File name**

The name of the instance(s) that will be generated. By default, instance.json is used.

**Number of instances**

The desired number of JSON instances to be generated. When more than one instance is generated, the index of the instance will be added to its file name.

**Property value**

You can specify the way the values of the properties are generated. The following options are available:
• None - Assigns empty values for properties (a template file will be generated). This is the default value.
• Default - Assigns the name of the property as the value (for strings) or assigns the specified minimum value (for numbers).
• Random - Assigns random values according to schema restrictions.

Generate optional properties
If selected, the JSON instance will be generated with optional properties that are defined in the JSON schema. Otherwise, only the required properties will be generated.

Generate additional content
If selected, the JSON instance will be generated with additional properties that are defined in the JSON schema as additionalProperties and additional items that are defined as additionalItems (in the case of an Array).

Choice strategy
You can specify the way an instance will be generated from a schema that contains a CombinedSchema (with either oneOf or anyOf). The following options are available:

• First - The first defined schema in oneOf or anyOf will be used.
• Random - A random schema defined in oneOf or anyOf will be used.

Recursion level
This option controls the maximum allowed depth (must be a number), in case the selected schema contains recursive calls of $ref schemas referencing one another. By default, it is set to 1, meaning that the generation for the recursive calls will stop after the first iteration.

Open first instance in editor
If selected, the first generated instance is opened in the editor.

You can click OK at any point to generate the sample JSON files.

XSD to JSON Schema Converter
Oxygen XML Editor includes a tool for converting an XML Schema file (XSD) to a JSON Schema file. The XSD to JSON Schema action for invoking the tool can be found in the Tools > JSON Tools menu. It requires an additional add-on to be installed, so the first time you invoke the action, Oxygen XML Editor will present a dialog box asking if you want to install it. Once installed, you need to restart Oxygen XML Editor and the XSD to JSON Schema action will invoke the tool.

Quick Installation
You can drag the following Install button and drop it into the main editor in Oxygen (version 24.1 or newer) to quickly initiate the installation process:
Manual Installation

To manually install the add-on, follow these instructions:

1. Go to Help > Install new add-ons to open an add-on selection dialog box.
2. Enter or paste https://www.oxygenxml.com/InstData/Addons/default/updateSite.xml in the Show add-ons from field or select it from the drop-down menu.
3. Select the XSD to JSON Schema add-on and click Next.
4. Read the end-user license agreement. Then select the I accept all terms of the end-user license agreement option and click Finish.
5. Restart the application.

Result: The XSD to JSON Schema dialog box is now available and can be selected from the Tools > JSON Tools menu.

Converting XSD to JSON Schema

To convert an XML Schema (XSD) to a JSON Schema, follow these steps:

1. Select the XSD to JSON Schema action from the Tools > JSON Tools menu.

   Step Result: The XSD to JSON Schema dialog box is displayed:

   ![XSD to JSON Schema Dialog Box](image)

2. In the XSD URL field, choose or enter the URL of the XML Schema document. The conversion supports XSD versions 1.0 and 1.1.
3. In the Output file field, choose the path for the resulting output file.
4. [Optional] You can select the Open in Editor option to open the resulting JSON Schema document in the main editing pane.
5. For the JSON Schema version option, choose the version of the resulting JSON schema. The possible choices are: Draft 4, Draft 6, Draft 7, and Draft 2019-09.
6. [Optional] If you select the **Restrict additional content** option, then `additionalProperties` (for objects) and `additionalItems` (for arrays) will be set to `false` in the resulting schema. By default, these keys are not in the schema, meaning that providing additional content (according to the schema) is allowed.

7. [Optional] You can select the **Preserve case of names from the XSD** option if you want the names from the XSD to remain unchanged in the resulting JSON Schema. Otherwise, the default JAXB naming algorithm will be applied (for example, "some.nAMe" is changed to "SomeNAMe", or "Some_oth3r_name" is changed to "SomeOth3RName").

8. Click the **Convert** button.

**Result:** The original XSD document is now converted to a JSON Schema document. The resulting JSON Schema will be the specified draft and will contain:

- The `$id` of the schema, generated from XSD `targetNamespace`.
- The `$definitions` section, which declares complex and `enum` types.
- The `anyOf` section, which lists possible top-level elements as an array of objects.

**Other Possible Results:**

- If an XSD type extends another type, then its schema is combined with the schema of the base type using the `allOf` keyword.
- If an extension in XSD defines an element with the same name as an attribute in the base, a property named `rest` is generated to avoid name conflicts in JSON.
- If a property of a complex type is a collection property, the schema of the collection items will be wrapped in the JSON array schema.

**Conversion Mappings**

The following table lists the specific conversion mapping details.

<table>
<thead>
<tr>
<th>XML Schema Type</th>
<th>JSON Schema Representation</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>anySimpleType</code></td>
<td>string, number, integer, boolean, null</td>
</tr>
<tr>
<td><code>anyType</code></td>
<td>string, number, integer, boolean, null, object, array</td>
</tr>
<tr>
<td><code>string</code></td>
<td>string</td>
</tr>
<tr>
<td><code>normalizedString</code></td>
<td>string</td>
</tr>
<tr>
<td><code>token</code></td>
<td>string</td>
</tr>
<tr>
<td><code>language</code></td>
<td>string</td>
</tr>
<tr>
<td><code>Name</code></td>
<td>string</td>
</tr>
<tr>
<td><code>NCName</code></td>
<td>string</td>
</tr>
<tr>
<td><code>ID</code></td>
<td>string</td>
</tr>
<tr>
<td><code>IDREF</code></td>
<td>string</td>
</tr>
<tr>
<td>XML Schema Type</td>
<td>JSON Schema Representation</td>
</tr>
<tr>
<td>---------------------</td>
<td>-------------------------------------</td>
</tr>
<tr>
<td>IDREFS</td>
<td>array of strings</td>
</tr>
<tr>
<td>ENTITY</td>
<td>string</td>
</tr>
<tr>
<td>ENTITIES</td>
<td>array of strings</td>
</tr>
<tr>
<td>NMTOKEN</td>
<td>string</td>
</tr>
<tr>
<td>NMTOKENS</td>
<td>array of strings</td>
</tr>
<tr>
<td>boolean</td>
<td>boolean</td>
</tr>
<tr>
<td>base64Binary</td>
<td>array of integers</td>
</tr>
<tr>
<td>hexBinary</td>
<td>array of integers</td>
</tr>
<tr>
<td>float</td>
<td>number</td>
</tr>
<tr>
<td>decimal</td>
<td>number</td>
</tr>
<tr>
<td>integer</td>
<td>integer</td>
</tr>
<tr>
<td>nonPositiveInteger</td>
<td>integer</td>
</tr>
<tr>
<td>negativeInteger</td>
<td>integer</td>
</tr>
<tr>
<td>long</td>
<td>integer</td>
</tr>
<tr>
<td>int</td>
<td>integer</td>
</tr>
<tr>
<td>short</td>
<td>integer</td>
</tr>
<tr>
<td>byte</td>
<td>integer</td>
</tr>
<tr>
<td>nonNegativeInteger</td>
<td>integer</td>
</tr>
<tr>
<td>unsignedLong</td>
<td>integer</td>
</tr>
<tr>
<td>unsignedInt</td>
<td>integer</td>
</tr>
<tr>
<td>unsignedShort</td>
<td>integer</td>
</tr>
<tr>
<td>unsignedByte</td>
<td>integer</td>
</tr>
<tr>
<td>positiveInteger</td>
<td>integer</td>
</tr>
<tr>
<td>double</td>
<td>number</td>
</tr>
<tr>
<td>anyURI</td>
<td>string with &quot;format&quot;:&quot;uri&quot;</td>
</tr>
<tr>
<td>QName</td>
<td>object with &quot;namespaceURI&quot;, &quot;localPart&quot;, &quot;prefix&quot;</td>
</tr>
<tr>
<td>duration</td>
<td>string</td>
</tr>
<tr>
<td>dateTime</td>
<td>string with &quot;format&quot;:&quot;date-time&quot;</td>
</tr>
<tr>
<td>date</td>
<td>string with &quot;format&quot;:&quot;date&quot;</td>
</tr>
</tbody>
</table>
### XML Schema Type

<table>
<thead>
<tr>
<th>XML Schema Type</th>
<th>JSON Schema Representation</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>time</code></td>
<td>string with &quot;format&quot;:&quot;time&quot;</td>
</tr>
</tbody>
</table>

### Conversion Limitations

In most cases, the conversion creates an equivalent schema, but there are some limitations:

- Restrictions/facets are not taken into consideration when converting (fractionDigits, pattern, totalDigits, whiteSpace, minInclusive, maxInclusive, and the restrictions for length, except enumeration). However, extensions and indicators are properly converted (minOccurs, maxOccurs, group, sequence, choice).
- The `<documentation>` element is not converted into `<description>`.
- The `@substitutionGroup` attribute for an element that has no declared type becomes a reference to the element that can substitute it.
- The `@block` attribute is not taken into consideration during the conversion.

### Validating JSON Schema Documents

A valid JSON Schema document is a well-formed document that also conforms to the JSON meta-schema rules that defines the legal syntax of a JSON Schema document.

If a JSON document includes a meta-schema URL in the document root with the "$schema" key, the file will be validated as a JSON Schema against the specified meta-schema.

### Quick Reference

- If there is a "$schema": "http://json-schema.org/draft-04/schema" property in the schema root, then Draft 4 will be used.
- If there is a "$schema": "http://json-schema.org/draft-06/schema" property in the schema root, then Draft 6 will be used.
- If there is a "$schema": "http://json-schema.org/draft-07/schema" property in the schema root, then Draft 7 will be used.
- If there is a "$schema": "http://json-schema.org/draft/2019-09/schema" property in the schema root, then a Draft 7 compatibility mode will be used.
- If there is a "$schema": "http://json-schema.org/draft/2020-12/schema" property in the schema root, then a Draft 7 compatibility mode will be used.
- If there is a "$schema" property in the schema root, but with a different draft value, then an error will be displayed ("could not determine version").
- If none of these are found, then it is validated as a simple JSON instance.
- You could also select the JSON Schema Validator in a JSON validation scenario (on page 1118) and it will use the version specified in the JSON Schema, or if a version is not specified, the JSON Schema draft-04 will be used.
For information about how to associate a JSON Schema for the purposes of validation, see *Associating a Schema Through a Validation Scenario* (on page 1122).

For information about using a JSON Schema to validate documents, see *Validating JSON Documents Against JSON Schema or Schematron* (on page 1114).

### Syntax Highlighting in JSON Schema Documents

Oxygen XML Editor supports syntax highlighting in **Text** mode to make it easier to read the semantics of the structured content by displaying each type of code in different colors and fonts.

To customize the colors or styles used for the syntax highlighting colors for JSON Schema files, follow these steps:

1. Open the **Preferences** dialog box (**Options > Preferences**) (on page 124).
2. Go to **Editor > Syntax Highlight** (on page 224).
3. Select and expand the **JSON Schema** section in the top pane.
4. Select the component you want to change and customize the colors or styles using the selectors to the right of the pane.

You can see the effects of your changes in the **Preview** pane.

**Related Information:**

- *Syntax Highlight Preferences* (on page 224)

### Editing YAML Documents

Oxygen XML Editor includes a specialized YAML editor with various editing features for files that have the `yaml/yml` file extension. It includes the usual **text editing actions** (on page 521), **syntax highlighting** (on page 1183), **automatic validation** (on page 1178), and **formatting and indenting** (on page 1177). Oxygen XML Editor also includes a document template to help you get started with YAML documents. The template is called **YAML** and it can be found in the **New Document** folder in the **New document wizard** (on page 368).

**Tip:**

You can experiment with a sample of a YAML file available at: `[OXYGEN-INSTALL-DIR]/samples/yaml/personal.yaml`.

### Format and Indent YAML Documents

Oxygen XML Editor includes support for formatting and indenting YAML documents. You can trigger a format and indent operation for your YAML document using the **Format and Indent** toolbar button.

Some of the formatting actions that are performed include:
• Indents the document with the indent size specified in the Editor > Format preferences page (on page 202).
• Removes empty lines and extra spaces between keys and values.
• Compacts the string values (for example, description) and limits it to 80 characters on a row.

To batch format/indent multiple YAML files, select and right-click the files in the Project view, then select Edit > Format and Indent Files.

Related Information:
YAML to JSON Converter (on page 1137)
JSON to YAML Converter (on page 1137)

Validating YAML Documents

Automatic Validation

Oxygen XML Editor includes built-in validation engine for YAML documents to help you keep them well-formed. YAML documents are validated automatically as you type and the validation engine detects syntax errors (such as improper indentation or duplicate keys), based on the YAML specification. The built-in validation also works on files that consist of multiple YAML documents.

Manual Validation Actions

To manually validate the currently edited YAML document, use one of the following actions:

- **Validate** (Ctrl + Shift + V (Command + Shift + V on macOS))
  Available from the ✓ * Validation drop-down menu on the toolbar, the Document > Validate menu, or from the Validate submenu when invoking the contextual menu on one or more YAML documents in the Project view (on page 402). The validation is done based on the YAML specification.

- **Validate with**
  Available from the ✓ * Validation drop-down menu on the toolbar or the Document > Validate menu. This action opens a dialog box that allows you to specify a JSON Schema for validating the current YAML document (it also works on files that consist of multiple YAML documents).

  **Note:**
  The Validate with action does not work for files loaded through a custom protocol plugin (on page 2427) developed independently and added to Oxygen XML Editor after installation.

- **Check Well-Formedness** (Ctrl + Shift + W (Command + Shift + W on macOS))
Available from the *Validation* drop-down menu on the toolbar, the Document > Validate menu, or from the Validate submenu when invoking the contextual menu in the Project view (on page 402). This action checks the document for syntax errors to make sure it is well-formed.

**Validate with Schema**

Available from the Validate submenu when invoking the contextual menu on one or more YAML documents in the Project view (on page 402). This action opens a dialog box that allows you to specify a JSON Schema to be used for the validation.

**Batch Validation**

The built-in validation engine can also be used to batch validation multiple YAML files at once by selecting multiple files in the Project view, right-click, and select Validate > Check Well-Formedness or Validate > Validate. This automatically validates the selected files using the built-in YAML validation engine, based on the YAML specification.

**Creating a YAML Validation Scenario**

The built-in YAML Validator engine that can be specified in a validation scenario to validate YAML documents. In this case, the validation is done against a specified JSON Schema.

**Creating a YAML Validation Scenario**

To create a validation scenario, follow these steps:

1. Select the 🔄 Configure Validation Scenario(s) action in one of the following ways:
   - From the *Validation* toolbar drop-down menu.
   - From the Document > Validate menu.
   - From the Validate submenu, when invoking the contextual menu on a file in the Project view (on page 402).

   **Step Result:** The Configure Validation Scenario(s) dialog box is displayed.

2. Click the New button.

   **Step Result:** A validation scenario configuration dialog box is displayed.
This scenario configuration dialog box allows you to configure the following information and options:

**Name**

The name of the validation scenario.

**Storage**

You can choose between storing the scenario in the Project Options (on page 3222) or Global Options (on page 3219).

**URL of the file to validate**

The URL of the main module that includes the current module. It is also the entry module of the validation process when the current one is validated. To edit the URL, double-click its cell and specify the URL of the main module by doing one of the following:

- Enter the URL in the text field or select it from the drop-down list.
- Use the Browse drop-down button to browse for a local, remote, or archived file.
- Use the Insert Editor Variable button to insert an editor variable (on page 323) or a custom editor variable (on page 332).
Figure 434. Insert an Editor Variable

```
${Desktop} - My Desktop
${start-dir} - Start directory of custom validator
${standard-params} - List of standard params for command line
${fn} - The current file name without extension
${currentFileURL} - The path of the currently edited file (URL)
${rootURL} - The path of current file directory (URL)
${frameworks} - Oxygen frameworks directory (URL)
${pdu} - Project directory (URL)
${oxygenHome} - Oxygen installation directory (URL)
${home} - The path to user home directory (URL)
${proj} - Project name
${env(VAR_NAME)} - Value of environment variable VAR_NAME
${system(var.name)} - Value of system variable var.name
```

**File type**

The type of the document that is validated in the current validation unit. Oxygen XML Editor automatically selects the file type depending on the value of the **URL of the file to validate** field.

**Validation engine**

For YAML documents, the built-in **YAML Validator** engine (**Default engine**) is used.

**Automatic validation**

If this option is selected, the validation operation defined by this row is also applied by the automatic validation feature (on page 775). If the **Automatic validation** feature is disabled in the Document Checking preferences page (on page 226), then this option is ignored, as the preference setting has a higher priority.

**Schema**

Displays the specified schema.

**Specify Schema**

Opens the **Specify Schema** dialog box that allows you to set a schema to be used for validating YAML documents.

**Move Up**

Moves the selected scenario up one spot in the list.

**Move Down**

Moves the selected scenario down one spot in the list.

**Add**

Adds a new validation unit to the list.
Remove

Removes an existing validation unit from the list.

3. Configure any of the existing validation units according to the information above. You can use the buttons at the bottom of the table to add, remove, or move validation units.

4. Click OK.

**Result:** The newly created validation scenario will now be included in the list of scenarios in the Configure Validation Scenario(s) dialog box. You can select the scenario in this dialog box to associate it with the current document and click the **Apply associated** button to run the validation scenario.

---

**Related Information:**

Assigning a JSON Schema Directly in YAML Documents *(on page 1182)*

---

**Associating a JSON Schema Directly in YAML Documents**

**Associate Schema Action**

The schema used by the validation engine can be associated with the current document by using the `Associate Schema` action. The association can specify a relative file path or a URL of the schema.

To associate a JSON Schema to the current YAML document, follow these steps:

1. Select the `Associate Schema` action from the toolbar (or **Document > Schema** menu).

**Step Result:** The **Associate Schema** dialog box is displayed:

![Figure 435. Associate Schema Dialog Box](image)

This dialog box contains the following options for YAML documents:

- **URL** - Allows you to specify or select a URL for the schema. It also keeps a history of the last used schemas. The URL must point to the schema file that can be loaded from the local disk or from a remote server through HTTP(S), FTP(S) or a custom protocol *(on page 2444)*.
- **Use path relative to file location** - Select this option if the YAML instance document and the associated schema contain relative paths. The location of the schema file is inserted in the YAML instance document as a relative file path. This practice allows you, for example, to share these documents with other users without running into problems caused by multiple project locations on physical disk.
2. Select the JSON Schema that will be associated with the YAML document.

3. Click OK.

Result: A $schema property is added at the beginning of the document with its value set to the specified URL. If the document already contained a schema association, the old association will be replaced with the new one.

Tip:
To quickly open the schema used for validating the current document, select the Open Associated Schema action from the toolbar (or Document > Schema menu).

Related Information:
Creating a YAML Validation Scenario (on page 1179)

Syntax Highlighting in YAML Documents

Oxygen XML Editor supports syntax highlighting in Text mode to make it easier to read the semantics of the structured content by displaying each type of code in different colors and fonts.

To customize the colors or styles used for the syntax highlighting colors for YAML files, follow these steps:

1. Open the Preferences dialog box (Options > Preferences) (on page 124).
2. Go to Editor > Syntax Highlight (on page 224).
3. Select and expand the YAML section in the top pane.
4. Select the component you want to change and customize the colors or styles using the selectors to the right of the pane.

You can see the effects of your changes in the Preview pane.

Related Information:
Syntax Highlight Preferences (on page 224)

Editing StratML Documents

Strategy Markup Language (StratML) is an XML vocabulary and schema for strategic plans. Oxygen XML Editor supports StratML Part 1 (Strategic Plan) and StratML Part 2 (Performance Plans and Reports) and provides templates for the following documents:

- Strategic Plan (StratML Part 1)
- Performance Plan (StratML Part 2)
- Performance Report - (StratML Part 2)
- Strategic Plan - (StratML Part 2)

You can view the components of a StratML document in the Outline view (on page 539). Oxygen XML Editor implements a default XML with XSLT transformation scenario for this document type, called StratML to HTML.
Editing XLIFF Documents

XLIFF (XML Localization Interchange File Format) is an XML-based format that was designed to standardize the way multilingual data is passed between tools during a localization process. Oxygen XML Editor provides the following support for editing XLIFF documents:

**XLIFF Version 1.2, 2.0, and 2.1 Support:**

- New document templates for XLIFF documents.
- A default CSS file (xliff.css) used for rendering XLIFF content in Author mode is stored in \[OXYGEN_INSTALL_DIR\]/frameworks/xliff/css/.
- Validation and content completion support using local catalogs. The default catalog (catalog.xml) for version 1.2 is stored in \[OXYGEN_INSTALL_DIR\]/frameworks/xliff/schemas/1.2, for version 2.0 in \[OXYGEN_INSTALL_DIR\]/frameworks/xliff/schemas/2.0, and for version 2.1 in \[OXYGEN_INSTALL_DIR\]/frameworks/xliff/schemas/2.1.

**XLIFF Version 2.0 and 2.1 Enhanced Support:**

Support for validating XLIFF 2.0 and 2.1 documents using modules. For version 2.0, the default modules are stored in \[OXYGEN_INSTALL_DIR\]/frameworks/xliff/schemas/2.0/modules and for version 2.1, they are stored in \[OXYGEN_INSTALL_DIR\]/frameworks/xliff/schemas/2.1.

**Editing XLIFF Documents in Author Mode**

By default, when you create a new XLIFF document from a template (on page 368), Oxygen XML Editor opens it in Text mode. Aside from the normal editing features found in Text mode, you can also switch to Author mode where Oxygen XML Editor offers some special form controls specifically for XLIFF documents. These form controls simply allow you to add or edit XLIFF attribute values and content in a visual mode.

For XLIFF version 2.0 and 2.1 documents, you can also change the style of the visual editing mode. The Styles drop-down menu on the toolbar offers the following styles that are specifically designed to render XLIFF 2.0 and 2.1 documents in Author mode:

- Default
- Classic
- Translate

**Editing JavaScript Documents**

This section explains the features of the Oxygen XML Editor JavaScript Editor and how you can use them.
JavaScript Editing Actions

Oxygen XML Editor allows you to create and edit JavaScript files and assists you with useful features such as syntax highlight, content completion, and outline view. To enhance your editing experience, you can select entire blocks (parts of text delimited by brackets) by double-clicking somewhere inside the brackets.

Figure 436. JavaScript Editor Text Mode

The contextual menu of the JavaScript editor offers the following actions:

- **Cut**
  
  Allows you to cut fragments of text from the editing area.

- **Copy**
  
  Allows you to copy fragments of text from the editing area.

- **Paste**
  
  Allows you to paste fragments of text in the editing area.

- **Toggle Comment**
  
  Allows you to comment a line or a fragment of the JavaScript document you are editing. This option inserts a single comment for the entire fragment you want to comment.

- **Toggle Line Comment**
  
  Allows you to comment a line or a fragment of the JavaScript document you are editing. This option inserts a comment for each line of the fragment you want to comment.

- **Go to Matching Bracket**
  
  Use this option to find the closing, or opening bracket, matching the bracket at the cursor position. When you select this option, Oxygen XML Editor moves the cursor to the matching bracket, highlights its row, and decorates the initial bracket with a rectangle.
Note:
A rectangle decorates the opening or closing bracket that matches the current one, at all times.

Source

Allows you to select one of the following actions:

To Lower Case
Converts the selection content to lower case characters.

To Upper Case
Converts the selection content to upper case characters.

Capitalize Lines
Converts to upper case the first character of every selected line.

Join and Normalize Lines
Joins all the rows you select to one row and normalizes the content.

Insert new line after
Inserts a new line after the line at the cursor position.

Modify all matches
Use this option to modify (in-place) all the occurrences of the selected content. When you use this option, a thin rectangle replaces the highlights and allows you to start editing. If matches with different letter cases are found, a dialog box is displayed that allows you select whether you want to modify only matches with the same letter case or all matches.

Open

Allows you to select one of the following actions:

- Open File at Cursor - select this action to open the source of the file located at the cursor position
- Open File at Cursor in System Application - select this action to open the source of the file located at the cursor position with the application that the system associates with the file

Compare

Select this option to open the Compare Files tool to compare the file you are editing with a file you choose in the dialog box.

Folding

When you invoke the contextual menu from the folding (on page 3219) triangles in the stripe on the left side of the editor, the following actions are available:
Collapse Other Folds (Ctrl + NumPad/ (Command + NumPad/ on macOS))
Folds all the elements except the current element.

Collapse Child Folds (Ctrl + NumPad. (Command + NumPad. on macOS))
Folds the elements indented with one level inside the current element.

Expand Child Folds
Unfolds all child elements of the currently selected element.

Expand All (Ctrl + NumPad* (Command + NumPad* on macOS))
Unfolds all elements in the current document.

Validating JavaScript Files
You have the possibility to validate the JavaScript document you are editing. Oxygen XML Editor uses the Mozilla Rhino library for validation. For more information about this library, go to https://github.com/mozilla/rhino. The JavaScript validation process checks for errors in the syntax. Calling a function that is not defined is not treated as an error by the validation process. The interpreter discovers this error when executing the faulted line. Oxygen XML Editor can validate a JavaScript document both on-request and automatically.

Content Completion in JavaScript Documents
When you edit a JavaScript document, the Content Completion Assistant (on page 3217) presents you a list of the elements you can insert at the cursor position. It can be manually activated with the Ctrl + Space (Command + Space on macOS) shortcut.

For an enhanced assistance, JQuery methods are also presented. The following icons decorate the elements in the content completion list of proposals depending on their type:

- f - function
- v - variable
- o - object
- p - property
- M - method

Note:
These icons decorate both the elements from the content completion list of proposals and from the Outline view (on page 1189).
The **Content Completion Assistant** collects:

- Method names from the current file and from the library files.
- Functions and variables defined in the current file.

If you edit the content of a function, the content completion list of proposals contains all the local variables defined in the current function, or in the functions that contain the current one.

### Syntax Highlighting in JavaScript Documents

Oxygen XML Editor supports syntax highlighting in **Text** mode to make it easier to read the semantics of the structured content by displaying each type of code in different colors and fonts.

To customize the colors or styles used for the syntax highlighting colors for JavaScript files, follow these steps:

1. Open the **Preferences** dialog box (Options > Preferences) (on page 124).
2. Go to Editor > Syntax Highlight (on page 224).
3. Select and expand the JavaScript section in the top pane.
4. Select the component you want to change and customize the colors or styles using the selectors to the right of the pane.
You can see the effects of your changes in the Preview pane.

Related Information:

Syntax Highlight Preferences (on page 224)

**JavaScript Outline View**

Oxygen XML Editor present a list of all the components of the JavaScript document you are editing in the Outline view. By default, it is displayed on the left side of the editor. If the view is not displayed, it can be opened by selecting it from the Window > Show View menu.

![JavaScript Outline View](image)

The following icons decorate the elements in the Outline view depending on their type:

- **function**
- **variable**
- **object**
- **property**
- **method**

The contextual menu of the JavaScript Outline view contains the usual Cut, Copy, Paste, and Delete actions. From the Settings menu, you can select the Update selection on cursor move option to synchronize the Outline view with the editing area.
Editing XProc Scripts

XProc is an XML pipeline language that can be used to script transformations. An XProc script is edited as an XML document that is validated against a RELAX NG schema, or if the script has an associated transformation scenario, then the XProc engine selected in the scenario is used as the validating engine (if the XProc engine supports validation). The default engine for XProc scenarios is a version of the Calabash engine that comes bundled with Oxygen XML Editor version 24.1.

XProc Content Completion

Oxygen XML Editor helps you edit a XProc scripts through the Content Completion Assistant (on page 3217), offering proposals that are valid at the cursor position. It can be manually activated with the Ctrl + Space (Command + Space on macOS) shortcut.

The content completion inside the `<input/inline>` element from the XProc namespace http://www.w3.org/ns/xproc offers elements from the following schemas depending both on the `@port` attribute and the parent of the `<input>` element. When invoking the content completion inside the `<inline>` XProc element, the list of content completion proposals is populated as follows:

- If the value of the `@port` attribute is `stylesheet` and the `<xslt>` element is the parent of the `<input>` elements, the Content Completion Assistant offers XSLT elements.
- If the value of the `@port` attribute is `schema` and the `<validate-with-relax-ng>` element is the parent of the `<input>` element, the Content Completion Assistant offers RELAX NG schema elements.
- If the value of the `@port` attribute is `schema` and the `<validate-with-xml-schema>` element is the parent of the `<input>` element, the Content Completion Assistant offers XML Schema schema elements.
- If the value of the `@port` attribute is `schema` and the `<validate-with-schematron>` element is the parent of the `<input>` element, the Content Completion Assistant offers either ISO Schematron elements or Schematron 1.5 schema elements.
- If the above cases do not apply, then the Content Completion Assistant offers elements from all the schemas from the above cases.
Figure 439. XProc Content Completion

XProc Syntax Highlighting

The XProc editor assists you in writing XPath expressions by offering dedicated coloring schemes for syntax highlighting.

To customize the colors or styles used for the syntax highlighting colors for XProc, follow these steps:

1. Open the Preferences dialog box (Options > Preferences) (on page 124).
2. Go to Editor > Syntax Highlight (on page 224).
3. Select and expand the XML section in the top pane.
4. Select the component you want to change and customize the colors or styles using the selectors to the right of the pane.
5. Select the XML tab in the Preview pane to see the effects of your changes.

Enabling Extensions in Calabash

If you are using the default Calabash engine, it is possible to configure extensions (for a list of the valid extensions, see http://xmlcalabash.com/docs/reference/cfg.extension.html).

To configure an extension:

1. Edit the following file: OXYGEN_INSTALL_DIR/lib/xproc/calabash/engine.xml.
2. Add the extension and its value as a system-property, as in the following example:

```xml
<system-property name="com.xmlcalabash.allow-text-results" value="true"/>
```
Related Information:
- Creating an XProc Transformation Scenario (on page 1530)
- Integrating an External XProc Engine (on page 1534)
- XProc Preferences (on page 244)

Editing Schematron Schemas

Schematron is a simple and powerful Structural Schema Language for making assertions about patterns found in XML documents. It relies almost entirely on XPath query patterns for defining rules and checks. Schematron validation rules allow you to specify a meaningful error message. This error message is provided to you if an error is encountered during the validation stage.

There are numerous online resources out there to help you get started with writing Schematron rules. Here are just a few that might help you:

- Guide to Schema Writing with Schematron
- Presentation: Schematron Development with Oxygen

Oxygen XML Editor assists you in editing Schematron documents with schema-based content completion, syntax highlight, search and refactor actions, and dedicated icons for the Outline view (on page 1212). You can create a new Schematron schema using one of the Schematron templates available in the New document wizard (on page 368).

For information about applying and detecting Schematron schemas, see Associating a Schema to XML Documents (on page 816).

Validating XML Documents Against Schematron

The Skeleton XSLT processor is used for validation and conforms with ISO Schematron or Schematron 1.5. It allows you to validate XML documents against Schematron schemas (on page 1206) or against combined RELAX NG / W3C XML Schema and Schematron.

How to Specify the Query Language Binding

You can specify the query language binding to be used in the Schematron schema by doing the following:
• For embedded ISO Schematron, open the Preferences dialog box (Options > Preferences) (on page 124), go to XML > XML Parser > Schematron, and select it in the Embedded rules query language binding option (on page 240).

• For standalone ISO Schematron, specify the version by setting the query language to be used in a @queryBinding attribute on the schema root element. For more information, see the Query Language Binding section of the Schematron specifications.

• For Schematron 1.5 (standalone and embedded), open the Preferences dialog box (Options > Preferences) (on page 124), go to XML > XML Parser > Schematron, and select the version in the XPath Version option (on page 241).

Multi-Lingual Support in Schematron Messages

You can specify the desired language for the validation messages in the Schematron Preferences page (on page 240). The Schematron validation messages can be presented in multiple languages by defining the language for each message using the Schematron <diagnostics> element.

For example, you can define a diagnostic for each language and reference the ID of the diagnostics in the <assert> element. You can specify the language of the diagnostic message by adding the xml:lang attribute on the sch:diagnostic element or on its parent:

```xml
<sch:assert test="bone" diagnostics="d_en d_de">
   A dog should have a bone.
</sch:assert>
...

<sch:diagnostics>
   <sch:diagnostic id="d_en" xml:lang="en">
      A dog should have a bone.
   </sch:diagnostic>
   <sch:diagnostic id="d_de" xml:lang="de">
      Das Hund muss ein Bein haben.
   </sch:diagnostic>
</sch:diagnostics>
```

How to Customize Color Schemes in Schematron

The Schematron editor renders the XPath expressions with dedicated color schemes. To customize the coloring schemes, open the Preferences dialog box (Options > Preferences) (on page 124) and go to Editor > Syntax Highlight.

Schematron Transformation Scenario

When you create a Schematron document, Oxygen XML Editor provides a built-in transformation scenario. You can use this scenario to obtain the XSLT style-sheet corresponding to the Schematron schema. You can apply this XSLT stylesheet to XML documents to obtain the Schematron validation results.

For more information about the Schematron support in Oxygen XML Editor, watch our video demonstrations:
Examples of Schematron Rules and Quick Fixes

This topic is meant to provide some basic examples of Schematron Rules and Schematron Quick Fixes (SQF) to help you create and impose your own rules and quick fixes.

Other examples and ideas can also be found at:

- Public GitHub project with the Schematron file used for Oxygen's User Guide
- Public GitHub project with sample Schematron Quick Fixes

Schematron Examples

Schematron Use Case 1: Impose a Relax NG Schema Declaration

**Description:** The following sample rule is useful if, for example, you need to enforce the use of Relax NG schema declarations in all of your documents (i.e. instead of using DTD schemas).

**Sample Code:**

```xml
<?xml version="1.0" encoding="UTF-8"?>
<sch:schema xmlns:sch="http://purl.oclc.org/dsdl/schematron"
     queryBinding="xslt2" xmlns:saxon="http://saxon.sf.net"/>
<sch:let name="rngDeclaration"
     value="processing-instruction('xml-model')[saxon:get-pseudo-attribute('schematypens')='http://relaxng.org/ns/structure/1.0']"/>
<sch:pattern>
    <sch:rule context="/element()">
        <sch:assert test="exists($rngDeclaration)">You must define a Relax NG schema declaration in the document (DTD schemas are not supported).</sch:assert>
    </sch:rule>
</sch:pattern>
</sch:schema>
```

**Result:** The engine checks for a Relax NG schema declaration in the document and displays an error if it is missing. The error is reported on the document's root element (/element()).
Schematron Use Case 2: Check for Missing IDs

**Description:** The following sample rule checks for missing or undefined IDs in a TEI document. Specifically, it looks for IDs from the `tei:rs/@ref` attribute defined in the document named `persons.xml` (as `xml:id` of a TEI `person` element).

**Sample Code:**

```xml
<?xml version="1.0" encoding="UTF-8"?>
<sch:schema xmlns:sch="http://purl.oclc.org/dsdl/schematron" queryBinding="xslt2">
  <sch:ns uri="http://www.tei-c.org/ns/1.0" prefix="tei"/>
  <sch:let name="personIds" value="document('../persons.xml')/tei:TEI//tei:person/@xml:id"/>
  <sch:pattern>
    <sch:rule context="tei:rs">
      <sch:let name="refIds" value="for $id in tokenize(@ref, ' ') return substring-after($id, '#')"/>
      <sch:let name="missingIds" value="for $id in $refIds return (if($id = $personIds) then '' else $id)"/>
      <sch:report test="$missingIds != ''">
        The following ids "<sch:value-of select="$missingIds"/>" are not defined in "<sch:value-of select="$personIds"/>
      </sch:report>
    </sch:rule>
  </sch:pattern>
</sch:schema>
```

where the XML document looks something like this:

```xml
<tei xmlns="http://www.tei-c.org/ns/1.0">
  <rs ref="../SomePerson/persons.xml#EDP ../personography/HAMpersons.xml#SD">text</rs>
  <rs ref="../SomePerson/persons.xml#EDP">text</rs>
</tei>
```

**Result:** The engine displays an error message listing the missing/undefined IDs.

Schematron Use Case 3: Check for Broken Links

**Description:** The following sample rule detects broken links in DITA `<xref>` or `<link>` elements. The first example only checks links that do not contain an anchor (#).

**Sample Code:**

```xml
<rule
  context="*[contains(@class, ' topic/xref ') or contains(@class, ' topic/link ')]
  [@href][not(contains(@href, '#'))][not(@scope = 'external')]"
```
For links that contain an anchor, the Schematron rule must look something like this:

```xml
<rule context="*[contains(@class, 'topic/xref') or contains(@class, 'topic/link')][@href][contains(@href, '#')][not(@scope = 'external')][not(@type) or @type='dita']">
  <let name="file" value="substring-before(@href, '#')"/>
  <let name="idPart" value="substring-after(@href, '#')"/>
  <let name="topicId" value="if (contains($idPart, '/')) then substring-before($idPart, '/') else $idPart"/>
  <let name="id" value="substring-after($idPart, '/')"/>
  <assert test="document($file, .)//*[@id=$topicId]">
    Invalid topic id "<value-of select="$topicId"/>
  </assert>
  <assert test="$id ='' or document($file, .)//*[@id=$id][ancestor::*[contains(@class, 'topic/topic')][1][@id=$topicId]]">
    The id "<value-of select="$id"/" is not in the scope of the referenced topic id "<value-of select="$topicId"/>. </assert>
</rule>
```

**Result:** The engine displays an error message when a broken link or cross reference is detected.

### Schematron Use Case 4: Check for Duplicate IDs

**Description:** The following sample rule detects if there are two sibling `<step>` elements with the same `@id` value in a DITA Task document.

**Sample Code:**

```xml
<sch:rule context="*[contains(@class, 'task/step')]">
  <sch:let name="id" value="@id"/>
  <sch:report test="preceding-sibling::element()[contains(@class, 'task/step')][@id = $id]">
    Element with duplicate ID "<sch:value-of select="$id"/>" detected.
  </sch:report>
</sch:rule>
```
**Result:** The engine displays an error message when a duplicate ID is detected in sibling `<step>` elements within a DITA Task document.

**Schematron Use Case 5: Check for Duplicate DITA Topic References**

**Description:** The following sample rule checks a DITA map for duplicate `<topicref>` elements with the same `@href` value.

**Sample Code:**

```xml
<sch:rule context="*[contains(@class, ' map/topicref ')]">
  <sch:let name="href" value="@href"/>
  <sch:report
    test="preceding::element()[contains(@class, ' map/topicref ')][@href = $href]">
    Duplicate topicref "<sch:value-of select="$href"/>" detected in map.
  </sch:report>
</sch:rule>
```

**Result:** The engine displays an error message when multiple `<topicref>` elements with the same `@href` value are detected in a DITA map.

**Schematron Use Case 6: Restrict Certain Words from the Title**

**Description:** The following sample rule checks for instances of specified words to be restricted from a `<title>` element (in this example, the words `test` and `hello` are restricted).

**Sample Code:**

```xml
<sch:schema xmlns:sch="http://purl.oclc.org/dsdl/schematron"
  queryBinding="xslt2">
  <sch:let name="words" value="'test,hello'"/>
  <sch:let name="wordsToMatch" value="replace($words, ',', '|')"/>
  <sch:pattern>
    <sch:rule context="title">
      <sch:report
test="matches(text(), $wordsToMatch)" role="warn">
        The following words should not be added in the title:
        <sch:value-of select="$words"/>
      </sch:report>
    </sch:rule>
  </sch:pattern>
</sch:schema>
```

**Result:** The engine displays an error message if one of the specified restricted words appear in a title.
Schematron Use Case 7: Check the Location of a Resource

Description: The following sample rule checks if the path to a resource (in this case, an image) is specified correctly. Specifically, this sample rule reports that the image must be located in the current project (the images location must be relative to the parent folder and no more than one "../" in the path.

Sample Code:

```xml
<sch:rule context="image">
  <sch:report test="count(tokenize(@href, '\.\./')) > 2"> The image must be located in the current project. It is currently located in: <sch:value-of select="@href"/>
  </sch:report>
</sch:rule>
```

Result: The engine displays an error message if an image is detected in a location other than the current project, relative to the parent folder.

Schematron Use Case 8: Check for Extra Spaces at Beginning/End of Elements

Description: The following sample rule checks for spaces at the beginning and end of elements.

Tip: You could specify a list of elements to check to make the rule context-sensitive.

Sample Code:

```xml
<rule context="p|ph|codeph|filename|indexterm|xref|user-defined|user-input">
  <let name="firstNodeIsElement" value="node()[1] instance of element()"/>
  <let name="lastNodeIsElement" value="node()[last()] instance of element()"/>
  <report test="(not($firstNodeIsElement) and matches(.,'^\s',';j'))
    or (not($lastNodeIsElement) and matches(.,'\s$',';j'))" role="warning"> Textual elements should not begin or end with whitespace.</report>
</rule>
```

Result: The engine displays an error message if a whitespace is detected at the beginning or end of a textual element.

Schematron Use Case 9: Impose Capitalizing the First Letter

Description: The following sample rule detects if elements start with a capital letter or a number. The rule is implemented using abstract patterns. The abstract pattern starts-with-capital has one argument representing the element to be checked. There are two implementations of the abstract pattern, one that specifies the <tittle> element as the element to verify, and one that specifies the <li> element.

Sample Code:
Schematron Use Case 10: Check for Specified Terms in a Paragraph

Description: The following sample rule checks if any DITA `<p>` elements contain certain keywords defined in an external document.

Sample Code:

```xml
<sch:pattern>
  <sch:let name="keys" value="document('keys-common.ditamap')//keyword"/>
  <sch:rule context="p">
    <sch:let name="text" value="."/>
    <sch:let name="matchedKeys" value="keys[contains($text, normalize-space(.))][]"/>
    <sch:report id="now001" test="count($matchedKeys) > 0" role="error">
      The paragraph text contains the keywords: <sch:value-of select="$matchedKeys"/>
    </sch:report>
  </sch:rule>
</sch:pattern>
```

Result: The engine displays an error message if any of the keywords listed in an external document are detected within a DITA `<p>` element.

Schematron Use Case 11: Impose a Minimum Value

Description: The following sample rule determines the `<type>` element value with the minimum version specified by the `@version` attribute and then verifies that they are all equal to the determined value.

Sample Code:

```xml
<sch:schema xmlns:sch="http://purl.oclc.org/dsdl/schematron" queryBinding="xslt2">
  <sch:let name="typeValue" value="//Node1[not(@version >
```
where the XML file would look something like this:

```xml
<root>
  <Node1 version="1">
    <Element1>Value1</Element1>
    <Type>123456</Type>
  </Node1>
  <Node1 version="2">
    <Element1>Value1</Element1>
    <Type>123456</Type>
  </Node1>
  <Node1 version="3">
    <Element1>Value1</Element1>
    <Type>1234567</Type>
  </Node1>
</root>
```

**Result:** The engine displays an error message if a `<type>` element value does not equal the minimum version specified by the `<version>` attribute.

### SQF (Schematron Quick Fix) Examples

#### SQF Use Case 1: Impose a DITA Prolog

**Description:** The following sample Schematron rule checks a DITA topic to make sure it contains `<prolog>`, `<critdates>`, `<revised>` elements and the sample Quick Fix proposes options for inserting the missing elements.

**Sample Code:**

```xml
<sch:schema xmlns:sch="http://purl.oclc.org/dsdl/schematron" queryBinding="xslt2"
            xmlns:sqf="http://www.schematron-quickfix.com/validator/process">
  <sch:pattern>
    <sch:rule context="*[contains(@class, ' topic/topic ')]">
      <sch:assert sqf:fix="add_prolog" test="prolog" role="warn">Every topic must contain prolog/critdates/revised elements where the revised modified date is in
```
YYYY-MM-DD format.</sch:assert>

<sqf:fix id="add_prolog">
  <sqf:description>
    <sqf:title>Add prolog/critdates/revised elements, where the revised element's
    @modified attribute value is the current date in YYYY-MM-DD
    format.</sqf:title>
  </sqf:description>
  <sqf:add match="*[contains(@class, ' topic/body ')]" node-type="element"
position="before" target="prolog">
    <critdates>
      <revised modified=""/>
    </critdates>
  </sqf:add>
</sqf:fix>
</sch:rule>

<sch:rule context="*[contains(@class, ' topic/prolog ')]">
  <sch:report role="warn" test="not(critdates)" sqf:fix="add_critdates">
    The prolog element must have critdates/revised elements with the @modified attribute value in YYYY-MM-DD format.</sch:report>
  <sqf:fix id="add_critdates">
    <sqf:description>
      <sqf:title>Add the critdates element.</sqf:title>
    </sqf:description>
    <sqf:add node-type="element" target="critdates">
      <revised modified=""/>
    </sqf:add>
  </sqf:fix>
</sch:rule>

<sch:rule context="*[contains(@class, ' topic/critdates ')]">
  <sch:report role="warn" test="not(revised)" sqf:fix="add_revised">
    The critdates element must have revised @modified in YYYY-MM-DD format. </sch:report>
  <sqf:fix id="add_revised">
    <sqf:description>
      <sqf:title>Add the revised element.</sqf:title>
    </sqf:description>
    <sqf:add node-type="element" target="revised"/>
  </sqf:fix>
</sch:rule>
Result: The engine displays an error message if the `<prolog>`, `<critdates>`, or `<revised>` elements are missing from a DITA topic and the Quick Fix mechanism proposes options for inserting the missing elements.

SQF Use Case 2: Impose an ID for all DITA Section Elements

Description: The following sample Schematron rule checks if each DITA `<section>` element has a specified ID and the sample Quick Fix proposes options for inserting the missing IDs.

Sample Code:

```xml
<!-- Add IDs to all sections to impose link targets -->

<!-- Generate an id based on the section title. If there is no title then generate a random id. -->
```

```xml
<!-- Add IDs to all sections to impose link targets -->
<!-- Generate an id based on the section title. If there is no title then generate a random id. -->
```

```xml
<sqf:fix id="addId">
  <sqf:description>
    <sqf:title>Add @id to the current section</sqf:title>
    <sqf:p>Add an @id attribute to the current section. The ID is generated from the section title.</sqf:p>
  </sqf:description>
</sqf:fix>

```xml
<sqf:fix id="addIds">
  <sqf:description>
    <sqf:title>Add @id to all sections</sqf:title>
    <sqf:p>Add an @id attribute to each section from the document. The ID is generated from the section title.</sqf:p>
  </sqf:description>
</sqf:fix>
```
Oxygen XML Editor 24.1 | 8 - Editing Documents | 1203

<!-- Generate an id based on the section title. If there is no title then
generate a random id. -->
<sqf:add match="//section[not(@id)]" target="id" node-type="attribute"
select="
concat('section_',
if (exists(title) and string-length(title) > 0)
then substring(lower-case(replace(replace(
normalize-space(string(title)), '\s', '_'),
'[^a-zA-Z0-9_]', '')), 0, 50)
else generate-id())"/>
</sqf:fix>
</sch:rule>
</sch:pattern>
</sch:schema>

Result: The engine displays an error message if an @id attribute is missing for any <section> element in a DITA
topic and the Quick Fix mechanism proposes options for inserting the missing ID.

SQF Use Case 3: Impose a Short Description in an Abstract Element
Description: The following sample Schematron rule checks a DITA topic to make sure it contains a <shortdesc>
element inside an <abstract> element and the sample Quick Fix proposes options for correcting the missing
structure.
Sample Code:
<sch:schema xmlns:sch="http://purl.oclc.org/dsdl/schematron" queryBinding="xslt2"
xmlns:sqf="http://www.schematron-quickfix.com/validator/process"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<sch:pattern>
<sch:rule context="shortdesc">
<sch:assert test="parent::abstract" sqf:fix="moveToAbstract moveToExistingAbstract">
The short description must be added in an abstract element
</sch:assert>
<!-- Check if there is an abstarct element -->
<sch:let name="abstractElem" value="preceding-sibling::abstract |
following-sibling::abstract"/>

<!-- Create an abstract element and add the short description -->
<sqf:fix id="moveToAbstract" use-when="not($abstractElem)">
<sqf:description>
<sqf:title>Move short description in an abstract element</sqf:title>
</sqf:description>
sqf:replace>
<abstract>


Result: The engine displays an error message if an <abstract> element does not contain a <shortdesc> element and the Quick Fix mechanism proposes options for inserting the missing structure or to move the <shortdesc> element inside the <abstract> element.

SQF Use Case 4: Impose a Certain Article Type

Description: The following sample Schematron rule checks the @article-type attribute to make sure its value is one of the specified allowed values (abstract, addendum, announcement, article-commentary) and the sample Quick Fix proposes options for replacing any other detected value with one of the allowed values.

Sample Code:
<sch:let name="articleTypes" value="('abstract', 'addendum', 'announcement',
' article-commentary')"/>

<sch:pattern>
<sch:rule context="article/@article-type">
  <sch:assert test=".
  = $articleTypes" sqf:fix="setArticleType">
    Should be one of the article types:
    <sch:value-of select="$articleTypes"/>
  </sch:assert>

  <sqf:fix id="setArticleType" use-for-each="$articleTypes">
    <sqf:description>
      <sqf:title>Set article type to '
        <sch:value-of select="$sqf:current"/>
      '</sqf:title>
    </sqf:description>

    <sqf:replace node-type="attribute" target="article-type" select="$sqf:current"/>
  </sqf:fix>
</sch:rule>
</sch:pattern>

Result: The engine displays an error message if an @article-type attribute has any other value other than abstract, addendum, announcement, or article-commentary and the Quick Fix mechanism proposes options for replacing the disallowed value with one of those four allowed values (using the use-for-each construct).

**SQF Use Case 5: Impose Certain Attributes and Values**

**Description:** The following sample Schematron rule checks the @rowsep and @colsep attributes are added on the <table> element and their value is set to 1. The Quick Fix proposes options for adding the attributes in case they are missing or set the correct value.

**Sample Code:**

```xml
<?xml version="1.0" encoding="UTF-8"?>
<sch:schema xmlns:sch="http://purl.oclc.org/dsdl/schematron" queryBinding="xslt2"
  xmlns:sqf="http://www.schematron-quickfix.com/validator/process">
  <sch:pattern>
    <sch:rule context="table">
      <sch:assert test="@rowsep = 1" sqf:fix="addRowsep">The @rowsep should be set to 1</sch:assert>

      <sch:assert test="@colsep = 1" sqf:fix="addColsep">The @colsep should be set to 1</sch:assert>

      <sqf:fix id="addRowsep">
        <sqf:description>
          <sqf:title>Add @rowsep attribute</sqf:title>
        </sqf:description>
      </sqf:fix>
    </sch:rule>
  </sch:pattern>
</sch:schema>
```
Modular Contextual Schematron Editing Using 'Main Files' Support

Smaller interrelated modules that define a complex Schematron cannot be correctly edited or validated individually, due to their interdependency with other modules. For example, a diagnostic defined in a main schema document is not visible when you edit an included module. Oxygen XML Editor provides the support for defining the main module (or modules), thus allowing you to edit any of the imported/included schema files in the context of the larger schema structure.

You can set a main Schematron document either using the main files support from the Project view (on page 418), or using a validation scenario.

To set a main file using a validation scenario, add validation units that point to the main schemas. Oxygen XML Editor warns you if the current module is not part of the dependencies graph computed for the main schema. In this case, it considers the current module as the main schema.

The advantages of editing in the context of main file include:

• Correct validation of a module in the context of a larger schema structure.
• Content Completion Assistant (on page 3217) displays all the referable components valid in the current context. This include components defined in modules other than the currently edited one.

Presenting Schematron Validation Issues

The possible issues that might occur during the validation process when validating XML documents against Schematron are presented with colored underlines in the editing pane, colored markers in the right vertical stripe, and details about the issues are presented in the Errors panel at the bottom area of the Oxygen XML Editor window. Each error is flagged with a severity level that can be: warning, error, fatal or info.

To set a severity level, Oxygen XML Editor looks for the following information:
• The role attribute, which can have one of the following values:
  - **warn** or **warning** - Sets the severity level to *warning*. By default, underlined with a yellow squiggly line in the editing pane and a yellow marker in the right vertical stripe.
  - **error** - Sets the severity level to *error*. By default, underlined with a red squiggly line in the editing pane and a red marker in the right vertical stripe.
  - **fatal** - Sets the severity level to *fatal*. By default, underlined with a red squiggly line in the editing pane and a red marker in the right vertical stripe.
  - **info** or **information** - Sets the severity level to *information*. By default, underlined with a blue squiggly line in the editing pane and a blue marker in the right vertical stripe.

• The start of the message, after trimming leading white-spaces. Oxygen XML Editor looks to match the following exact string of characters (case-sensitive):
  - **Warning**: - Sets the severity level to *warning*. By default, underlined with a yellow squiggly line in the editing pane and a yellow marker in the right vertical stripe.
  - **Error**: - Sets the severity level to *error*. By default, underlined with a red squiggly line in the editing pane and a red marker in the right vertical stripe.
  - **Fatal**: - Sets the severity level to *fatal*. By default, underlined with a red squiggly line in the editing pane and a red marker in the right vertical stripe.
  - **Info**: - Sets the severity level to *info*. By default, underlined with a blue squiggly line in the editing pane and a blue marker in the right vertical stripe.

• If none of the previous rules match, Oxygen XML Editor sets the severity level to *error*. By default, underlined with a red squiggly line in the editing pane and a red marker in the right vertical stripe.

**Tip:**
You can configure the color for each type in the *Document Checking* preferences page (on page 226).

**Related Information:**
- Validating XML Documents Against a Schema (on page 775)
- Validation Scenario (on page 787)
- Associating a Schema to XML Documents (on page 816)
- Presenting Validation Errors in Author Mode (on page 780)
- Presenting Validation Errors in Text Mode (on page 777)

**Integrating Schematron Rules in a Framework and Sharing Them**

Custom Schematron rules are a great way to ensure consistency for XML authoring, especially when there is a large team working on the same set of documents. You can use Schematron for numerous use cases. For example, to restrict certain elements from being used, to impose restrictions on the amount of text for an element, or to impose restrictions on certain elements based on various attribute values or text content set in other elements. Furthermore, you can define quick fixes for each Schematron rule (on page 1236) to offer technical writers proposed solutions for reported problems.
Once you define the Schematron rules, they can be shared with the other members of your team by integrating them in a framework (on page 3219) (document type) configuration.

**How to Integrate Schematron Rules in a Framework**

To integrate a Schematron rule in an existing framework bundled with the application, follow these steps:

1. Create a folder structure for an extended framework and save it somewhere on disk where you have full write access (for example, custom Frameworks/dita-extension).
2. In that new folder structure, create another folder that will contain all of your custom Schematron files (for example, custom Frameworks/dita-extension/rules).
3. Define the Schematron rules in an existing or new Schematron file and save it in the folder you created in step 2. There are numerous online resources out there to help you get started with writing Schematron rules. Here are just a few that might help you:
   - Guide to Schema Writing with Schematron
   - Presentation: Schematron Development with Oxygen
4. Open the Preferences dialog box (Options > Preferences) (on page 124) and go to Document Type Association > Locations (on page 139). In this preferences page, add the path to your custom Frameworks folder in the Additional frameworks directories list, then click OK or Apply to save your changes.
5. Go to the Document Type Association preferences page (on page 138) and select a framework configuration (for example, DITA) and use the Extend button to create an extension for it.
6. Give the extension an appropriate name (for example, DITA - Custom), select External for the Storage option, and specify an appropriate path to your framework configuration file (for example, path/to/.../custom Frameworks/dita-extension/dita-extension.framework).
7. Make whatever changes you desire to the extension, then go to the Validation tab, edit the default validation scenario (select the scenario and click the Edit button), and add an extra validation unit to it (one that uses your custom Schematron file). For more details about editing validation scenarios, see Configuring Validation Scenarios for a Framework (on page 2218).
8. Click OK to close the dialog box and then OK or Apply to save the changes to the Document Type Association preferences page (on page 138).
9. Open an XML document that matches your framework configuration and test the new rule.
10. You can continue to refine the Schematron and develop additional rules as needed.

**Sharing Schematron Rules**

To share Schematron rules with other members of your team, you simply need to share the framework where you integrated the Schematron rules. There are several methods for sharing frameworks and you can find details here: Sharing a Framework (on page 2288).
Validating Schematron Documents

By default, a Schematron schema is validated as you type. To change this, open the Preferences dialog box (Options > Preferences) (on page 124), go to Editor > Document Checking, and deselect the Enable automatic validation option (on page 226).

To validate a Schematron document manually, select the Validate action from the Validation toolbar drop-down menu or the Document > Validate menu. When Oxygen XML Editor validates a Schematron schema, it expands all the included modules so the entire schema hierarchy is validated. The validation problems are highlighted directly in the editor, making it easy to locate and fix any issues.

Oxygen XML Editor offers an error management mechanism capable of pinpointing errors in XPath expressions and in the included schema modules.

Related Information:
Presenting Schematron Validation Issues (on page 1206)

Content Completion in Schematron Documents

Oxygen XML Editor helps you edit a Schematron schema through the Content Completion Assistant (on page 3217), offering proposals that are valid at the cursor position. It can be manually activated with the Ctrl + Space (Command + Space on macOS) shortcut.

When you edit the value of an attribute that refers a component, the proposed components are collected from the entire schema hierarchy. For example, if the editing context is phase/active/@pattern, the Content Completion Assistant proposes all the defined patterns.

Note:

For Schematron resources, the Content Completion Assistant collects its components starting from the main files (on page 3220). The main files can be defined in the project or in the associated validation scenario. For further details about the Main Files support go to Defining Main Files at Project Level (on page 418).

If the editing context is an attribute value that is an XPath expression (such as assert/@test or report/@test), the Content Completion Assistant offers the names of XPath functions, the XPath axes, and user-defined variables.
The **Content Completion Assistant** displays XSLT 1.0 functions and optionally XSLT 2.0 / 3.0 functions in the attributes `path`, `select`, `context`, `subject`, `test` depending on the Schematron options (on page 240) that are set in Preferences pages. If the Saxon 6.5.5 namespace (`xmlns:saxon="http://icl.com/saxon"`) or the Saxon 10.6 namespace is declared in the Schematron schema (`xmlns:saxon="http://saxon.sf.net/"`) the content completion also displays the XSLT Saxon extension functions as in the following figure:

![Figure 440. XSLT Extension Functions in Schematron Schema Content Completion](image)

The **Content Completion Assistant** also includes code templates that can be used to quickly insert code fragments (on page 536) into Schematron documents.

**Syntax Highlighting in Schematron**

Oxygen XML Editor supports syntax highlighting in Text mode to make it easier to read the semantics of the structured content by displaying each type of code in different colors and fonts.

To customize the colors or styles used for the syntax highlighting colors for Schematron schemas, follow these steps:

1. Open the **Preferences** dialog box (Options > Preferences) (on page 124).
2. Go to Editor > Syntax Highlight (on page 224).
3. Select and expand the XML section in the top pane.
4. Select the component you want to change and customize the colors or styles using the selectors to the right of the pane.
5. Select the XML tab in the Preview pane to see the effects of your changes.
Oxygen XML Editor also allows you to specify syntax highlighting colors for specific XML elements and attributes with specific namespace prefixes. This can be done in the Editor > Syntax Highlight > Elements/Attributes by Prefix preferences page (on page 225).

Related Information:
Syntax Highlight Preferences (on page 224)

Embedding Schematron Rules in XML Schema or RELAX NG

Schematron rules can be embedded into an XML Schema through annotations (using the `<appinfo>` element), or in any element on any level of a RELAX NG Schema (taking into account that the RELAX NG validator ignores all elements that are not in the RELAX NG namespace).

Oxygen XML Editor supports Schematron validation schemas and it is able to extract and use the embedded rules.

Validating XML Documents with XML Schema and Embedded Schematron

To validate an XML document with XML Schema and its embedded Schematron, you can associate the document like this:

```xml
<?xml-model href="percent.xsd" type="application/xml"
   schematypens="http://purl.oclc.org/dsdl/schematron"?>
```

Validating XML Documents with RELAX NG and Embedded Schematron

To validate an XML document with RELAX NG schema and its embedded Schematron rules, you need to associate the document with both schemas like this:

```xml
<?xml-model href="percent.rng" type="application/xml"
   schematypens="http://relaxng.org/ns/structure/1.0"?>
<?xml-model href="percent.rng" type="application/xml"
   schematypens="http://purl.oclc.org/dsdl/schematron"?>
```

The second association validates your document with Schematron rules extracted from the RELAX NG Schema.

Note:
When you work with XML Schema or Relax NG documents that have embedded Schematron rules, Oxygen XML Editor provides two built-in validation scenarios: Validate XML Schema with embedded Schematron for XML schema, and Validate Relax NG with embedded Schematron for Relax NG. You can use one of these scenarios to validate the embedded Schematron rules.
Example: Embedded Schematron in XML Schema

```xml
<xsd:appinfo>
  <sch:pattern>
    <sch:rule context="...">
      <sch:assert test="...">Message.</sch:assert>
    </sch:rule>
  </sch:pattern>
</xsd:appinfo>
```

Example: Embedded Schematron in Relax NG Schema

```xml
<grammar
  xmlns="http://relaxng.org/ns/structure/1.0"
  xmlns:sch="http://purl.oclc.org/dsdl/schematron">
  <sch:pattern>
    <sch:rule context="...">
      <sch:assert test="...">Message.</sch:assert>
    </sch:rule>
  </sch:pattern>
</grammar>
```

Related Information:
Embedding Schematron Quick Fixes in Relax NG or XML Schema (on page 1250)

Schematron Outline View

The Outline view for Schematron schemas presents a list of components in a tree-like structure and it allows for quick access to a component by name. By default, it is displayed on the left side of the editor. If the view is not displayed, it can be opened by selecting it from the Window > Show View menu.
The following actions are available in the Settings menu on the Outline view toolbar:

Filter returns exact matches

The text filter of the Outline view returns only exact matches.

Selection update on cursor move

Controls the synchronization between Outline view and source document. The selection in the Outline view can be synchronized with the cursor moves or the changes in the editor. Selecting one of the components from the Outline view also selects the corresponding item in the source document.

Flat presentation mode of the filtered results

When active, the application flattens the filtered result elements to a single level.

Show comments and processing instructions

Show/hide comments and processing instructions in the Outline view.

Show element name

Show/hide element name.

Show text

Show/hide additional text content for the displayed elements.

Show attributes

Show/hide attribute values for the displayed elements. The displayed attribute values can be changed from the Outline preferences panel (on page 306).

Configure displayed attributes

Displays the XML Structured Outline preferences page (on page 306).
The following contextual menu actions are also available in the Outline view:

**Append Child**
Displays a list of elements that you can insert as children of the current element.

**Insert Before**
Displays a list of elements that you can insert as siblings of the current element, before the current element.

**Insert After**
Displays a list of elements that you can insert as siblings of the current element, after the current element.

**Edit Attributes**
Opens a dialog box that allows you to edit the attributes of the currently selected component.

**Toggle Comment**
Comments/uncomments the currently selected element.

**Cut**
Cuts the currently selected component.

**Copy**
Copies the currently selected component.

**Delete**
Deletes the currently selected component.

**Expand More**
Expands the structure of a component in the Outline view.

**Collapse All**
Collapses the structure of all the component in the Outline view.

The upper part of the Outline view contains a filter box that allows you to focus on the relevant components. Type a text fragment in the filter box and only the components that match it are presented. For advanced usage you can use wildcard characters (such as * or ?) and separate multiple patterns with commas.

**Schematron Resource Hierarchy/Dependencies View**

The Resource Hierarchy/Dependencies view displays the hierarchy or dependencies for resources included in a Schematron schema. If the view is not displayed, it can be opened by selecting it from the Window > Show View menu.

If you want to see the hierarchy of a schema, select the desired schema in the Project view (on page 402) and choose Resource Hierarchy from the contextual menu.
If you want to see the dependencies of a schema, select the desired schema in the Project view (on page 402) and choose Resource Dependencies from the contextual menu.

The following actions are available on the toolbar of the Resource Hierarchy/Dependencies view:

- **Refresh**
  Refreshes the Hierarchy/Dependencies structure.

- **Stop**
  Stops the hierarchy/dependencies computing.

- **Show Hierarchy**
  Allows you to choose a resource to compute the hierarchy structure.

- **Show Dependencies**
  Allows you to choose a resource to compute the dependencies structure.

- **Configure dependencies search scope**
  Allows you to configure a scope to compute the dependencies structure. There is also an option for automatically using the defined scope for future operations.

- **History**
  Provides access to the list of previously computed dependencies. Use the Clear history button to remove all items from this list.
The contextual menu for a resource listed in the Resource Hierarchy/Dependencies view contains the following actions:

**Open**
- Opens the resource. You can also double-click a resource in the Hierarchy/Dependencies structure to open it.

**Go to reference**
- Opens the source document where the resource is referenced.

**Copy location**
- Copies the location of the resource.

**Move resource**
- Moves the selected resource.

**Rename resource**
- Renames the selected resource.

**Resource Hierarchy**
- Shows the hierarchy for the selected resource.

**Resource Dependencies**
- Shows the dependencies for the selected resource.

**Add to Main Files**
- Adds the currently selected resource in the Main Files directory (on page 418).

**Expand More**
- Expands more of the children of the selected resource from the Hierarchy/Dependencies structure.

**Collapse All**
- Collapses all children of the selected resource from the Hierarchy/Dependencies structure.

**Tip:**
- When a recursive reference is encountered in the Hierarchy view, the reference is marked with a special icon 🅸.

**Note:**
- The Move resource or Rename resource actions give you the option to update the references to the resource (on page 1217).
Moving/Renaming Schematron Resources

You can move and rename a resource presented in the Resource/Hierarchy Dependencies view, using the Rename resource and Move resource refactoring actions from the contextual menu.

When you select the Rename action in the contextual menu of the Resource/Hierarchy Dependencies view, the Rename resource dialog box is displayed. The following fields are available:

- **New name** - Presents the current name of the edited resource and allows you to modify it.
- **Update references of the renamed resource(s)** - Select this option to update the references to the resource you are renaming. A Preview option is available that allows you to see what will be updated before selecting Rename to process the operation.

When you select the Move action from the contextual menu of the Resource/Hierarchy Dependencies view, the Move resource dialog box is displayed. The following fields are available:

- **Destination** - Presents the path to the current location of the resource you want to move and gives you the option to introduce a new location.
- **New name** - Presents the current name of the moved resource and gives you the option to change it.
- **Update references of the moved resource(s)** - Select this option to update the references to the resource you are moving, in accordance with the new location and name. A Preview option is available that allows you to see what will be updated before selecting Move to process the operation.

Highlight Component Occurrences in Schematron Documents

When you position your mouse cursor over a component in a Schematron document, Oxygen XML Editor searches for the component declaration and all its references and highlights them automatically.

Customizable colors are used: one for the component definition and another one for component references. Occurrences are displayed until another component is selected.

To change the default behavior of Highlight Component Occurrences, open the Preferences dialog box (Options > Preferences) and go to Editor > Mark Occurrences. You can also trigger a search using the Search > Search Occurrences in File Ctrl + Shift + U (Command + Shift + U on macOS) action from contextual menu. Matches are displayed in separate tabs of the Results view.

Searching and Refactoring Operations in Schematron Documents

Search Actions

The following search actions can be applied on pattern, phase, or diagnostic types and are available from the Search submenu in the contextual menu of the current editor or from the Document > References menu:

**Search References**

Searches all references of the item found at current cursor position in the defined scope, if any. If a scope is defined, but the currently edited resource is not part of the range of resources
determined by this, a warning dialog box is displayed and you have the possibility to define another search scope.

**Search References in**

Searches all references of the item found at current cursor position in the file or files that you specify when define a scope in the **Search References** dialog box.

**Search Declarations**

Searches all declarations of the item found at current cursor position in the defined scope if any. If a scope is defined, but the currently edited resource is not part of the range of resources determined by this, a warning dialog box will be displayed and you have the possibility to define another search scope.

**Search Declarations in**

Searches all declarations of the item found at current cursor position in the file or files that you specify when you define a scope for the search operation.

**Search Occurrences in File**

Searches all occurrences of the item at the cursor position in the currently edited file.

**Refactoring Actions**

The following refactoring actions can be applied on **pattern**, **phase**, or **diagnostic** types and are available from the **Refactoring** submenu in the contextual menu of the current editor or from the **Document > Refactoring** menu:

**Rename Component**

Allows you to rename the current component (in-place). The component and all its references in the document are highlighted with a thin border and the changes you make to the component at the cursor position are updated in real time to all occurrences of the component. To exit the in-place editing, press the **Esc** or **Enter** key on your keyboard.

**Rename Component in**

Opens a dialog box that allows you to rename the selected component by specifying the new component name and the files to be affected by the modification. If you click the **Preview** button, you can view the files to be affected by the action.
Searching and Refactoring Operations Scope in Schematron Documents

The scope is a collection of documents that define the context of a search and refactor operation. To control it you can use the Change scope operation, available in the Quick Assist action set or on the Resource Hierarchy/Dependency View toolbar. You can restrict the scope to the current project or to one or multiple working sets (on page 3224). The Use only Main Files, if enabled checkbox allows you to restrict the scope of the search and refactor operations to the resources from the Main Files directory. Click read more for details about the Main Files support (on page 418).
The scope you define is applied to all future search and refactor operations until you modify it. Contextual menu actions allow you to add or delete files, folders, and other resources to the working set (on page 3224) structure.

**Quick Assist Support in Schematron Documents**

The *Quick Assist support* (on page 3223) improves the development work flow, offering fast access to the most commonly used actions when you edit schema documents.

The *Quick Assist feature* (on page 3223) is activated automatically when the cursor is positioned over the name of a component. It is accessible via a yellow bulb icon (✔️) placed at the current line in the stripe on the left side of the editor. Also, you can invoke the quick assist menu by using the **Alt + 1** (**Meta + Alt + 1** on macOS) keyboard shortcuts.

The *Quick Assist* support offers direct access to the following actions:
Renamed Component in

Renames the component and all its dependencies.

Search Declarations

Searches the declaration of the component in a predefined scope. It is available only when the context represents a component name reference.

Search References

Searches all references of the component in a predefined scope.

Component Dependencies

Searches the component dependencies in a predefined scope.

Change Scope

Configures the scope that will be used for future search or refactor operations.

Rename Component

Allows you to rename the current component in-place.

Search Occurrences

Searches all occurrences of the component within the current file.

Schematron Unit Test (XSpec)

XSpec is a behavior driven development (BDD) framework for XSLT, XQuery, and Schematron. XSpec consists of syntax for describing the behavior of your XSLT, XQuery, or Schematron code, and some code that enables you to test your code against those descriptions.

Creating a Schematron Unit Test

To create a Schematron Unit Test, go to File > New > Schematron Unit Test. This is simple document template to help you get started.

Running a Schematron Unit Test

To run a Unit Test, open the XSpec file in an editor and click Apply Transformation Scenario(s) on the main toolbar. This will run the built-in Run XSpec Test transformation scenario that is defined in the XSpec framework (on page 3219).

Testing a Stylesheet

An XSpec file contains one or more test scenarios.

Example

Suppose you have this Schematron rule that says sections should have a title:
The XSpec test could look like this:

```xml
<x:description xmlns:x="http://www.jenitennison.com/xslt/xspec" schematron="demo-01.sch">
  <x:scenario label="section should have a title">
    <x:context>
      <article>
        <section>
          <title>Introduction</title>
          <p>This is an example.</p>
        </section>
        <section>
          <p>This is an example.</p>
        </section>
      </article>
    </x:context>
    <x:expect-not-assert id="a002" location="/article[1]/section[1]"/>
    <x:expect-assert id="a002" location="/article[1]/section[2]"/>
  </x:scenario>
</x:description>
```

The `<sch:assert>` with the `id="a002"` is not expected to be triggered on the first section since it includes a title. This requirement is expressed with the `<x:expect-not-assert>` element.

Since the second section does not have a title, you would expect the Schematron rule to be triggered and this requirement is expressed with the `<x:expect-assert>` element.

For more details about how to write Schematron tests and various samples, see https://github.com/xspec/xspec/wiki/Writing-Scenarios-for-Schematron#writing-tests.

### Adding a Catalog to an XSpec Transformation

If your Schematron needs a catalog, you can add one to the XSpec transformation by doing one of the following:
• If you are using a project (on page 398) in Oxygen XML Editor, create a catalog.xml file in the project directory. This catalog will then be loaded automatically.

• Edit (on page 1545) the Run XSpec Test transformation scenario, go to the Parameters tab (on page 1509), and set the value of the catalog parameter to the location of your catalog file.

Editing Schematron Quick Fixes

Oxygen XML Editor provides support for editing the Schematron Quick Fixes (on page 815) (SQF). They help you resolve issues that appear in XML documents that are validated against Schematron schemas by offering you solution proposals. The Schematron Quick Fixes are an extension of the Schematron language and they allow you to define fixes for Schematron validation messages. Specifically, they are associated with assert or report messages. You can define a library of Quick Fixes by editing them directly in the current Schematron file or in a separate file. Oxygen XML Editor assists you in editing Schematron Quick Fixes with schema-based content completion, syntax highlighting, and validation as you type.

A typical use case is using Schematron Quick Fixes to assist content authors with common editing tasks. For example, you can use Schematron rules to automatically report certain validation warnings (or errors) when performing regular editing tasks, such as inserting specific elements or changing IDs to match specific naming conventions.

For information about applying and detecting the Schematron schemas that include SQF, see Associating a Schema to XML Documents (on page 816).

Displaying the Schematron Quick Fix Proposals

The defined Schematron Quick Fixes are displayed on validation errors in Text mode and Author mode.

Figure 447. Example of a Schematron Quick Fix

<table>
<thead>
<tr>
<th>Element</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;head&gt;</td>
<td>The &quot;title&quot; element is missing.</td>
</tr>
<tr>
<td></td>
<td>Insert title element.</td>
</tr>
<tr>
<td></td>
<td>Insert &quot;title&quot; element with H1 value</td>
</tr>
<tr>
<td></td>
<td>Insert the title element as child. Set the value of the &quot;title&quot; element to the value of &quot;h1&quot; element.</td>
</tr>
</tbody>
</table>

Related Information:

Oxygen XML Blog: Schematron Checks to Help Technical Writing

Schematron Quick Fix Specifications

Examples of Schematron Rules and Quick Fixes

This topic is meant to provide some basic examples of Schematron Rules and Schematron Quick Fixes (SQF) to help you create and impose your own rules and quick fixes.

Other examples and ideas can also be found at:
Schematron Examples

Schematron Use Case 1: Impose a Relax NG Schema Declaration

**Description:** The following sample rule is useful if, for example, you need to enforce the use of Relax NG schema declarations in all of your documents (i.e. instead of using DTD schemas).

**Sample Code:**

```xml
<?xml version="1.0" encoding="UTF-8"?>
<sch:schema xmlns:sch="http://purl.oclc.org/dsdl/schematron"
queryBinding="xslt2" xmlns:saxon="http://saxon.sf.net/"
>
  <sch:let name="rngDeclaration"
    value="processing-instruction('xml-model')
      [saxon:get-pseudo-attribute('schematypens')='http://relaxng.org/ns/structure/1.0']"/>

  <sch:pattern>
    <sch:rule context="/element()">
      <sch:assert test="exists($rngDeclaration)"/>
      You must define a Relax NG schema declaration in the document (DTD schemas are not supported).
    </sch:rule>
  </sch:pattern>
</sch:schema>
```

**Result:** The engine checks for a Relax NG schema declaration in the document and displays an error if it is missing. The error is reported on the document's root element (`/element()`).

Schematron Use Case 2: Check for Missing IDs

**Description:** The following sample rule checks for missing or undefined IDs in a TEI document. Specifically, it looks for IDs from the `tei:rs/@ref` attribute defined in the document named `persons.xml` (as `xml:id` of a TEI `person` element).

**Sample Code:**

```xml
<?xml version="1.0" encoding="UTF-8"?>
<sch:schema xmlns:sch="http://purl.oclc.org/dsdl/schematron" queryBinding="xslt2">
  <sch:ns uri="http://www.tei-c.org/ns/1.0" prefix="tei"/>
  <sch:let name="personIds"
    value="document('..//persons.xml')/tei:TEI//tei:person/@xml:id"/>

  <sch:pattern>
    <sch:rule context="tei:rs">
      <sch:let name="refIds"
        value="for $id in tokenize(@ref, ' ') return substring-after($id, '#')"/>

```
where the XML document looks something like this:

```xml
<tei xmlns="http://www.tei-c.org/ns/1.0">
  <rs ref="../SomePerson/persons.xml#EDP ../personography/HAMpersons.xml#SD">text</rs>
  <rs ref="../SomePerson/persons.xml#EDP">text</rs>
</tei>
```

**Result:** The engine displays an error message listing the missing/undefined IDs.

### Schematron Use Case 3: Check for Broken Links

**Description:** The following sample rule detects broken links in DITA `<xref>` or `<link>` elements. The first example only checks links that do not contain an anchor (#).

**Sample Code:**

```xml
<rule context="*[contains(@class, ' topic/xref ') or contains(@class, ' topic/link ')]
  [@href][not(contains(@href, '#'))][not(@scope = 'external')]
  [not(@type) or @type='dita']">
  <assert test="doc-available(resolve-uri(@href, base-uri(.)))">
    The document linked by `<value-of select="local-name()"/>
    `<value-of select="@href"/> does not exist!</assert>
</rule>
```

For links that contain an anchor, the Schematron rule must look something like this:

```xml
<rule context="*[contains(@class, ' topic/xref ') or contains(@class, ' topic/link ')]
  [@href][contains(@href, '#')][not(@scope = 'external')]
  [not(@type) or @type='dita']">
  <let name="file" value="substring-before(@href, '#')"/>
  <let name="idPart" value="substring-after(@href, '#')"/>
  <let name="topicId"
    value="if (contains($idPart, '/') then substring-before($idPart, '/') else $idPart")/>
  <let name="id" value="substring-after($idPart, '/')"/>
```
<assert test="document($file, .)//*[@id=$topicId]">
  Invalid topic id "<value-of select="$topicId"/>"  </assert>

<assert test="$id ='' or document($file, .)//*[@id=$id]">
  No such id "<value-of select="$id"/>" is defined!  </assert>

<assert test="$id='' or document($file, .)//*[@id=$id]
[ancestor::*[contains(@class, ' topic/topic ')][1][@id=$topicId]]">
  The id "<value-of select="$id"/>" is not in the scope of the referenced topic id "<value-of select="$topicId"/>".  </assert>
</rule>

Result: The engine displays an error message when a broken link or cross reference is detected.

Schematron Use Case 4: Check for Duplicate IDs

Description: The following sample rule detects if there are two sibling <step> elements with the same @id value in a DITA Task document.

Sample Code:

```xml
<sch:rule context="*[contains(@class, ' task/step ')]">
  <sch:let name="id" value="@id"/>
  <sch:report
    test="preceding-sibling::element(){contains(@class, ' task/step ')}[@id = $id]">
    Element with duplicate ID "<sch:value-of select="$id"/>" detected.
  </sch:report>
</sch:rule>
```

Result: The engine displays an error message when a duplicate ID is detected in sibling <step> elements within a DITA Task document.

Schematron Use Case 5: Check for Duplicate DITA Topic References

Description: The following sample rule checks a DITA map for duplicate <topicref> elements with the same @href value.

Sample Code:

```xml
<sch:rule context="*[contains(@class, ' map/topicref ')]">
  <sch:let name="href" value="@href"/>
  <sch:report
    test="preceding::element(){contains(@class, ' map/topicref ')}[@href = $href]">
    Duplicate topicref "<sch:value-of select="$href"/>" detected in map.
  </sch:report>
</sch:rule>
```
Result: The engine displays an error message when multiple `<topicref>` elements with the same `@href` value are detected in a DITA map.

Schematron Use Case 6: Restrict Certain Words from the Title

Description: The following sample rule checks for instances of specified words to be restricted from a `<title>` element (in this example, the words `test` and `hello` are restricted).

Sample Code:

```xml
<sch:schema xmlns:sch="http://purl.oclc.org/dsdl/schematron"
            queryBinding="xslt2">
    <sch:let name="words" value="test,hello'/">
    <sch:let name="wordsToMatch" value="replace($words, ',', '|')"/>
    <sch:pattern>
        <sch:rule context="title">
            <sch:report test="matches(text(), $wordsToMatch)" role="warn">
                The following words should not be added in the title:
            </sch:report>
        </sch:rule>
    </sch:pattern>
</sch:schema>
```

Result: The engine displays an error message if one of the specified restricted words appear in a title.

Schematron Use Case 7: Check the Location of a Resource

Description: The following sample rule checks if the path to a resource (in this case, an image) is specified correctly. Specifically, this sample rule reports that the image must be located in the current project (the images location must be relative to the parent folder and no more than one `../` in the path).

Sample Code:

```xml
<sch:rule context="image">
    <sch:report test="count(tokenize(@href, '\.\./')) > 2">
        The image must be located in the current project. It is currently located in: <sch:value-of select="@href"/>
    </sch:report>
</sch:rule>
```

Result: The engine displays an error message if an image is detected in a location other than the current project, relative to the parent folder.

Schematron Use Case 8: Check for Extra Spaces at Beginning/End of Elements

Description: The following sample rule checks for spaces at the beginning and end of elements.
Tip:
You could specify a list of elements to check to make the rule context-sensitive.

Sample Code:

```xml
<rule context="p|ph|codeph|filename|indexterm|xref|user-defined|user-input">
  <let name="firstNodeIsElement" value="node()[1] instance of element()"/>
  <let name="lastNodeIsElement" value="node()[last()] instance of element()"/>
  <report test="(not($firstNodeIsElement) and matches(.,'^\s',';j'))
              or (not($lastNodeIsElement) and matches(.,'\s$',';j'))"
          role="warning">
    Textual elements should not begin or end with whitespace.</report>
</rule>
```

Result: The engine displays an error message if a whitespace is detected at the beginning or end of a textual element.

Schematron Use Case 9: Impose Capitalizing the First Letter

Description: The following sample rule detects if elements start with a capital letter or a number. The rule is implemented using abstract patterns. The abstract pattern `starts-with-capital` has one argument representing the element to be checked. There are two implementations of the abstract pattern, one that specifies the `<title>` element as the element to verify, and one that specifies the `<li>` element.

Sample Code:

```xml
<sch:pattern abstract="true" id="starts-with-capital">
  <sch:rule context="$element" role="information">
    <sch:let name="firstNodeIsElement" value="node()[1] instance of element()"/>
    <sch:report test="(not($firstNodeIsElement) and (not(matches(.,'^[A-Z|0-9]'))))">
      Start the element &lt;$element&gt; with a capital letter.</sch:report>
  </sch:rule>
</sch:pattern>

<sch:pattern is-a="starts-with-capital">
  <sch:param name="element" value="title"/>
</sch:pattern>

<sch:pattern is-a="starts-with-capital">
  <sch:param name="element" value="li"/>
</sch:pattern>
```

Result: The engine displays an error message if a title begins with a word that does not contain a capital letter or number as its first character.
Schematron Use Case 10: Check for Specified Terms in a Paragraph

**Description:** The following sample rule checks if any DITA `<p>` elements contain certain keywords defined in an external document.

**Sample Code:**

```xml
<sch:pattern>
    <sch:let name="keys" value="document('keys-common.ditamap')//keyword"/>
    <sch:rule context="p">
        <sch:let name="text" value="."/>
        <sch:let name="matchedKeys" value="$keys[contains($text, normalize-space(.))]">
            The paragraph text contains the keywords: $matchedKeys
        </sch:let>
        <sch:report id="now001" test="count($matchedKeys) > 0" role="error">
            The paragraph text contains the keywords: $matchedKeys
        </sch:report>
    </sch:rule>
</sch:pattern>
```

**Result:** The engine displays an error message if any of the keywords listed in an external document are detected within a DITA `<p>` element.

Schematron Use Case 11: Impose a Minimum Value

**Description:** The following sample rule determines the `<type>` element value with the minimum version specified by the `@version` attribute and then verifies that they are all equal to the determined value.

**Sample Code:**

```xml
<sch:schema xmlns:sch="http://purl.oclc.org/dsdl/schematron" queryBinding="xslt2">
    <sch:let name="typeValue" value="/Node1[not(@version > ../Node1/@version)]/Type/text()"/>

    <sch:pattern>
        <sch:rule context="Type">
            <sch:assert test="text() = $typeValue">
                The Type value must be "$typeValue"
            </sch:assert>
        </sch:rule>
    </sch:pattern>
</sch:schema>
```

where the XML file would look something like this:

```xml
<root>
    <Node1 version="1">
        <Element1>Value1</Element1>
        <Type>123456</Type>
    </Node1>
</root>
```
Result: The engine displays an error message if a `<type>` element value does not equal the minimum version specified by the `@version` attribute.

**SQF (Schematron Quick Fix) Examples**

**SQF Use Case 1: Impose a DITA Prolog**

**Description:** The following sample Schematron rule checks a DITA topic to make sure it contains `<prolog>`, `<critdates>`, `<revised>` elements and the sample Quick Fix proposes options for inserting the missing elements.

**Sample Code:**

```xml
<schema xmlns:sch="http://purl.oclc.org/dsdl/schematron" queryBinding="xslt2"
   xmlns:sqf="http://www.schematron-quickfix.com/validator/process">
  <pattern>
    <rule context="*[contains(@class, ' topic/topic ')]">
      <assert sqf:fix="add_prolog" test="prolog" role="warn">
        Every topic must contain prolog/critdates/revised elements where the revised modified date is in YYYY-MM-DD format.</assert>
      <fix id="add_prolog">
        <title>Add prolog/critdates/revised elements, where the revised element's @modified attribute value is the current date in YYYY-MM-DD format.</title>
        <description>
          Add prolog/critdates/revised elements, where the revised element's @modified attribute value is the current date in YYYY-MM-DD format.</description>
        <add match="*[contains(@class, ' topic/body ')]" node-type="element" position="before" target="prolog">
          <critdates>
            <revised modified=""></revised>
          </critdates>
        </add>
      </fix>
    </rule>
  </pattern>
</schema>
```
<sch:rule context="*[contains(@class, ' topic/prolog ')]">
  <sch:report role="warn" test="not(critdates)" sqf:fix="add_critdates">The prolog element must have critdates/revised elements with the @modified attribute value in YYYY-MM-DD format.</sch:report>
  <sqf:fix id="add_critdates">
    <sqf:description>
      <sqf:title>Add the critdates element.</sqf:title>
    </sqf:description>
    <sqf:add node-type="element" target="critdates">
      <revised modified=""></revised>
    </sqf:add>
  </sqf:fix>
</sch:rule>

<sch:rule context="*[contains(@class, ' topic/critdates ')]">
  <sch:report role="warn" test="not(revised)" sqf:fix="add_revised">The critdates element must have revised @modified in YYYY-MM-DD format.</sch:report>
  <sqf:fix id="add_revised">
    <sqf:description>
      <sqf:title>Add the revised element.</sqf:title>
    </sqf:description>
    <sqf:add node-type="element" target="revised"/>
  </sqf:fix>
</sch:rule>

Result: The engine displays an error message if the `<prolog>`, `<critdates>`, or `<revised>` elements are missing from a DITA topic and the Quick Fix mechanism proposes options for inserting the missing elements.

**SQF Use Case 2: Impose an ID for all DITA Section Elements**

**Description:** The following sample Schematron rule checks if each DITA `<section>` element has a specified ID and the sample Quick Fix proposes options for inserting the missing IDs.

**Sample Code:**

```xml
<sch:schema xmlns:sch="http://purl.oclc.org/dsdl/schematron"
  xmlns:sqf="http://www.schematron-quickfix.com/validator/process">
  <sch:pattern>
    <!-- Add IDs to all sections to impose link targets -->
    <sch:rule context="section">
      <sch:assert test="$id" sqf:fix="addId addIds">[Bug] All sections should have an @id attribute</sch:assert>
```
<sqf:fix id="addId">
  <sqf:description>
    <sqf:title>Add @id to the current section</sqf:title>
    <sqf:p>Add an @id attribute to the current section. The ID is generated from the section title.</sqf:p>
  </sqf:description>
</sqf:fix>

<!-- Generate an id based on the section title. If there is no title then generate a random id. -->
 sqf:add target="id" node-type="attribute"
  select="
    concat('section_,'
    if (exists(title) and string-length(title) > 0)
    then
      substring(lower-case(replace(replace(
        normalize-space(string(title)), '\s', '_'),
      '^[a-zA-Z0-9-\_]', '',)), 0, 50)
    else generate-id())"/
</sqf:fix>

<sqf:fix id="addIds">
  <sqf:description>
    <sqf:title>Add @id to all sections</sqf:title>
    <sqf:p>Add an @id attribute to each section from the document. The ID is generated from the section title.</sqf:p>
  </sqf:description>
</sqf:fix>

<!-- Generate an id based on the section title. If there is no title then generate a random id. -->
 sqf:add match="/section[not(@id)]" target="id" node-type="attribute"
  select="
    concat('section_,'
    if (exists(title) and string-length(title) > 0)
    then substring(lower-case(replace(replace(
      normalize-space(string(title)), '\s', '_'),
    '^[a-zA-Z0-9-\_]', '',)), 0, 50)
    else generate-id())"/>
</sqf:fix>
</sch:rule>
</sch:pattern>
</sch:schema>

**Result:** The engine displays an error message if an @id attribute is missing for any `<section>` element in a DITA topic and the Quick Fix mechanism proposes options for inserting the missing ID.
SQF Use Case 3: Impose a Short Description in an Abstract Element

**Description:** The following sample Schematron rule checks a DITA topic to make sure it contains a `<shortdesc>` element inside an `<abstract>` element and the sample Quick Fix proposes options for correcting the missing structure.

**Sample Code:**

```xml
<sch:schema xmlns:sch="http://purl.oclc.org/dsdl/schematron" queryBinding="xslt2"
xmlns:sqf="http://www.schematron-quickfix.com/validator/process"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <sch:pattern>
    <sch:rule context="shortdesc">
      <sch:assert test="parent::abstract" sqf:fix="moveToAbstract moveToExistingAbstract">
        The short description must be added in an abstract element
      </sch:assert>
    </sch:rule>
    <!-- Check if there is an abstract element -->
    <sch:let name="abstractElem" value="preceding-sibling::abstract |
    following-sibling::abstract"/>
    <!-- Create an abstract element and add the short description -->
    <sqf:fix id="moveToAbstract" use-when="not($abstractElem)">
      <sqf:description>
        <sqf:title>Move short description in an abstract element</sqf:title>
      </sqf:description>
      <sqf:replace>
        <abstract>
          <xsl:apply-templates mode="copyExceptClass" select="."/>
        </abstract>
      </sqf:replace>
    </sqf:fix>
    <!-- Move the short description in the abstract element-->
    sqf:fix id="moveToExistingAbstract" use-when="$abstractElem"?
    <sqf:description>
      <sqf:title>Move short description in the abstract element</sqf:title>
    </sqf:description>
    <sch:let name="shortDesc">
      <xsl:apply-templates mode="copyExceptClass" select="."/>
    </sch:let>
    <sqf:add match="$abstractElem" select="$shortDesc"/>
    <sqf:delete/>
  </sch:pattern>
</sch:schema>
```
<xsl:template match="node() | @*" mode="copyExceptClass">
  <xsl:copy copy-namespaces="no">
    <xsl:apply-templates select="node() | @*" mode="copyExceptClass"/>
  </xsl:copy>
</xsl:template>

<!-- Template used to skip the @class attribute from being copied -->
<xsl:template match="@class" mode="copyExceptClass"/>

</xsl:schema>

Result: The engine displays an error message if an <abstract> element does not contain a <shortdesc> element and the Quick Fix mechanism proposes options for inserting the missing structure or to move the <shortdesc> element inside the <abstract> element.

SQF Use Case 4: Impose a Certain Article Type

Description: The following sample Schematron rule checks the @article-type attribute to make sure its value is one of the specified allowed values (abstract, addendum, announcement, article-commentary) and the sample Quick Fix proposes options for replacing any other detected value with one of the allowed values.

Sample Code:

```xml
<sch:schema xmlns:sch="http://purl.oclc.org/dsdl/schematron" queryBinding="xslt2"
  xmlns:sqf="http://www.schematron-quickfix.com/validator/process">
  <sch:let name="articleTypes" value="('abstract', 'addendum', 'announcement',
      'article-commentary')="/"
  
  <sch:rule context="article/@article-type">
    <sch:assert test=". = $articleTypes" sqf:fix="setArticleType">
      Should be one of the article types:
      <sch:value-of select="$articleTypes"/>
    </sch:assert>

    <sqf:fix id="setArticleType" use-for-each="$articleTypes">
      <sqf:description>
        <sqf:title>Set article type to '<<sch:value-of select="$sqf:current"'/>'
      </sqf:title>
    </sqf:description>

    <sqf:replace node-type="attribute" target="article-type" select="$sqf:current"/>
  </sqf:fix>
</sch:rule>
```
Result: The engine displays an error message if an @article-type attribute has any other value other than abstract, addendum, announcement, or article-commentary and the Quick Fix mechanism proposes options for replacing the disallowed value with one of those four allowed values (using the use-for-each construct).

SQF Use Case 5: Impose Certain Attributes and Values

Description: The following sample Schematron rule checks the @rowsep and @colsep attributes are added on the <table> element and their value is set to 1. The Quick Fix proposes options for adding the attributes in case they are missing or set the correct value.

Sample Code:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<sch:schema xmlns:sch="http://purl.oclc.org/dsdl/schematron" queryBinding="xslt2"
    xmlns:sqf="http://www.schematron-quickfix.com/validator/process">
    <sch:rule context="table">
        <sch:assert test="@rowsep = 1" sqf:fix="addRowsep">The @rowsep should be set to 1</sch:assert>
        <sch:assert test="@colsep = 1" sqf:fix="addColsep">The @colsep should be set to 1</sch:assert>
        <sqf:fix id="addRowsep">
            <sqf:description>
                <sqf:title>Add @rowsep attribute</sqf:title>
            </sqf:description>
            <sqf:add node-type="attribute" target="rowsep" select="'1'"/>
        </sqf:fix>
        <sqf:fix id="addColsep">
            <sqf:description>
                <sqf:title>Add @colsep attribute</sqf:title>
            </sqf:description>
            <sqf:add node-type="attribute" target="colsep" select="'1'"/>
        </sqf:fix>
    </sch:rule>
</sch:pattern>
</sch:schema>
```
Defining Schematron Quick Fixes

You can define and customize Schematron Quick Fixes (on page 3223) directly in the current Schematron file or in a separate Schematron file. The Schematron Quick Fixes are an extension of the Schematron language and they allow you to define fixes for Schematron error messages. You can reference the Quick Fixes using the @sqf:fix attribute inside the <assert> or <report> elements (for example: <assert test="title" sqf:fix="removeComments">Remove comments</assert>).

Defining a Schematron Quick Fix

The basics of a Schematron Quick Fix is defined by an ID, name, description, and the operations to be executed.

- **ID** - Defined by the @id attribute from the <sqf:fix> element and must be unique in the current context. It is used to refer the Quick Fix from a <report> or <assert> element.
- **Name** - The name of the Quick Fix is defined by the <sqf:title> element.
- **Description** - Defined by the text in the paragraphs (<sqf:p>) of the <sqf:description> element.
- **Operations** - The following basic types of operations (elements) (on page 1237) are supported:
  - <sqf:add> Element - To add a new node or fragment in the document.
  - <sqf:delete> Element - To remove a node from the document.
  - <sqf:replace> Element - To replace a node with another node or fragment.
  - <sqf:stringReplace> Element - To replace text content with other text or a fragment.

Figure 448. Schematron Quick Fix Components

The assertion message that generates the Quick Fix is added as the <sqf:description> of the problem to be fixed. The <sqf:title> is presented as the name of the Quick Fix. The content of the paragraphs (<sqf:p>) within the <sqf:description> element are presented in the tooltip message when the Quick Fix is selected.

Additional Elements Supported in the Schematron Quick Fixes

<sqf:user-entry>
This element defines a value that must be set manually by the user. For more information, see User Entry SQF Operation (on page 1242).

<sqf:call-fix>

This element calls another Quick Fix within a Quick Fix. The called Quick Fix must be defined globally or in the same Schematron rule as the calling Quick Fix. A calling Quick Fix adopts the activity elements of the called Quick Fix and should not include other activity elements. You can also specify which parameters are sent by using the <sqf:with-param> child element.

<sqf:group>

Allows you to group multiple Quick Fixes and refer them from an <assert> or <report> element.

<sqf:fixes>

Is defined globally and contains global fixes and groups of fixes.

<sqf:copy-of>

Used to copy the selected nodes that are specified by the @select attribute. The element with its attribute is treated as an xsl:copy-of with a @select attribute, as defined in the XSLT specification.

<sqf:param>

Defines a parameter for a Quick Fix. If the parameter is defined as abstract then the type and default value should not be specified and the fix can be called from an abstract pattern that defines this parameter.

Other SQF Notes

Note:
The sqf:default-fix attribute is ignored in Oxygen XML Editor.

For more details on editing Schematron Quick Fixes, go to: Schematron Quick Fix Specifications

Basic Schematron Quick Fix Operations

There are four basic operations that can be executed in a Schematron Quick Fix (on page 3223): Add, Delete, Replace, and String Replace.

Add

The <sqf:add> element allows you to add a node to the instance. An anchor node is required to select the position for the new node. The anchor node can be selected by the @match attribute. Otherwise, it is selected by the @context attribute of the rule.

The @target attribute defines the name of the node to be added. It is required if the value of the @node-type attribute is set to anything other than "comment".
The `<sqf:add>` element has a `@position` attribute and it determines the position relative to the anchor node. The new node could be specified as the first child of the anchor node, the last child of the anchor node, before the anchor node, or after the anchor node (`first-child` is the default value). If you want to add an attribute to the anchor node, do not use the `@position` attribute.

**Note:**
If you insert an element and its content is empty, Oxygen XML Editor will insert the required element content.

### An Example of the `<sqf:add>` Element:

```xml
<schema xmlns="http://purl.oclc.org/dsdl/schematron"
    xmlns:sqf="http://www.schematron-quickfix.com/validator/process"
    queryBinding="xslt2">
  <pattern>
    <rule context="head">
      <assert test="title" sqf:fix="addTitle">title element missing.</assert>
      <sqf:fix id="addTitle">
        <sqf:description>
          <sqf:title>Insert title element.</sqf:title>
        </sqf:description>
        <sqf:add target="title" node-type="element">Title text</sqf:add>
      </sqf:fix>
    </rule>
  </pattern>
</schema>
```

### Specific Add Operations:

- **Insert Element** - To insert an element, use the `<sqf:add>` element, set the value of the `@node-type` attribute as "element", and specify the element `QName (on page 3223)` with the `@target` attribute. If the element has a prefix, it must be defined in the Schematron using a namespace declaration (`<ns uri="namespace" prefix="prefix"/>`).
- **Insert Attribute** - To insert an attribute, use the `<sqf:add>` element, set the value of the `@node-type` attribute as "attribute", and specify the attribute `QName (on page 3223)` with the `@target` attribute. If the attribute has a prefix, it must be defined in the Schematron using a namespace declaration (`<ns uri="namespace" prefix="prefix"/>`).
- **Insert Fragment** - If the `@node-type` attribute is not specified, the `<sqf:add>` element will insert an XML fragment. The XML fragment must be well-formed. You can specify the fragment in the `<sqf:add>` element or by using the `@select` attribute.
• **Insert Comment** - To insert a comment, use the `<sqf:add>` element and set the value of the `@node-type` attribute as "comment".

• **Insert Processing Instruction** - To insert a processing instruction, use the `<sqf:add>` element, set the value of the `@node-type` attribute as "pi" or "processing-instruction", and specify the name of the processing instruction in the `@target` attribute.

**Delete**

The `<sqf:delete>` element allows you to remove any type of node (such as elements, attributes, text, comments, or processing instructions). To specify nodes for deletion, the `<sqf:delete>` element can include a `@match` attribute that is an XPath expression (the default value is `/`). If the `@match` attribute is not included, it deletes the context node of the Schematron rule.

**An Example of the `<sqf:delete>` Element:**

```xml
<schema xmlns="http://purl.oclc.org/dsdl/schematron" queryBinding="xslt2"

  xmlns:sqf="http://www.schematron-quickfix.com/validator/process">

  <pattern>
    <rule context="*[@xml:lang]">
      <report test="@xml:lang" sqf:fix="remove_lang">
        The attribute "xml:lang" is forbidden.</report>
      <sqf:fix id="remove_lang">
        <sqf:description>
          <sqf:title>Remove "xml:lang" attribute</sqf:title>
        </sqf:description>
      </sqf:fix>
      <sqf:delete match="@xml:lang"/>
    </rule>
  </pattern>

</schema>
```

**Replace**

The `<sqf:replace>` element allows you to replace nodes. Similar to the `<sqf:delete>` element, it can include a `@match` attribute. Otherwise, it replaces the context node of the rule. The `<sqf:replace>` element has three tasks. It identifies the nodes to be replaced, defines the replacing nodes, and defines their content.

**An Example of the `<sqf:replace>` Element:**

```xml
<schema xmlns="http://purl.oclc.org/dsdl/schematron"

  xmlns:sqf="http://www.schematron-quickfix.com/validator/process"

queryBinding="xslt2">

  <pattern>
    <rule context="title">
      <report test="exists(ph)" sqf:fix="resolvePh" role="warn">
        
      </report>
    </rule>
  </pattern>

</schema>
```
Other Attributes for Replace Operations:

- **node-type** - Determines the type of the replacing node. The permitted values include:
  - **keep** - Keeps the node type of the node to be replaced.
  - **element** - Replaces the node with an element.
  - **attribute** - Replaces the node with an attribute.
  - **pi** - Replaces the node with a processing instruction.
  - **comment** - Replaces the node with a comment.

- **target** - By using a `QName` (on page 3223) it gives the replacing node a name. This is necessary when the value of the `@node-type` attribute is anything other than "comment".

- **select** - Allows you to choose the content of the replacing nodes. You can use XPath expressions with the `@select` attribute. If the `@select` attribute is not specified then the content of the `<sqf:replace>` element is used instead.

String Replace

The `<sqf:stringReplace>` element is different from the others. It can be used to find a sub-string of text content and replace it with nodes or other strings.

Attributes for the String Replace Operation:

- **match** - Allows you to select text nodes that contain the sub-strings you want to replace.
- **select** - Allows you to select the replacing fragment, in case you do not want to set it in the content of the `<stringReplace>` element.
- **regex** - Matches the sub-strings using a regular expression.

Note:

Consider the following information about using regular expressions in the `<stringReplace>` element:
The regular expressions from this operation are compiled as Java regular expressions. For more information, see https://docs.oracle.com/javase/7/docs/api/java/util/regex/Pattern.html.

The \textit{j} flag allows you to use the standard Java regular expression engine, which allows native Java regular expression syntax. This allows, for example, the use of $\backslash \texttt{b}$ in a regular expression to match word boundaries. For more information, see https://www.saxonica.com/html/documentation/functions/fn/matches.html.

Regular expressions in the \texttt{<sqf:stringReplace>} element always have the \textit{dot matches all} flag set to "true". Therefore, the line terminator will also be matched by the regular expression.

- **flags** - Specifies flags to control the interpretation of the regular expression (given in the \texttt{@regex} attribute).

\textbf{Attention:}

The context of the content within the \texttt{<sqf:stringReplace>} element is set to the whole text node, rather than the current sub-string.

\textbf{An Example of the \texttt{<sqf:stringReplace>} Element:}

```xml
<?xml version="1.0" encoding="UTF-8"?>
<sch:schema xmlns:sch="http://purl.oclc.org/dsdl/schematron"
           xmlns:sqf="http://www.schematron-quickfix.com/validator/process"
           queryBinding="xslt2">

  <sch:pattern>
    <sch:rule context="text()">
      <sch:report test="matches(., 'Oxygen', 'i')">
        sqf:fix="changeWord">The oXygen word is not allowed</sch:report>
        <sqf:fix id="changeWord">
          <sqf:description>
            <sqf:title>Replace word with product</sqf:title>
          </sqf:description>
          <sqf:stringReplace regex="Oxygen" flags="i"><ph keyref="product"/>
        </sqf:stringReplace>
      </sqf:fix>
    </sch:report>
  </sch:rule>
</sch:pattern>
</sch:schema>
```
Related Information:
User Entry SQF Operation (on page 1242)
Restricting Quick Fix Operations (on page 1242)
Examples of Schematron Rules and Quick Fixes (on page 1194)

User Entry SQF Operation

The `<sqf:user-entry>` element defines a value that must be set manually by the user. If multiple `<user-entry>` elements are defined, Oxygen XML Editor will display a dialog box for each one where the user can specify values. Also, the `<user-entry>` element can be used as an XPath variable where the XPath variable is the name of the `<user-entry>`. Note that the `@default` attribute defines a default value for the operation by using an XPath expression (as in the example below) and its value will be presented in the user entry dialog box.

An Example of the `<sqf:user-entry>` Element:

```xml
<sqf:fix id="editTitle">
  <sqf:description>
    <sqf:title>Edit the journal title</sqf:title>
  </sqf:description>
  <sqf:user-entry name="newTitle" default="@title">
    <sqf:title>Edit the title:</sqf:title>
  </sqf:description>
</sqf:user-entry>
  <sqf:replace match="@title" target="title" node-type="keep" select="$newTitle"/>
</sqf:fix>
```

Restricting Quick Fix Operations

To restrict a Quick Fix (on page 3223) or a specific operation to only be available if certain conditions are met, the `@use-when` attribute can be included in the `<sqf:fix>` element or any of the SQF operation elements. The condition of the `@use-when` attribute is an XPath expression and the fix or operation will be performed only if the condition is satisfied. In the following example, the `use-when` condition is applied to the `<sqf:fix>` element:

```xml
<sqf:fix id="last" use-when="$colWidthSummarized - 100 lt $lastWidth" role="replace">
  <sqf:description>
    <sqf:title>Subtract excessive width from the last element.</sqf:title>
  </sqf:description>
</sqf:fix>
```
<let name="delta" value="colWidthSummarized - 100"/>
<sqf:add match="html:col[last()]" target="width" node-type="attribute">
    <let name="newWidth" value="number(substring-before(@width,'%')) - $delta"/>
    <value-of select="concat($newWidth,'%')"/>
</sqf:add>
</sqf:fix>

Related Information:
Basic Schematron Quick Fix Operations (on page 1237)

Formatting/Indenting Content Inserted by SQF Operations

Content that is inserted by the Add, Replace, or String Replace Schematron Quick Fix (on page 3223) operations is automatically indented unless you set the value of the @xml:space attribute to preserve on the operation element. There are several methods available to format the content that is inserted:

- **xsl:text** - You can use an `<xsl:text>` element to format the inserted content and keep the automatic indentation, as in the following example:

```xml
<sqf:add position="last-child">
    <row><xsl:text>
        <entry>First column</entry><xsl:text>
    </row>
    <row><xsl:text>
        <entry>Second column</entry><xsl:text>
    </row>
</sqf:add>
```

- **xml:space** - Use the `@xml:space` attribute and set its value to preserve to format the content and specify the spacing between elements, as in the following example:

```xml
<sqf:add node-type="element" target="codeblock" xml:space="preserve">
    /* a long sample program */
    Do forever
    Say "Hello, World"
    End<sqf:add>
```

Related Information:
Basic Schematron Quick Fix Operations (on page 1237)
Executing Schematron Quick Fixes in Other Documents

You can apply Schematron Quick Fixes (on page 3223) over nodes from referenced documents (using XInclude or external entities), and you can access them as nodes in your current document.

Also, you can apply the Quick Fixes over other documents using the doc() function in the value of the @match attribute. For example, you can add a new <Key> element with the value of newVal as the last child in the <KeyList> element from the keylist.xml file using the following operation:

```
<sqf:add match="doc('keylist.xml')/KeyList" target="Key" node-type="element"
   select="'newVal'" position="last-child"/>
```

The keylist.xml file can have a structure similar to this:

```
<KeyList>
   <Key>one</Key>
   <Key>two</Key>
</KeyList>
```

Generate Multiple Similar Quick Fixes

You can generate the same Schematron Quick Fix (on page 3223) for multiple matches. To do this, you can add the @use-for-each attribute inside the <sqf:fix> element and for each match of the XPath expression in the value of the @use-for-each attribute, a Quick Fix will be presented to the user. The XPath expression does not change the context of the Quick Fix. If you want to access the current match from the XPath expression, you can use the $sqf:current variable.

**Example:**

Suppose you want to restrict the user from entering more than 4 list items in a list. The following example presents an error on any list that has more than 4 list items and offers a Quick Fix with multiple proposals where the user would specify which list item to remove.

```
<sch:rule context="ul">
   <sch:report test="count(li) gt 4" sqf:fix="removeAnyItem">
       The list cannot contain more than 4 entries.
   </sch:report>
   <sqf:fix id="removeAnyItem" use-for-each="1 to count(li)"
      sqf:description>
       <sqf:title>Remove item #<sch:value-of select="$sqf:current"/></sqf:title>
   </sqf:description>
   <sqf:delete match="li[$sqf:current]">
   </sqf:delete>
</sch:rule>
```
Localizing SQF Messages

Oxygen XML Editor provides support for presenting Schematron Quick Fix messages in multiple languages. The language used for the SQF messages is the language specified in the 'Message Language option in the Schematron preferences page (on page 241). If you want to provide an alternative message for a specific language, you can reference IDs or key values for the specific alternate text phrase. In Oxygen XML Editor, the alternate text phrase is defined in a `<sch:diagnostic>` element and it can be used in conjunction with `<sch:assert>` or `<sch:report>` elements.

Example:

The following example presents a quick fix with a different message depending on whether the user’s language is English or German.

```xml
<sch:rule context="dog">
    <sch:assert test="bone" diagnostics="d_en d_de" sqf:fix="addBone" >
        A dog should have a bone. </sch:assert>

    <sqf:fix id="addBone">
        <sqf:description>
            <sqf:title ref="fix_en fix_de" xml:lang="en">Add a bone</sqf:title>
            <sqf:p ref="fix_desc_en fix_desc_de" xml:lang="en">Add bone element as child</sqf:p>
        </sqf:description>
        <sqf:add node-type="element" target="bone"/>
    </sqf:fix>
</sch:rule>
....
<sch:diagnostics xml:lang="en">
    <sch:diagnostic id="d_en"> A dog should have a bone. </sch:diagnostic>
    <sch:diagnostic id="fix_en"> Add a bone </sch:diagnostic>
    <sch:diagnostic id="fix_desc_en">Add a bone element as child</sch:diagnostic>
</sch:diagnostics>
<sch:diagnostics xml:lang="de">
    <sch:diagnostic id="d_de"> Ein Hund sollte ein Bein haben. </sch:diagnostic>
    <sch:diagnostic id="fix_de"> Fügen Sie einen Knochen hinzu </sch:diagnostic>
    <sch:diagnostic id="fix_desc_de"> Fügen Sie ein Knochen element als untergeordnetes Element hinzu </sch:diagnostic>
</sch:diagnostics>
```

Integrating SQF in a Framework and Sharing Them

You can use Schematron Quick Fixes (on page 3223) to assist your content authors by imposing rules for an entire framework (on page 3219) (document type) and offering fixes when a rule violation is detected.
For example, if you are using DITA, you may want your contributors to avoid inserting a figure (\texttt{<fig>} element) inside a paragraph (\texttt{<p>} element) that contains other content since it may result in undesirable placement or spacing in the output. The Schematron rule and its \textit{Quick Fix} for this particular use-case could look like this:

```xml
<schema xmlns="http://purl.oclc.org/dasl/schematron"
   xmlns:sqf="http://www.schematron-quickfix.com/validator/process"
   queryBinding="xslt2">
  <pattern id="check.figure.location">
    <rule context="p/fig">
      <report test="true()" role="warn" sqf:fix="moveAfter">
        A figure inside a paragraph doesn't transform well into PDF. </report>
      <sqf:fix id="moveAfter">
        <sqf:description>
          Move after the paragraph. </sqf:title>
        </sqf:description>
        <let name="figToMove" value="."/>
        <sqf:add match="parent::p" select="$figToMove" position="after"/>
        <sqf:delete match="."/>
      </sqf:fix>
    </rule>
  </pattern>
</schema>
```

The result of this example would be that the user will see a warning if they insert a \texttt{<fig>} element inside a \texttt{<p>} element and they are presented with the option of selecting the \textit{Quick Fix} that would move the figure outside the paragraph.

**How to Integrate SQF in a Framework**

To integrate a Schematron \textit{Quick Fix} in a framework (on page 3219), follow these steps:

1. Create a folder structure for an extended framework and save it somewhere on disk where you have full write access (for example, \texttt{custom_frameworks/dita-extension}).
2. In that new folder structure, create another folder that will contain all of your custom Schematron files (for example, \texttt{custom_frameworks/dita-extension/rules}).
3. Define the Schematron \textit{Quick Fix} for a rule (on page 1236) in an existing or new Schematron file and save it in the folder you created in step 2.
4. Open the Preferences dialog box (Options > Preferences) (on page 124) and go to Document Type Association > Locations (on page 139). In this preferences page, add the path to your \texttt{custom_frameworks} folder in the Additional frameworks directories list, then click OK or Apply to save your changes.
5. Go to the Document Type Association preferences page (on page 138) and select a framework configuration (for example, DITA) and use the Extend button to create an extension for it.
6. Give the extension an appropriate name (for example, *DITA - Custom*), select **External** for the **Storage** option, and specify an appropriate path to your framework configuration file (for example, `path/to/.../custom_frameworks/dita-extension/dita-extension.framework`).

7. Make whatever changes you desire to the extension, then go to the **Validation** tab, edit the default validation scenario (select the scenario and click the 📋 **Edit** button), and add an extra validation unit to it (one that uses your custom Schematron file that includes the SQF). For more details about editing validation scenarios, see **Configuring Validation Scenarios for a Framework** *(on page 2218)*.

8. Click **OK** to close the dialog box and then **OK** or **Apply** to save the changes to the **Document Type Association** preferences page *(on page 138)*.

9. Add a reference to the Schematron file that includes the SQF in your framework by following the procedure in **Associating a Schema in Validation Scenarios Defined in the Document Type** *(on page 821)*.

10. Open a document in your framework and test the new rule and **Quick Fix**.

11. You can continue to refine the Schematron and develop additional rules as needed.

**Sharing Schematron Quick Fixes**

To share Schematron Quick Fixes with other members of your team, you simply need to share the framework where you integrated the SQF. There are several methods for sharing frameworks and you can find details here: **Sharing a Framework** *(on page 2288)*.

**Related Information:**

- **Defining Schematron Quick Fixes** *(on page 1236)*
- **Basic Schematron Quick Fix Operations** *(on page 1237)*
- **Associating a Schema in Validation Scenarios Defined in the Document Type** *(on page 821)*
- **Sharing a Framework** *(on page 2288)*

**Validating Schematron Quick Fixes**

By default, Schematron **Quick Fixes** *(on page 3223)* are validated as you edit them within the Schematron file or while editing them in a separate file. To change this, open the **Preferences** dialog box *(Options > Preferences)* *(on page 124)*, go to **Editor > Document Checking**, and deselect the **Enable automatic validation** option *(on page 226)*.

To validate Schematron **Quick Fixes** manually, select the ✔️ **Validate** action from the ✔️ • **Validation** toolbar drop-down menu or the **Document > Validate** menu. The validation problems are highlighted directly in the editor, making it easy to locate and fix any issues.

**Related Information:**

- **Validating XML Documents Against a Schema** *(on page 775)*
- **Validation Scenario** *(on page 787)*
Content Completion in SQF

Oxygen XML Editor helps you edit Schematron Quick Fixes (on page 3223) embedded in a Schematron document by offering proposals that are valid at the cursor position in a Content Completion Assistant (on page 3217). It can be manually activated with the Ctrl + Space (Command + Space on macOS) shortcut.

When you edit the value of an attribute that references a Quick Fix ID, the ids are collected from the entire definition scope. For example, if the editing context is $assert/@sqf:fix$, the Content Completion Assistant proposes all fixes defined locally and globally.

If the editing context is an attribute value that is an XPath expression (such as $sqf:add/@match$ or $replace/@select$), the Content Completion Assistant offers the names of XPath functions, the XPath axes, and user-defined variables and parameters.

The Content Completion Assistant displays XSLT 1.0 functions (and optionally XSLT 2.0 / 3.0 functions) in the $@path$, $@select$, $@context$, $@subject$, and $@test$ attributes, depending on the Schematron options (on page 240) that are set in Preferences pages. If the Saxon namespace ($xmlns:saxon="http://icl.com/saxon"$) or the Saxon namespace is declared in the Schematron schema ($xmlns:saxon="http://saxon.sf.net/*$) the content completion also displays the XSLT Saxon extension functions.

Highlight Quick Fix Occurrences in SQF

When you position your mouse cursor over a Quick Fix (on page 3223) ID in a Schematron document, Oxygen XML Editor searches for the Quick Fix declaration and all its references and highlights them automatically.

Customizable colors are used: one for the Quick Fix definition and another one for its references. Occurrences are displayed until another Quick Fix is selected.

To change the default behavior of Highlight Component Occurrences, open the Preferences dialog box (Options > Preferences) (on page 124) and go to Editor > Mark Occurrences. You can also trigger a search using the Search > Search Occurrences in File (Ctrl + Shift + U (Command + Shift + U on macOS)) action from contextual menu. Matches are displayed in separate tabs of the Results view (on page 548).

Searching and Refactoring Operations in SQF

Search Actions

The following search actions can be applied on Quick Fix (on page 3223) IDs and are available from the Search submenu in the contextual menu of the current editor or from the Document > References menu:

- **Search References**

  Searches all references of the item found at current cursor position in the defined scope, if any. If a scope is defined, but the currently edited resource is not part of the range of resources.
determined by this, a warning dialog box is displayed and you have the possibility to define another search scope.

**Search References in**

Searches all references of the item found at current cursor position in the file or files that you specify when define a scope in the Search References dialog box.

**Search Declarations**

Searches all declarations of the item found at current cursor position in the defined scope if any. If a scope is defined, but the currently edited resource is not part of the range of resources determined by this, a warning dialog box will be displayed and you have the possibility to define another search scope.

**Search Declarations in**

Searches all declarations of the item found at current cursor position in the file or files that you specify when you define a scope for the search operation.

**Search Occurrences in File**

Searches all occurrences of the item at the cursor position in the currently edited file.

**Refactoring Actions**

The following refactoring actions can be applied on Quick Fix IDs and are available from the Refactoring submenu in the contextual menu of the current editor or from the Document > Refactoring menu:

**Rename Component**

Allows you to rename the current component (in-place). The component and all its references in the document are highlighted with a thin border and the changes you make to the component at the cursor position are updated in real time to all occurrences of the component. To exit the in-place editing, press the Esc or Enter key on your keyboard.

**Rename Component in**

Opens a dialog box that allows you to rename the selected component by specifying the new component name and the files to be affected by the modification. If you click the Preview button, you can view the files to be affected by the action.
Embedding Schematron Quick Fixes in Relax NG or XML Schema

Schematron Quick Fixes (on page 3223) can be embedded into an XML Schema through annotations (using the <appinfo> element), or in a Schematron rule embedded in the RELAX NG Schema. For more information about embedding Schematron in XML Schema or Relax NG, see Embedding Schematron Rules in XML Schema or RELAX NG (on page 1211).

Oxygen XML Editor is able to extract and use the embedded Schematron Quick Fixes. To make the embedded Schematron Quick Fixes available, follow these steps:

1. Define a validation against a schema (on page 775).
2. For the Schema type, choose XML Schema Or Relax NG.
3. Select the Embedded Schematron rules option.

Example: Embedded Schematron Quick Fix in XML Schema

```xml
<xsd:appinfo>
  <sch:pattern>
    <sch:rule context="...">
      <sch:assert test="..." sqf:fix="fixId">Message.</sch:assert>
      <sqf:fix id="fixId"/>
  </sch:rule>
</sch:pattern>
</xsd:appinfo>
```
Example: Embedded Schematron Quick Fix in Relax NG

```xml
<grammar

    xmlns="http://relaxng.org/ns/structure/1.0"
    xmlns:sch="http://purl.oclc.org/dsdl/schematron">

    <sch:pattern>
        <sch:rule context="...">
            <sch:assert test="..." sqf:fix="fixId">Message.</sch:assert>
            <sqf:fix id="fixId">
                ....
            </sqf:fix>
        </sch:rule>
    </sch:pattern>

    <start>
        ............
    </start>

</grammar>
```

Tip:
For more extensive examples, see the samples in the `[OXYGEN_INSTALL_DIR]/samples/schematron` folder.

Related Information:
- Embedding Schematron Rules in XML Schema or RELAX NG (on page 1211)
- Defining Schematron Quick Fixes (on page 1236)

Editing SVG Files

SVG (Scalable Vector Graphics) is a platform for two-dimensional graphics. It has two parts: an XML-based file format and a programming API for graphical applications. Some of the key features include support for shapes, text, and embedded raster graphics with many painting styles, scripting through languages such as ECMAScript, and support for animation.

SVG is a vendor-neutral, open standard that has important industry support. Companies such as Adobe, Apple, and IBM have contributed to its W3C specifications. Many documentation frameworks (on page 3219) (including DocBook) have support for SVG by defining the graphics directly in the document.
Oxygen XML Editor adds SVG support by using the Batik distribution package (an open source project developed by the Apache Software Foundation) and the default XML Catalog (on page 826) resolves the SVG DTD.

Note:
Batik partially supports SVG 1.1. For a detailed list of supported elements, attributes, and properties, see the Batik Implementation Status page.

How to Render SVG Images that Use Java Scripting

1. Copy the js.jar library from the Batik distribution into the Oxygen XML Editor lib folder.
2. Restart the application.

SVG 1.2 Rendering Issues

Oxygen XML Editor uses the Apache Batik open source library to render SVG images and it only has partial support for SVG 1.2. For more information, see http://xmlgraphics.apache.org/batik/dev/svg12.html.

This partial support could lead to some rendering issues in Oxygen XML Editor. For example, if you are using the Inkscape SVG editor, it is possible for it to save the SVG as 1.1, while using SVG 1.2 elements (such as <flowRoot>) inside it. This means that the image will not be properly rendered inside the application.

Note:
SVG images shown in the Author visual editing mode are rendered as static images, without support for animations and JavaScript.

Standalone SVG Viewer

Oxygen XML Editor includes a simple SVG Viewer that allows you to work with SVG images.

To open the viewer, select SVG Viewer from the Tools menu.
You can browse for and open any SVG file that has the .svg or .svgz extension.

If the file is included in the current project, you can open it in the viewer by right-clicking the image file in the Project view (on page 402) and selecting Open with > SVG Viewer.

**Actions Available in the SVG Viewer**

The following actions are available in the SVG Viewer:

**Zoom in**

To zoom in on an image, use any of the following methods:

- Scroll forward with the mouse wheel.
- Select **Zoom in** from the contextual menu.
- Use the **Ctrl + I (Command + I on macOS)** keyboard shortcut.

**Zoom out**

To zoom out on an image, use any of the following methods:

- Scroll backward with the mouse wheel.
- Use the **Ctrl + O (Command + O on macOS)** keyboard shortcut.
- Select **Zoom out** from the contextual menu.

**Rotate**

To rotate an image, use either of the following methods:

- Use the **Ctrl + Right-Click + Drag (Command + Right-Click + Drag on macOS)** shortcut.
- Select **Rotate** from the contextual menu. This rotates the image exactly 90 degrees clockwise.
Refresh

To refresh (or reset) an image, use either of the following methods:

- Use the Ctrl + T (Command + T on macOS) keyboard shortcut.
- Select Refresh from the contextual menu.

Move

To move an image, use either of the following methods:

- Use the Arrow Keys on your keyboard.
- Use the Shift + Left-Click + Drag shortcut.

Pan

To pan an image, click and drag the image with your mouse.

Related Information:

- Editing SVG Files (on page 1251)

Integrated SVG Viewer in the Results Panel

Oxygen XML Editor includes an integrated SVG Viewer that can render the results of an XSLT transformation scenario that generates SVG images in the Results panel (on page 548) at the bottom of the editor. This is useful for developing XSL stylesheets with the capability of producing SVG graphics.

To enable this feature, select Show in results view as > SVG in the Output tab of the XSLT transformation scenario configuration dialog box (on page 1474). When you run the transformation, the SVG result is then rendered in an integrated SVG viewer in the Results panel (on page 548).

Example of a Use-Case

Suppose you have an XML document that describes the evolution of your sales over a time period and you want to create a graphic for it. You could use the following steps to accomplish this task:

1. Start with a static SVG image, written directly in Oxygen XML Editor or exported from a external graphics tool.
2. Extract the parts that are dependent upon the data from the XML document and create an XSL template to produce the image.
3. Create an XML transformation with XSLT scenario (on page 1464).
4. While configuring the transformation scenario, select Show in results view as > SVG in the Output tab (on page 1474) of the configuration dialog box.
5. Run the transformation.

The SVG image is rendered in an integrated SVG viewer in the Results panel (on page 548) at the bottom of the editor.
Editing HTML Documents

Oxygen XML Editor provides a special framework for editing HTML files (html or htm file extensions) with a variety of specialized editing features, including validation, content completion, syntax highlighting, HTML-specific actions, and more. You can edit HTML documents in Text or Author mode.

Oxygen XML Editor also includes a built-in XHTML framework (on page 1374) (files with the http://www.w3.org/1999/xhtml namespace or with the xhtml or xht file extension) that has a full set of features (full editing support, document templates, enhanced CSS rendering, specific actions, validation, content completion, transformation scenarios, and more). Oxygen XML Editor also includes support for importing HTML files as an XML document (on page 2098).

Resources

For more information about the HTML support in Oxygen XML Editor, see the following resources:

- Video: HTML Support in Oxygen.

Related information

XHTML Document Type (Framework) (on page 1374)

HTML Editor

Oxygen XML Editor includes a specialized HTML editor and various editing features for files that have the html or htm file extensions. The encoding is detected automatically based on the value specified in the `charset` attribute of the `<meta>` element.
Note:
If an HTML document has an XHTML namespace, or there is an XSD schema declared, or there is a
PUBLIC ID specified in a DOCTYPE, or there is a SYSTEM ID with a value other than "about:legacy-
compat", then the document will be opened as an XHTML file.

**New Document Template**

Oxygen XML Editor includes a new document template to help you get started creating HTML content. It
is available when creating new documents from templates *(on page 368)* and can be found in the New
Document folder or by typing *html* in the search field.

**Text Mode Editor**

You can edit HTML files in the Text editing mode *(on page 517)* using all of its useful features *(on page
517)*. It includes content completion *(on page 1259)* based on a special HTML schema, syntax highlighting
*(on page 1259)*, a specialized Outline view *(on page 1261)* that presents the structure, folding support *(on
page 1260)*, and more.

HTML documents support formatting and indenting single or multiple documents to make them more
readable. The formatting works even if the document is not XML well-formed and it also works on embedded
CSS or JavaScript code. The HTML formatting details are similar to those for XML documents. For details, see
Formatting and Indenting XML Documents *(on page 555)*.

**Author Mode Editor**

You can edit HTML files in the visual Author editing mode *(on page 588)*, but when opening an HTML
document in Author mode, if it is not considered well-formed according to XML standards, you will see a
warning message at the top of the editor explaining that once you make a modification, the document will
be automatically converted to proper XML structure. For more details, see XML Well-Formedness Details for
HTML Documents *(on page 1257)*.

When editing HTML documents in Author mode, you have access to many of the same authoring features and
actions as you have with XHTML documents *(on page 1375)*. You also have the benefit of CSS rendering and
you can specify a CSS file to be associated with an HTML document *(on page 1262)*.

**HTML-Specific Contextual Menu Actions**

There are some specialized actions (available in the contextual menu when you right-click anywhere in the
current HTML document) that invoke features unique to HTML documents. These contextual menu actions
include:

- **View in Browser/System Application**
  - Opens the HTML document in your default browser.

- **Minify HTML**
Compresses the HTML document by removing unnecessary white spaces, without affecting the functionality of the document, but significantly reducing the loading time in Web browsers.

**HTML to XML Well-formed (Available in both Text and Author modes)**

Converts the currently edited HTML document to be XML well-formed. This means that unclosed tags will be properly closed, unquoted attribute values will be quoted, and more. This is helpful if, for example, you use XSLT stylesheets while applying transformations on HTML documents (since the transformation will require the HTML document to be XML well-formed).

**XML Well-Formedness Details for HTML Documents**

When opening an HTML document in **Author** mode, if it is not XML well-formed, you will see a warning message at the top of the editor explaining that once you make a modification, the document will be automatically converted to proper XML structure. Examples of things that are automatically converted include:

- Missing end tags are added to applicable elements.
- Empty tags are closed.
- Missing quote characters are added to applicable attributes.
- Entity references that are allowed in the **HTML5 specification** are converted to the corresponding character.

**Tip:**

It is also possible to manually convert HTML documents to be XML well-formed. There are two ways to do this, depending on whether you want to convert a single document or batch convert multiple documents at once:

- **Single Document** - To convert a single HTML document, right-click anywhere in the currently open HTML document (in the main editor) and select **HTML to XML Well-formed**.
- **Multiple Documents** - To batch convert multiple HTML documents at once, go to the **Project** view, select the documents you want to convert, then right-click and select **HTML to XML Well-formed**.

**Notes:**

- All selected HTML files are backed up before being processed (same path/name but with the ".bak" extension added at the end).
- Any detected conversion errors are grouped and listed in a dedicated tab in the **Results** pane at the bottom of the application.
- A brief report is displayed at the end of the operation.
**HTML Validation**

Oxygen XML Editor includes a built-in default validator used for validating HTML documents. It is based upon the W3C HTML Validator and the HTML documents are validated against the [W3C HTML5 specification](https://www.w3.org/html). The validator in Oxygen XML Editor only supports HTML5 structure. It presents the errors in the editor similar to XML documents ([on page 777](#)). It also checks the embedded CSS content and the warnings and errors are presented similar to the CSS editor ([on page 1076](#)).

**Validating HTML Against a Schematron**

It is also possible to validate HTML documents against a Schematron schema. Besides the default HTML validator, Oxygen XML Editor also includes a built-in *HTML Schematron Validator* engine. There are several ways to validate an HTML document against a Schematron:

- **Configure a Validation Scenario** - You can [create or edit a validation scenario](#), change the **File type** column to *HTML Document*, change the **Validation engine** column to *HTML Schematron Validator*, and specify the Schematron document in the **Schema** column.

- **Manually Validate a Single Document** - You can use the *Validate with* action from the *Validation* drop-down menu on the toolbar. This opens a dialog box where you can specify the Schematron document to validate the current document against.

- **Batch Validate Multiple Documents** - You can select multiple HTML documents in the **Project** view, right-click, and use the *Validate with schema* action from the **Validate** submenu. This opens a dialog box where you can specify the Schematron document to validate the selected documents against.

**Notes:**

- The Schematron must use the HTML5 namespace to reference the elements from the instance.
- Implicit HTML elements (i.e. `<html>`, `<body>`, `<tbody>`) must be included in an XPath expression in the Schematron document, even if they are missing from the HTML document.

**Tip:**

The Oxygen XML Editor installation directory includes a *samples* folder that contains numerous sample files to help you learn about features, certain file types, and XML technologies. For example, inside the *samples* folder, there is an *html* folder with a *schematron* subfolder where you can find some samples that illustrate HTML validation against a Schematron schema.

**Validating HTML Against Other Types of Schema**

If your HTML document is XML well-formed, you could also configure a validation scenario to validate it as an XML document against other types of schemas. You would [create or edit a validation scenario](#).
787), make sure the File type column is set to XML Document, select the appropriate Validation engine, and specify the schema document in the Schema column.

**HTML Content Completion Assistant**

Oxygen XML Editor includes an intelligent Content Completion Assistant (on page 3217) that offers proposals for inserting HTML structures that are valid at the current editing location. Content completion is even available for CSS and JavaScript code that is embedded in an HTML document.

The Content Completion Assistant is enabled by default. To disable it, open the Preferences dialog box (Options > Preferences) (on page 124), go to Editor > Content Completion, and deselect the Enable content completion option (on page 211).

![Figure 452. Content Completion Assistant in HTML](image)

**Using the Content Completion in HTML**

For HTML documents, the Content Completion Assistant uses a built-in schema and the list of proposals depend on the RELAX NG schema specified in the HTML framework. Using the content completion feature is the same as with any other XML document. For more details, see:

- Using the Content Completion Assistant in Text Mode (on page 532)
- Using the Content Completion Assistant in Author Mode (on page 616)

**Code Templates in the Content Completion**

Oxygen XML Editor includes a set of built-in code templates for HTML documents that can be selected from the Content Completion Assistant. The code templates are displayed with a symbol in the content completion list. You can also define your own code templates and share them with others. For more information, see Code Templates (on page 536).

**Content Completion for XPath Expressions**

When entering XPath expressions in the XPath toolbar or XPath Builder view, the Content Completion Assistant is available as you type to help you compose query patterns.

**Syntax Highlighting in HTML Documents**

Oxygen XML Editor supports syntax highlighting in Text mode to make it easier to read the semantics of the structured content by displaying each type of code in different colors and fonts.
For HTML documents, it handles attributes without quotes, unclosed or void elements, and it also offers highlighting for embedded CSS or JavaScript content.

To customize the colors or styles used for the syntax highlighting colors for HTML files, follow these steps:

1. Open the Preferences dialog box (Options > Preferences) (on page 124).
2. Go to Editor > Syntax Highlight (on page 224).
3. Select and expand the XHTML section in the top pane.
4. Select the component you want to change and customize the colors or styles using the selectors to the right of the pane.

You can see the effects of your changes in the Preview pane.

Related Information:
Syntax Highlight Preferences (on page 224)

Folding in HTML

In a large HTML document, elements can be collapsed so that only the needed data remains in focus. The folding features available for XML documents (on page 528) are also available in HTML documents, but it also provides folding for nested elements that are not closed.

Minifyng HTML Documents

Minification (or compression) of an HTML document is the practice of removing unnecessary white spaces, without affecting the functionality of the document, but significantly reducing the loading time in Web browsers. While a minified HTML document gains in terms of execution performance, it is more difficult to read.

To minify an HTML document, right-click anywhere in the editor for an HTML document that is open in Text mode (or right-click an HTML document in the Project view and select the Minify HTML action. This opens a dialog box with the following options:

- **Output file**
  Use this option to set the name and location of the resulting compressed/minified HTML document.

- **Remove comments**
  If selected (default), all the HTML comments and also the comments from embedded CSS or JavaScript code blocks will be removed from the resulting output file.

- **Compress on a single line**
  If selected (default), the resulting output file will consist of a single line, as all the ‘new line’ characters from the source document are removed.

- **Open output file in editor**
If selected (default), the resulting output file will be opened in Oxygen XML Editor.

When you click **OK**, the resulting HTML document is a compressed version of the original file for the purpose of enhanced performance, while losing some readability. The source HTML document is not affected.

**HTML Outline View**

The **Outline** view for HTML documents displays the structure of the HTML document you are editing. By default, it is displayed on the left side of the editor. In addition to the normal features available in the **Outline** view for XML documents (on page 539), the HTML **Outline** view also handles void elements, elements that are not closed, or attributes without quotes, and presents the tree structure of the HTML document correctly.

![HTML Outline View](image)

**Figure 453. HTML Outline View**

**Querying HTML Documents with XPath**

Oxygen XML Editor provides an **XPath** toolbar that makes it easy to quickly query HTML documents using XPath expressions. You can also use the dedicated **XPath Builder view** (on page 2002) that allows you to compose more complex XPath expressions and execute them over HTML documents (even if they are not well-formed according to XML standards). Both the **XPath toolbar** and **XPath Builder** view offer content completion as you type to help you compose expressions.

**XPath Toolbar**

You can run an XPath expression over an HTML document or on an entire project. For more information about this toolbar, see **XPath Toolbar** (on page 2000).
Note:
Implicit HTML elements (i.e. `<html>`, `<body>`, `<tbody>`) must be included in the XPath expression, even if they are missing from the HTML document.

Figure 454. XPath Toolbar for HTML

Associating a CSS with an HTML Document

The rendering of an HTML document in the Author mode is driven by a CSS stylesheet that conforms to the version 2.1 of the CSS specification from the W3C consortium. Oxygen XML Editor also supports stylesheets coded with the LESS dynamic stylesheet language.

To associate a CSS with an HTML document:

1. Use the Associate XSLT/CSS Stylesheet action that is available on the toolbar or in the Document > XML Document menu.
2. In the resulting dialog box, specify the URL for the CSS file and optionally a title, and click OK.

Result: A CSS association is added in the HTML document in a `<link>` element, as specified in the W3C stylesheet specification.

Editing Markdown Documents

Markdown was created as a lightweight markup language with plain text formatting syntax designed to provide syntax that is very easy to read and write, and to convert it to HTML and other formats. It is often used by content contributors who want a quick and easy way to write content without having to take their
fingers off the keyboard and without having to learn numerous codes or shortcuts, and it can easily be shared interchangeably between virtually any types of contributor and system.

Oxygen XML Editor provides a built-in Markdown editor that allows you to convert Markdown syntax to HTML or DITA and it includes a preview panel to help you visualize the final output. Aside from the plain text syntax that is common among most Markdown applications, the editor in Oxygen XML Editor also integrates many other powerful features that content authors are accustomed to using for other types of documents. Some of these additional unique features include:

- Additional toolbar and contextual menu actions.
- Automatic validation to help keep the syntax valid.
- Dedicated syntax highlighting to make Markdown documents even easier to read and write.
- Unique features for creating Markdown documents directly in DITA maps (on page 3218) and converting Markdown documents to DITA topics.
- Specialized syntax rules to combine popular syntax features from several specifications.

Resources

For more information about editing Markdown documents in Oxygen XML Editor, see our webinar: Oxygen Markdown Support.

Markdown Editor

Oxygen XML Editor provides an intuitive, dynamic, and easy-to-use Markdown editor. It is a split-screen editor with two panels that are synchronized in real time. The left side is a simple text editor that is specially designed for writing Markdown syntax. The right side is a WYSIWYG style preview of how changes will look in the output.

Markdown Text Editor Pane (Left Side)

The left pane is a simple text editor that is refined to accept Markdown syntax. At the same time, you still have many of the actions, options, and features that you are used to when editing any other type of document in Oxygen XML Editor.

The features of this special editor that are unique for Markdown documents include:

- **Unique Markdown Syntax Rules** - The Markdown editor in Oxygen XML Editor uses an integration of rules (on page 1277) that combine rules from common default Markdown syntax along with many of the rules used in the GitHub Flavored Markdown syntax.

- **Syntax Highlighting** - The Oxygen XML Editor syntax highlighting feature is integrated into the Markdown text editor to make it easier to read and write Markdown syntax. You can even customize the colors and styles for the syntax highlighting (on page 1274).

- **Automatic Spell Checking** - The Markdown editor supports the Oxygen XML Editor automatic spell checking feature (on page 456) that reports possible misspelled words as you type. You simply need to select the Automatic spell check option in the Spell Check preferences page (on page 229), then click the Select editors button and select Markdown Editor.
• **Helpful Toolbar and Contextual Menu Actions** - A variety of unique actions *(on page 1266)* are available from the toolbar to help you insert proper Markdown syntax. The contextual menu also includes some common editing actions, as well as unique actions to export (convert) Markdown documents to HTML or DITA.

• **Shortcut Keys** - Many of the shortcut keys that are most commonly used *(on page 55)* in Oxygen XML Editor also work in the Markdown editor.

**WYSIWYG Preview Pane (Right Side)**

The right pane is a WYSIWYG *Preview* pane that shows a visual representation of how changes made in the left-side text editor will be converted to **HTML**, **XDITA** (Lightweight DITA XML), or **DITA** output. The changes you make in the text editor are parsed continually and they are immediately visible in the *Preview* pane. There are two tabs available in the *Preview* pane, one for visualizing DITA output and one for visualizing HTML output. You can switch between the two tabs at the bottom of the pane.

The *Preview* pane includes the following features:

• **WYSIWYG Visualization** - This pane presents the Markdown syntax from the left-side text editor in a visual WYSIWYG style interface that is automatically synchronized as you type.

• **Synchronous caret and scroll synchronization** - Moving the cursor in the editor area will display the corresponding move in the *Preview* area. In addition, moving the cursor in the *Preview* area will display the corresponding move in the editor area.

• **Export Options** - The **DITA** tab includes a contextual menu action to export (convert) the current Markdown document to a DITA topic *(on page 1273)*. The **XDITA** tab includes a contextual menu action to export (convert) the current Markdown document to a Lightweight DITA topic *(on page 1273)*. Similarly, the **HTML** tab includes a contextual menu action to export (convert) it to an XHTML document *(on page 1273)*.

• **Automatic Validation** - As you edit Markdown documents, they are validated automatically *(on page 1274)*. The conversion engine constantly tries to parse your changes and if a change results in an error that prevents the parser from converting the syntax, an error message will be displayed in the *Preview* pane or *Results view* *(on page 548)* at the bottom of the editor.

• **Print Feature** - The Markdown editor includes a **Print** action that is available in the contextual menu and it allows you to configure options for printing the current document as you see it in the *Preview* pane.

• **Preview Markup** - The Markdown editor includes a **Tags Display Mode** drop-down menu *(on page 1268)* that is available on the toolbar and it allows you to control the amount of markup that is displayed in the *Preview* pane.

• **Specialized DITA Features** - The Markdown editor includes some unique, specialized features to integrate it with the powerful DITA support *(on page 3024)* in Oxygen XML Editor.
Creating New Markdown Documents

To create a new Markdown document in Oxygen XML Editor, follow these steps:

1. Click the **New** button on the toolbar or select **File > New**.
2. Select the **Markdown** document template (in the **New Document** folder).
3. Click the **Create** button.

**Result:** A new Markdown document is created and it is opened in the specialized Markdown Editor. 

Related Information:

Markdown Editor Syntax Rules and Specifications (on page 1277)
Actions Available in the Markdown Editor (on page 1266)
Working with Markdown Documents in DITA (on page 3024)
Creating New Markdown Documents (on page 1265)
Actions Available in the Markdown Editor

Aside from the actions that are available in Oxygen XML Editor for any type of document (such as the actions in the various menus and the common sections of the toolbar), a variety of unique actions are also available in the Markdown editor, from the toolbar and contextual menu.

Toolbar Actions

The following default actions are available on the Markdown toolbar when editing Markdown documents:

- **H1 Header (1st Level)**
  Inserts an *atx-style first-level header (on page 1278)* at the cursor position.

- **H2 Header (2st Level)**
  Inserts an *atx-style second-level header (on page 1278)* at the cursor position.

- **H3 Header (3rd Level)**
  Inserts an *atx-style third-level header (on page 1278)* at the cursor position.

- **Horizontal Rule**
  Inserts a *horizontal rule (on page 1278)* at the cursor position.

- **Bold (Strong)**
  Marks the selected text with *bold (on page 1279)*.

- **Italic (Emphasis)**
  Marks the selected text with *italics (on page 1279)*.

- **Strikethrough**
  Marks the selected text with a *strikethrough (on page 1279)*.

- **Code Block**
  Inserts (or surrounds selected text in) a *codeblock (on page 1283)*.

- **Blockquote**
  Inserts a *blockquote (on page 1282)* at the cursor position.

- **Insert Link**
  Opens the *Insert Link* dialog box that allows you to define a *link (on page 1280)* to insert at the cursor position.
**Insert Image**

Opens the **Insert Image** dialog box that allows you to define an image *(on page 1282)* to insert at the cursor position. You can type the URL of the image you want to insert or use browsing actions in the **Browse** drop-down menu.

**Insert Ordered List**

Inserts an ordered list *(on page 1285)* at the cursor position. Three child list items are also automatically inserted by default. You can also use this action to convert selected content to an ordered list.

**Insert Unordered List**

Inserts an unordered list *(on page 1285)* at the cursor position. Three child list items are also automatically inserted by default. You can also use this action to convert selected content to an unordered list.

**Insert Task List**

Inserts a task list *(on page 1287)* at the cursor position. Three child list items are also automatically inserted by default. You can also use this action to convert selected content to a task list.

**Insert Table**

Inserts a table *(on page 1287)* at the cursor position.
Tags Display Mode drop-down menu

Allows you to control the amount of markup that is displayed in the Preview pane and offers the following choices:

- **Full Tags with Attributes**
  Displays full tag names with attributes for both block (on page 3216) and inline elements (on page 3220).

- **Full Tags**
  Displays full tag names without attributes for both block elements and inline elements.

- **Block Tags**
  Displays full tag names for block elements and simple tags without names for inline elements.

- **Block Tags without Element Names**
  Displays tags for block elements but without element names for a more compact version of Block Tags mode. You can still see the element names by hovering over the tags.

- **Inline Tags**
  Displays full tag names for inline elements, while block elements are not displayed.

- **Partial Tags**
  Displays simple tags without names for inline elements, while block elements are not displayed.

- **No Tags**
  No tags are displayed. This is the most compact mode and is as close as possible to a word-processor view.

Configure Tags Display Mode

Use this option to go to the **Author preferences page (on page 176)** where you can configure the Tags Display Mode options.

Contextual Menu Actions

The following default actions are available in the contextual menu when editing Markdown documents:

- **Cut, Copy, Paste**
  Use these actions to execute the typical editing actions on the currently selected content.

Source submenu

This submenu includes the following actions:
To Upper Case

Converts the content selection to upper case characters.

To Lower Case

Converts the content selection to lower case characters.

Capitalize Lines

It capitalizes the first letter found on every new line that is selected. Only the first letter is affected, the rest of the line remains the same. If the first character on the new line is not a letter then no changes are made.

Convert Hexadecimal Sequence to Character (Ctrl + Shift + X (Command + Shift + X on macOS))

Converts a sequence of hexadecimal characters to the corresponding Unicode character (on page 463). The action can be invoked if there is a selection containing a valid hexadecimal sequence or if the cursor is placed at the right side of a valid hexadecimal sequence. A valid hexadecimal sequence can be composed of 2 to 4 hexadecimal characters and may or may not be preceded by the 0x or 0X prefix. Examples of valid sequences and the characters they will be converted to:

- 0x0045 will be converted to E
- 0X0125 to ĥ
- 265 to ɥ
- 2190 to ←

Note:
For more information about finding the hexadecimal value of a character, see Finding the Decimal, Hexadecimal, or Character Entity Equivalent (on page 466).

Base64 Encode/Decode submenu

This submenu includes the following actions for encoding or decoding base 64 schemes:

Import File to Encode and Insert

Encodes a file and then inserts the encoded content into the current document at the cursor position.

Decode Selection and Export to File

Decodes a selection of text from the current document and then exports (saves) the result to another file.

Encode Selection
Replaces a selection of text with the result of encoding that selection. By default, a dialog box is displayed that allows you to select the encoding to use. There is an option to choose to not show this dialog box in the future. In this case, the encoding that is specified in the Encoding for Base64, Base32, Hex conversions option in the Encoding preferences page (on page 169) will be used. Likewise, the same is true if the Show the dialog box for choosing the encoding for Base64, Base32, Hex conversions option is not selected in the Messages preference page (on page 308).

**Decode Selection**

Replaces a selection of text with the result of decoding that selection. By default, a dialog box is displayed that allows you to select the encoding to use. There is an option to choose to not show this dialog box in the future. In this case, the encoding that is specified in the Encoding for Base64, Base32, Hex conversions option in the Encoding preferences page (on page 169) will be used. Likewise, the same is true if the Show the dialog box for choosing the encoding for Base64, Base32, Hex conversions option is not selected in the Messages preference page (on page 308).

**Modify All Matches**

Use this option to modify (in-place) all the occurrences of the selected content (or the contiguous fragment where the cursor is located). When you use this option, a thin rectangle replaces the highlights and allows you to start editing. If matches with different letter cases are found, a dialog box is displayed that allows you select whether you want to modify only matches with the same letter case or all matches.

**Base32 Encode/Decode submenu**

This submenu includes the following actions for encoding or decoding base32 schemes:

- **Import File to Encode and Insert**
  
  Encodes a file and then inserts the encoded content into the current document at the cursor position.

- **Decode Selection and Export to File**
  
  Decodes a selection of text from the current document and then exports (saves) the result to another file.

- **Encode Selection**
Replaces a selection of text with the result of encoding that selection. By default, a dialog box is displayed that allows you to select the encoding to use. There is an option to choose to not show this dialog box in the future. In this case, the encoding that is specified in the Encoding for Base64, Base32, Hex conversions option in the Encoding preferences page (on page 169) will be used. Likewise, the same is true if the Show the dialog box for choosing the encoding for Base64, Base32, Hex conversions option is not selected in the Messages preference page (on page 308).

**Decode Selection**

Replaces a selection of text with the result of decoding that selection. By default, a dialog box is displayed that allows you to select the encoding to use. There is an option to choose to not show this dialog box in the future. In this case, the encoding that is specified in the Encoding for Base64, Base32, Hex conversions option in the Encoding preferences page (on page 169) will be used. Likewise, the same is true if the Show the dialog box for choosing the encoding for Base64, Base32, Hex conversions option is not selected in the Messages preference page (on page 308).

**Modify All Matches**

Use this option to modify (in-place) all the occurrences of the selected content (or the contiguous fragment where the cursor is located). When you use this option, a thin rectangle replaces the highlights and allows you to start editing. If matches with different letter cases are found, a dialog box is displayed that allows you select whether you want to modify only matches with the same letter case or all matches.

**Hex Encode/Decode submenu**

This submenu includes the following actions for encoding or decoding hex schemes:

**Import File to Encode and Insert**

Encodes a file and then inserts the encoded content into the current document at the cursor position.

**Decode Selection and Export to File**

Decodes a selection of text from the current document and then exports (saves) the result to another file.

**Encode Selection**
Replaces a selection of text with the result of encoding that selection. By default, a dialog box is displayed that allows you to select the encoding to use. There is an option to choose to not show this dialog box in the future. In this case, the encoding that is specified in the Encoding for Base64, Base32, Hex conversions option in the Encoding preferences page (on page 169) will be used. Likewise, the same is true if the Show the dialog box for choosing the encoding for Base64, Base32, Hex conversions option is not selected in the Messages preference page (on page 308).

**Decode Selection**

Replaces a selection of text with the result of decoding that selection. By default, a dialog box is displayed that allows you to select the encoding to use. There is an option to choose to not show this dialog box in the future. In this case, the encoding that is specified in the Encoding for Base64, Base32, Hex conversions option in the Encoding preferences page (on page 169) will be used. Likewise, the same is true if the Show the dialog box for choosing the encoding for Base64, Base32, Hex conversions option is not selected in the Messages preference page (on page 308).

**Modify All Matches**

Use this option to modify (in-place) all the occurrences of the selected content (or the contiguous fragment where the cursor is located). When you use this option, a thin rectangle replaces the highlights and allows you to start editing. If matches with different letter cases are found, a dialog box is displayed that allows you select whether you want to modify only matches with the same letter case or all matches.

**Join and Normalize Lines (Ctrl + J (Command + J on macOS))**

For the current selection, this action joins the lines by replacing the line separator with a single space character. It also normalizes the whitespaces by replacing a sequence of such characters with a single space.

**Insert new line after (Ctrl + Alt + Enter (Command + Alt + Enter on macOS))**

This action has the same result as moving the cursor to the end of the current line and pressing the ENTER key.

**Modify All Matches**

Use this option to modify (in-place) all the occurrences of the selected content (or the contiguous fragment where the cursor is located). When you use this option, a thin rectangle replaces the highlights and allows you to start editing. If matches with different letter cases are
found, a dialog box is displayed that allows you select whether you want to modify only matches with the same letter case or all matches.

Open submenu

The following actions are available in this submenu:

Open File at Cursor

Opens the file at the cursor position in a new panel. If the file path represents a directory path, it will be opened in system file browser. If the file at the specified location does not exist, an error dialog box is displayed and it includes a Create new file button that starts the New document wizard. This allows you to choose the type or the template for the file. If the action succeeds, the file is created with the referenced location and name and is opened in a new editor panel. If the file is an image file, it will be opened in the Image Preview pane (on page 469).

Open File at Cursor in System Application

Opens the file (identified by its link) or web page (identified by a web link) found at the cursor position. The target is opened in the default system application associated with that file type.

Compare

Opens the current file in the Compare Files tool (on page 474).

Show/Hide Preview

A toggle action that shows or hides the Preview pane.

Export as DITA Topic

Converts the current Markdown document into a DITA topic.

Export as XDITA Topic

Converts the current Markdown document into a Lightweight DITA XML topic.

Export as HTML

Converts the current Markdown document into an XHTML document.

Print (Available in the Preview pane)

Opens a page setup dialog box that allows you to configure printing options for the current document.

Related information

Markdown Editor (on page 1263)

Working with Markdown Documents in DITA (on page 3024)

Markdown Editor Syntax Rules and Specifications (on page 1277)
Syntax Highlighting in the Markdown Editor

Oxygen XML Editor supports syntax highlighting in the Markdown editor to make it easier to read the semantics of the structured content by displaying each type of XML code in different colors and fonts.

To customize the colors or styles used for the syntax highlighting colors for Markdown documents, follow these steps:

1. Open the Preferences dialog box (Options > Preferences) (on page 124).
2. Go to Editor > Syntax Highlight (on page 224).
3. Select and expand the Markdown section in the top pane.
4. Select the component you want to change and customize the colors or styles using the selectors to the right of the pane.

You can see the effects of your changes in the Preview pane.

Related Information:

Syntax Highlight Preferences (on page 224)
Markdown Editor (on page 1263)

Automatic Validation in Markdown Documents

Markdown documents are validated automatically as you type. The conversion engine constantly tries to parse your changes to display the results in the Preview pane. If a change results in an error that prevents the parser from converting the syntax, an error message will be displayed in the DITA tab or in the Results view (on page 548) at the bottom of the editor.

Examples of the type of errors that will be reported include headers being in the wrong order or the syntax of a document begins with something other than a 1st level header.

Validating Markdown Documents with Schematron

It is possible to validate Markdown documents with Schematron rules. There are two ways to create an association between Markdown documents and Schematron files:

- You can configure an association using the Markdown preferences page (on page 275). You can specify a Schematron file to validate converted HTML content, as well as one to validate converted DITA content.
- You can create a Schematron association for Markdown documents by adding a catalog mapping (on page 826) for one of the following URIs:
  - [http://www.oxygenxml.com/schematron/validation/markdown-as-html](http://www.oxygenxml.com/schematron/validation/markdown-as-html) - The obtained Schematron will be applied over HTML conversions.
  - [http://www.oxygenxml.com/schematron/validation/markdown-as-dita](http://www.oxygenxml.com/schematron/validation/markdown-as-dita) - The obtained Schematron will be applied over DITA conversions.
The catalog mapping is a fallback in case the Schematron validation is disabled in the Markdown preferences page (on page 275) or the path to the Schematron file is empty.

**Warning:**
If you are using a custom version of DITA-OT (on page 269), the mapping information might not be generated properly, causing issues with the Schematron validation. For example, error locations may not be accurate or synchronization may fail.

**Tip:**
Inside the samples folder, there is a schematron-validation folder with some files you can use to see how Schematron validation can be done with Markdown files. The path of the folder is: [OXYGEN_INSTALL_DIR]/samples/markdown/schematron-validation/.

Related Information:
Markdown Editor (on page 1263)

**Working with Markdown Documents in DITA**

Oxygen XML Editor includes some unique features that allow you to easily integrate Markdown documents in a DITA project. This is especially helpful for teams that have contributors who are familiar with the Markdown syntax, but they want their output to be generated from DITA projects. The integration between the Markdown editor and DITA includes actions to export or convert Markdown documents to DITA topics and the DITA tab in the Preview pane provides a visualization of how the DITA topic will look after conversion. Likewise, the XDITA tab in the Preview pane provides a visualization of how a Lightweight DITA topic will look after conversion. Keys that are defined in the root map are also resolved in the Preview pane.

**Export Markdown as a DITA Topic**

The Markdown editor includes an option to quickly convert the current Markdown document into a DITA topic. The Export as DITA Topic action is available in the contextual menu.

The conversion creates a new XML file that is defined as a DITA topic and opens it in the Text editing mode. You can then work with the document as you would with any other DITA topic, although you may need to manually correct some issues where the parser could not properly map Markdown syntax to DITA markup.

**Working with Markdown Documents in the DITA Maps Manager**

Oxygen XML Editor has some specialized features that allow you to integrate Markdown documents directly into your DITA project using the DITA Maps Manager (on page 2897). The following features are available for Markdown documents in the DITA Maps Manager view:
• **Insert Reference to Markdown Document** - You can use the New, Reference, and Reference to the currently edited file actions from the Append Child, Insert Before, or Insert After submenu when invoking the contextual menu in the DITA Maps Manager to insert a reference to a Markdown document at the selected location in the map. Markdown documents will be inserted as a topic reference (\texttt{topicref} element) with the \texttt{format} attribute set to \texttt{markdown}.

• **Validate Markdown Documents in DITA Maps** - When you use the Validate and Check for Completeness action from the DITA Maps Manager toolbar to check the integrity of the structure of a DITA map, Markdown documents that are referenced in the DITA map will be converted to DITA topics in the background and validated the same as any other DITA topic.

• **Transforming DITA Maps with Markdown Documents** - When transforming DITA maps that have Markdown documents referenced, the transformation will convert the Markdown documents to normal DITA output without you needing to manually convert the Markdown documents to DITA topics.

• **Manually Convert Markdown Documents to DITA Topics** - If you need to use DITA semantics that are not possible in Markdown syntax (such as content references, related links, and other DITA-specific syntax), you can manually convert the Markdown document into a DITA topic. To do so, right-click the Markdown document in the DITA Maps Manager and select Refactoring > Convert Markdown to DITA Topic. This will open a dialog box that allows you to configure options for converting the document to an XML file that is defined as a DITA topic.

**Figure 458. Convert Markdown to DITA Topic Dialog Box**

This dialog box includes the following options:

**Destination**
- The destination path for the new DITA topic.

**File Name**
- Presents the current name and allows you to change it.

**Update references**
- Select this option to update all references of the file in the DITA map and in the files referenced from the DITA map.

**Delete Markdown file**
If selected, the Markdown version of the file is deleted when the document is converted into a DITA file. If deselected (default value), when the document is converted into a DITA file, the original Markdown file is also preserved in its current location.

**Preview**

Select this button to display a preview of the changes Oxygen XML Editor is about to make.

**Convert**

Select this button to perform the conversion. If the Markdown file has `format="markdown"`, it will be converted to a DITA topic. If it has `format="mdita"`, it will be converted to a LightWeight DITA topic.

---

Tip:

Oxygen XML Editor comes with a sample ditamap project for converting Markdown to DITA. Go to the Project view (on page 402), open the sample.xpr project, and navigate to the dita/markdown-dita folder.

---

**Converting Multiple Markdown Documents to DITA**

Oxygen XML Editor offers an add-on that contributes actions in the Tools menu and contextual menu to enable batch conversion between various formats, including Markdown to DITA. For more information and instructions for installing the add-on, see Batch Documents Converter Add-on (on page 2533).

**DITA-Related Markdown Syntax**

For a list of Markdown rules and syntax examples that are specific to DITA, see the Markdown DITA Syntax Reference.

---

**Related information**

- Markdown Editor (on page 1263)
- Actions Available in the Markdown Editor (on page 1266)
- Markdown Editor Syntax Rules and Specifications (on page 1277)
- Automatic Validation in Markdown Documents (on page 1274)
- Markdown DITA Syntax Reference

---

**Markdown Editor Syntax Rules and Specifications**

The Markdown editor in Oxygen XML Editor uses rules that were integrated from the most common set of default Markdown syntax rules along with many of the GitHub Flavored Markdown rules.

This topic lists the Oxygen XML Editor implementation of the most commonly used syntax rules.
Headers

The Markdown editor supports two styles of headers, **Setext** and **Atx**.

- **Setext Style**

  Setext-style headers are underlined using equal signs (for first-level headers) and dashes (for second-level headers). Any number of equal signs or dashes will result in the same output.

  **Example: Setext Style Headers**

  ```markdown
  First-Level Header (H1)
  =========
  
  Second-Level Header (H2)
  ===========
  ```

- **Atx Style**

  Atx-style headers use 1-6 hash characters at the start of the line, corresponding to header levels 1-6. Optionally, you may close atx-style headers. This is purely cosmetic and the closing hashes do not need to match the number of hashes used to open the header. It is the number of opening hashes that determines the header level.

  **Example: Atx Style Headers**

  ```markdown
  # H1 text #
  ## H2 text
  ### H3 text ######
  #### H4 text
  ##### H5 text ######
  ####### H6 text
  ```

Horizontal Rules (for HTML output only)

You can produce a horizontal rule tag (`<hr>`) by placing three or more hyphens, asterisks, or underscores on a line by themselves (they also need to be preceded and followed by a blank line). Optionally, they can be separated by spaces.

**Example: Horizontal Rules**

```markdown
* * *
*****
```
```
Paragraphs and Line Breaks

A paragraph is simply one or more consecutive lines of text, separated by one or more blank lines. The text at the beginning of a paragraph should not be indented with spaces or tabs. To create a new paragraph, simply insert a blank line in between them.

Important:

When converting to HTML, if you break a paragraph on multiple lines (without a blank line in between them), it will create a break tag (<br>). When converting to DITA, the text is kept in a single paragraph in this case and a blank line is required to break a paragraph. This behavior differs slightly from the default Markdown rules.

Example: Paragraphs

This is a paragraph that contains
two lines of text. (In HTML, a break tag is created in between the two lines)

This is a new paragraph.

Styling Text

The Markdown editor supports some syntax rules for styling text (such as bold, italic, or strikethrough).

• **Italic (Emphasis)**

Text wrapped with one asterisk or underscore produces an italic (emphasis) tag.

*italic*
_italic_

• **Bold (Strong)**

Text wrapped with two asterisks or underscores produces a bold (strong) tag.

**bold**
__bold__

• **Strikethrough**

In HTML only, text wrapped with two tildes (~) produces a strikethrough tag.

~~strikethrough~~

• **Underline**

Text wrapped with two plus signs (+) produces an underline tag.

++underline++
Tip:
You can also combine these styling rules. For example, "**BoldText _ItalicText_ BoldText**" would produce italicized text within bold text. Also, if you surround an asterisk or underscore with spaces, it will be treated as a literal asterisk or underscore. To produce a literal asterisk or underscore at a position where it would otherwise be used as a styling delimiter, you can escape it with a backslash (for example, \"literal asterisks\").

Links

The Markdown editor supports two types of links, inline and reference. In both cases, it begins with link text that is delimited by [square brackets].

- **Inline Links**

  To create an inline link, use a set of regular parentheses immediately after the closing square bracket for the link text. Inside the parentheses, put the URL where you want the link to point, and optionally a title surrounded in quotes. Also, if you reference a local resource on the same server, you can use relative paths.

  **Examples: Inline Link**

  With a title:

  Text with [example link text](http://www.example.com/path "Title") inline link and title.

  Without a title:

  Text with [example link text](http://www.example.com/path) inline link without a title.

  Relative path:

  Text with [example link text](/relative_path/) inline link with relative path.

- **Reference Links**

  Reference-type links use a second set of square brackets that include a label (link identifier) to reference the target for the link (link identifier may consist of letters, numbers, spaces, and punctuation and it is not case-sensitive). You can optionally use a space to separate the sets of brackets. The labels (link identifiers) are only used for creating the links and do not appear in the output.

  **Examples: Reference Link**

  Text with [link text1][id 1] a reference-type link and [link text2][id_2] another one.

  Then, somewhere in the document, you need to define your link label on a line by itself. The link identifier must be within square brackets followed by a colon, then after one or more spaces the URL for the link. Optionally this can be followed by a title enclosed in single quotes, double quotes, or parentheses. Also, the link may optionally be enclosed in angle brackets (< >).

  [id 1]: http://example1.com/ "Optional Title"
  [id_2]: <http://example2.com/> "Optional Title2"
Other notes about Reference Links:

◦ You can put the title on a second line and use extra spaces or tabs for padding. This is useful for aesthetics when the URL is long.

```
[id]: http://example.com/long/path/to/resource/here
"Optional Title Here"
```

◦ The label (link identifier) can be missing, in which case the link text (in square brackets) is used as the name.

```
[My Link][]
```

and then defined as:

```
[My Link]: http://example.com/
```

Automatic Links

The Markdown editor supports a shortcut style for creating automatic links for URLs and email addresses. You simply surround the URL or email address with angle brackets.

Note:

These automatic links only work properly in HTML conversions. The Preview pane may display them properly in the DITA tab, but the DITA output will not properly recognize the format.

• URLs

By surrounding a URL with angle brackets, you can show the actual text of the URL while also making it clickable in the output.

```
<http://example.com/>
```

For example, in HTML it is converted to:

```
<a href="http://example.com/>http://example.com</a>
```

• Email Addresses

Automatic links for email addresses work similarly, except that Markdown will also perform a bit of randomized decimal and hex entity-encoding to help obscure your address from address-harvesting spambots.

```
<address@example.com>
```

In HTML, it is converted to something like:

```
<a href="\&#x6D;ailto:addr\#115;es@example.co\#111;">\#115;\#115;\#64;\#101;\#120;\#70;\#x6C;e\#2E;\#99;\#111;
```
Images

The Markdown editor uses an image syntax that is intended to resemble the syntax for two types of links (inline and reference). In both cases, the syntax for images begins with an exclamation mark, followed by `Alt` attribute text surrounded by square brackets, and then followed by a set of parentheses that contain the URL or path to the image.

- **Inline Images**

  For inline images, use a set of regular parentheses immediately after the closing square bracket for the `Alt` attribute text. Inside the parentheses, put the URL or path of the image, and optionally a title surrounded in quotes.

  **Examples: Inline Images**

  With a title:

  ```markdown
  Text with ![Alt text](/path/to/img.jpg "Optional title") inline image and a title.
  ```

  Without a title:

  ```markdown
  Text with ![Alt text](/path/to/img.jpg) inline link without a title.
  ```

- **Reference Images**

  For reference-type images, use a second set of square brackets that include a label (image identifier) to identify the image (it may consist of letters, numbers, spaces, and punctuation and it is not case-sensitive). You can optionally use a space to separate the sets of brackets. The labels (image identifiers) do not appear in the output.

  ```markdown
  Text with ![Alt text][id] a reference-type image.
  ```

  Then, somewhere in the document, you need to define your image label on a line by itself. The image identifier must be within square brackets followed by a colon, then after one or more spaces the URL or path of the image. Optionally this can be followed by a title enclosed in single quotes, double quotes, or parentheses.

  ```markdown
  [id]: url/to/image "Optional Title"
  ```

Blockquotes

The Markdown editor uses email-style greater than characters (>) for blockquotes. You only need to put the > before the first line of a hard-wrapped paragraph, but it looks better (and is clearer) if you put a > before every line.
• **Example: Blockquotes**

> This is a blockquote with two paragraphs. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Aliquam hendrerit mi posuere lectus. Vestibulum enim wisi, viverra nec, fringilla in, laoreet vitae, risus.

> Donec sit amet nisl. Aliquam semper ipsum sit amet velit. Suspendisse id sem consectetur libero luctus adipiscing.

• **Example: Nested Blockquotes**

Blockquotes can be nested by adding additional levels of > characters.

> This is the first level of quoting.

> > This is nested blockquote.

> Back to the first level.

• **Example: Blockquotes with Other Markdown Elements**

Blockquotes can also contain other Markdown elements (such as headers, lists, and code blocks).

> ## This is a header.

> 1. This is the first list item.

> 2. This is the second list item.

> Here's some example code:

> return shell_exec("echo $input | $markdown_script")

**Quoting Code (Inline and Code Blocks)**

The Markdown editor supports quoting code or commands inline within a sentence or in distinct blocks.

• **Inline**

You can quote or emphasize code within a sentence (inline) with single backticks (‘`). The text within the backticks will not be formatted.

**Example: Inline Code Emphasis**

This is a normal sentence with a `code` in the middle.

• **Code Blocks**
You can format code or text into its own distinct block by inserting a blank line before and after the content and using at least 4 spaces (or 1 tab), or by using opening and closing triple backticks (```) on separate lines.

**Example: Code Block**

```markdown
This is a normal paragraph:

This is a code block

This is a normal paragraph:

...

This is a code block

...
```

One level of indentation is removed from each line of a codeblock and it continues until it reaches a line that is not indented (or until the closing backticks).

**Example: Code Block with Indentation**

```
tell application "something"
  beep
end tell
```

For example, in HTML the result would look like this:

```html
<pre><code>tell application "Foo"
  beep
end tell</code></pre>

You can also add an optional language identifier to enable syntax highlighting in your code blocks. The Oxygen XML Editor Markdown editor syntax highlight supports the following languages: Java, JavaScript, CSS, and Python. When publishing Markdown content as part of a DITA project, more language values are supported and produce syntax highlights in the published WebHelp or PDF outputs: Adding Syntax Highlights for Codeblocks in the Output *(on page 1663)*.

**Example: Syntax Highlighting in Code Block**

```css
input[type="submit"] {
  color: white;
  font-weight: bold;
...
```
Inline XHTML (for HTML output only)

The Markdown editor supports writing inline XHTML. Since Markdown is just a writing format, it requires a conversion for publishing purposes. If you are using the HTML conversion, for any markup that is not covered by Markdown syntax, you can simply use XHTML syntax.

**Example: Inline XHTML**

```html
This is a regular paragraph.

```
<table>
  <tr>
    <td>Col 1</td>
    <td>Col 2</td>
  </tr>
</table>

This is another regular paragraph.
```

Lists

The Markdown editor supports ordered and unordered lists. You can also insert blockquotes (on page 1282) and code blocks (on page 1283) inside list items. List markers typically start at the left margin, but may be indented by up to three spaces.

- **Unordered Lists**

  For unordered lists, you can use asterisks (*), plus signs (+), and hyphens (-) interchangeably.

  ```markdown
  * List item 1
  + List item 2
  - List item 3
  ```

- **Ordered Lists**

  For ordered lists, use numbers followed by periods. The actual numbers you use have no effect on the output. It simply converts them to list items within an ordered list and the actual number of list items will determine the numbers in the output.

  ```
  1. List item 1
  8. List item 2
  5. List item 3
  ```

- **Nested Lists**

  You can create nested lists by indenting lines by three spaces.

  ```
  1. Ordered list item 1
     1. Nested ordered list item 1
  ```
2. Nested ordered list item 2
   * 2nd level nested unordered list item 1
   * 2nd level nested unordered list item 2
   * 3rd level nested unordered list item 1
2. Ordered list item 2

- **Paragraphs Inside Lists**

If list items are separated by blank lines, Markdown will wrap the items in a paragraph in the output.

* List item 1

* List item 2

For both DITA and HTML output, this would result in:

```html
<ul>
  <li><p>List item 1</p></li>
  <li><p>List item 2</p></li>
</ul>
```

- **Multiple Paragraphs Inside Lists**

List items may consist of multiple paragraphs. Each subsequent paragraph in a list item must be indented by either 4 spaces or one tab. Optionally, you can also indent each line of a paragraph to make it look nicer.

1. This is a list item with two paragraphs. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Aliquam hendrerit mi posuere lectus.

   Vestibulum enim wisi, viverra nec, fringilla in, laoreet vitae, risus. Donec sit amet nisl. Aliquam semper ipsum sit amet velit.

2. Suspendisse id sem consectetuer libero luctus adipiscing.

- **Blockquotes Inside Lists**

To put a *blockquote* within a list item, the blockquote delimiters (>) need to be indented so that they are under the first letter of the text after the list item marker.

* A list item with a blockquote:
  
  > This is a blockquote
  > inside a list item.

- **Code Blocks Inside Lists**
To put a code block within a list item, insert an empty line in between the list item and the code block, and the code block needs to be indented twice (with 8 spaces or 2 tabs), or if you are using the triple backticks method, the opening triple backtick needs to be indented with 4 spaces or 1 tab.

```
*   A list item with a code block:

    This is a code block inside a list item

    ...

    This is a code block inside a list item using the backticks method
    ...
```

### Task Lists

You can create task lists by prefacing list items with a hyphen followed by a space followed by square brackets (- [ ]). To mark a task as complete, use - [x].

**Example: Task Lists**

- [ ] Unfinished task 1
- [x] Finished task 2

### Definition Lists

You can create definition lists by using a colon plus a space for each list item.

**Example: Definition Lists**

```
Term 1
: Definition A
: Definition B
```

### Tables

You can create tables in the Markdown editor by using pipes (|) and hyphens (–).

**Creating a Table**

Pipes are used to separate each column, while hyphens are used to create column headers. The pipes on either end of the table are optional. Cells can vary in width and do not need to be perfectly aligned within columns, but there must be at least three hyphens in each column of the header row.

```
<table>
<thead>
<tr>
<th>First Header</th>
<th>Second Header</th>
</tr>
</thead>
<tbody>
<tr>
<td>------------</td>
<td>-------------</td>
</tr>
<tr>
<td>Column 1 Row 1 Cell</td>
<td>Column 2 Row 1 Cell</td>
</tr>
<tr>
<td>Column 1 Row 2 Cell</td>
<td>Column 2 Row 2 Cell</td>
</tr>
</tbody>
</table>
```

**Formatting Rules in Table Cells**
You can use formatting rules inside the cells of the table (such as links, inline code blocks, and text styling).

<table>
<thead>
<tr>
<th>First Header</th>
<th>Second Header</th>
</tr>
</thead>
<tbody>
<tr>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>'inline code'</td>
<td>Content with <strong>bold text</strong> inside cell</td>
</tr>
</tbody>
</table>

**Aligning Text in Tables**

You can align text to the left, right, or center of a column by including colons (:) to the left, right, or on both sides of the hyphens within the header row.

<table>
<thead>
<tr>
<th align="left">Left-aligned</th>
<th align="center">Center-aligned</th>
<th align="right">Right-aligned</th>
</tr>
</thead>
<tbody>
<tr>
<td align="left">Content Cell</td>
<td align="center">Content Cell</td>
<td align="right">Content Cell</td>
</tr>
</tbody>
</table>

**Joining Cells (Span a Cell Over Multiple Columns)**

You can join cells horizontally (span a cell over multiple columns) by using multiple consecutive pipe characters (|) to the right of the particular cell. The number of consecutive pipes indicate the number of columns the cell will span (| for two, || for three, and so on).

<table>
<thead>
<tr>
<th>First Header</th>
<th>Second Header</th>
<th>Third Header</th>
<th>Fourth Header</th>
</tr>
</thead>
<tbody>
<tr>
<td>Content Cell</td>
<td><em>Cell Span Over 3 Columns</em></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Emoji**

You can add *emoji* in the Markdown editor by surrounding the EMOJICODE with colons (:EMOJICODE:).

**Example: Emoji**

:smile:
:laughing:

The resulting emoticons will appear in the output, but they are not displayed in the *Preview* pane.

For a full list of available emoji codes, see [Emoji Cheat Sheet](#).

**Backslash Escapes**

You can ignore Markdown formatting by using backslash escapes (\) to generate literal characters that would otherwise have special meaning in the Markdown syntax. For example, if you want to surround a word with literal asterisks (instead of an italic or emphasis tag), you can use backslashes to escape the asterisks.

\"*literal asterisks*\"

The Markdown editor provides backslash escapes for the following characters:
Automatic Escaping for Special Characters

The Markdown editor includes support for automatically escaping special characters such as angle brackets (< >) and ampersands (&). If you want to use them as literal characters, you must escape them as entities, as in the table below. The exception to this is in HTML output, if the special characters for a valid tag (for example, <b>), they are treated as literal characters and escaping is not necessary.

<table>
<thead>
<tr>
<th>Literal Character</th>
<th>Escaping Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;</td>
<td>&lt;</td>
</tr>
<tr>
<td>&gt;</td>
<td>&gt;</td>
</tr>
<tr>
<td>&amp;</td>
<td>&amp;</td>
</tr>
</tbody>
</table>

Footnotes

The Markdown editor in Oxygen XML Editor supports normal and inline footnotes. The following examples show the required syntax.

• **Example: Normal Footnote**

Here is a footnote reference,[^1]

[^1]: Here is the footnote.

• **Example: Normal Footnote with Multiple Blocks**

Here is a footnote reference,[^longnote]

[^longnote]: Here is the footnote with multiple blocks.

Subsequent paragraphs are indented with 4 spaces or 1 tab to show that they belong to the previous footnote.
**Example: Inline Footnote**

Here is an inline note.^[Inlines notes are easier to write, since you don't have to pick an identifier and move down to type the note.]

**DITA-Related Markdown Syntax**

For information about unique Markdown features for DITA projects, see [Working with Markdown Documents in DITA (on page 3024)](on page 3024). Also, for a list of Markdown rules and examples for DITA, see the [Markdown DITA Syntax Reference](on page 3024).

**Related information**

- Default Markdown Syntax
- GitHub Flavored Markdown Rules
- Markdown Editor *on page 1263*
- Actions Available in the Markdown Editor *on page 1266*
- Working with Markdown Documents in DITA *on page 3024*
9.

Built-in XML Frameworks (Document Types)

Oxygen XML Editor includes a variety of specialized editors, views, and features that are dynamic according to the type of document that you open or create. Oxygen XML Editor includes fully supported built-in frameworks (on page 3219) for the most popular XML document types (DITA, DocBook, TEI, XHTML, JATS) with a full set of features, as well as support for other document types with more generic features (on page 516).

The built-in frameworks (on page 3219) are defined according to a set of rules and a variety of settings that improve editing capabilities for its particular file type. These settings include:

• A default grammar used for validation and content completion in both Author mode and Text mode.
• CSS stylesheets for rendering XML documents in Author mode.
• User actions invoked from toolbars or menus in Author mode.
• Built-in transformation scenarios used for publishing XML documents.
• XML Catalogs (on page 3225) used for mapping resources.
• New document templates to make it easy to create XML documents.
• User-defined extensions for customizing the interaction with the content author in Author mode.

It is also possible to create and configure your own custom frameworks (document types). For more information, see the Creating and Configuring Custom Frameworks (on page 2133) section.

For extensive details about the DITA editing features included in Oxygen XML Editor, see the DITA Authoring chapter (on page 2886).

DocBook 4 Document Type (Framework)

DocBook is a very popular set of tags for describing books, articles, and other prose documents, particularly technical documentation. DocBook provides a vast number of semantic element tags, divided into three broad categories: structural, block-level, and inline. DocBook content can then be published in a variety of formats, including HTML, PDF, WebHelp, and EPUB.

File Definition

A file is considered to be a DocBook 4 document when one of the following conditions are true:

• The root element name is <book> or <article>.
• The PUBLIC ID of the document contains the string -//OASIS//DTD DocBook XML.
Default Document Templates

There are a variety of default *DocBook 4* templates available when creating new documents from templates *(on page 368)* and they can be found in: Framework Templates > *DocBook 4*.

The default templates for DocBook 4 documents are located in the

*[OXYGEN_INSTALL_DIR]/frameworks/docbook/templates/Docbook 4* folder.

Default Schema for Validation and Content Completion

The default schema that is used if one is not detected in the DocBook 4 file is *docbookxi.dtd* and it is stored in *[OXYGEN_INSTALL_DIR]/frameworks/docbook/4.5/dtd/*.

Default CSS

The default CSS files used for rendering DocBook content in Author mode are stored in

*[OXYGEN_INSTALL_DIR]/frameworks/docbook/css/*.

By default, these default CSS files are merged with any that are specified in the document. For more information, see Configuring and Managing Multiple CSS Styles for a Framework *(on page 2145)*.

Default XML Catalog

The default *XML Catalog (on page 3225)*, *catalog.xml*, is stored in

*[OXYGEN_INSTALL_DIR]/frameworks/docbook/*.

Transformation Scenarios

Oxygen XML Editor includes numerous built-in DocBook transformation scenarios that allow you to transform DocBook 4 documents to a variety of outputs, such as WebHelp, PDF, HTML, HTML Chunk, XHTML, XHTML Chunk, and EPUB. All of them are listed in the *DocBook 4* section in the Configure Transformation Scenario(s) dialog box *(on page 1548)*.

For more information, see the DocBook Transformation Scenarios *(on page 1458)* section.

Resources

- Oxygen Video Tutorial: Editing DocBook Documents in Author Mode
- DocBook Specifications

Related Information:

- Editing XML Documents in Author Mode *(on page 588)*
- Editing XML Documents in Text Mode *(on page 517)*
- Adding Tables in DocBook *(on page 686)*
DocBook 4 Author Mode Actions

A variety of actions are available for DocBook 4 documents in the DocBook4 menu, toolbar, contextual menu, and the Content Completion Assistant (on page 3217).

DocBook 4 Toolbar Actions

The following default actions are available on the DocBook toolbar when editing in Author mode (by default, most of them are also available in the DocBook4 menu and in various submenus of the contextual menu):

- **Bold**
  Emphasizes the selected text by surrounding it with a *bold* tag. You can use this action on multiple non-contiguous selections.

- **Italic**
  Emphasizes the selected text by surrounding it with an *italic* tag. You can use this action on multiple non-contiguous selections.

- **Underline**
  Emphasizes the selected text by surrounding it with an *underline* tag. You can use this action on multiple non-contiguous selections.

- **Link Actions Drop-Down Menu**

  The following link actions are available from this menu:

  - **Cross reference (link)**
    Opens a dialog box that allows you to select a target to insert as a hypertext link.

  - **Cross reference (xref)**
    Inserts a cross reference to other parts of the document.

  - **Web Link (ulink)**
    Inserts a link that addresses its target with a URL (Universal Resource Locator).

  - **Insert OLink**
    Opens an OLink dialog box that allows you to insert a link that addresses its target indirectly, using the values of the @targetdoc and @targetptr attributes that are present in a Targetset (on page 1336) file.
After you choose the Targetset URL, the structure of the target documents is presented. For each target document (@targetdoc), its content is displayed allowing you to easily identify the @targetptr for the <olink> element that will be inserted. You can also use the search fields to quickly identify a target. If you already know the values for targetdoc and targetptr, you can insert them directly in the corresponding fields. You can also edit an <olink> using the Edit OLink action that is available on the contextual menu. The last used Targetset URL will be used to identify the edited target.

To insert XREF text into the <olink>, enter the text in the xreftext field and make sure the Insert xreftext in the OLink option is selected.

**Insert URI**

Inserts a URI element. The URI identifies a Uniform Resource Identifier (URI) in content.

**Edit OLink**

Opens a dialog box that allows you edit an existing OLink. See the Insert OLink action for more information.

**Insert Image**

Opens a dialog box that allows you to select the path of an image to insert at the cursor position (on page 719). Depending on the current location, an image-type element is inserted. If the action is invoked between two block elements (such as paragraphs), the dialog box also allows you to provide a title.
**Insert Media Resource**

Opens a **Choose Media dialog box (on page 747)** that allows you to select the URL of a media object to be inserted into a document at the cursor position. The result will be that a reference to the specified video, audio, or embedded HTML frame is inserted and rendered in **Author** mode so that it can be played directly from there.

**Insert XInclude**

Opens a dialog box that allows you to browse and select content to be included and automatically generates the corresponding XInclude instruction.

**Section Drop-Down Menu**

The following actions are available from this menu:

1. **Insert Section**
   
   Inserts a new section or subsection in the document, depending on the current context. For example, if the current context is `<sect1>`, then a `<sect2>` is inserted. By default, this action also inserts a `<para>` element as a child node. The `<para>` element can be deleted if it is not needed.

2. **Promote Section** (*Ctrl + Alt + LeftArrow (Command + Alt + LeftArrow on macOS)*)

   Promotes the current node as a sibling of the parent node.

3. **Demote Section** (*Ctrl + Alt + RightArrow (Command + Alt + RightArrow on macOS)*)

   Demotes the current node a child of the previous node.

4. **Insert Paragraph**

   Insert a new paragraph element at current cursor position.

5. **Insert Equation**

   Opens the **XML Fragment Editor** that allows you to insert and edit MathML notations (on page 749).

6. **Insert List Item**

   Inserts a list item in the current list type.

7. **Insert Ordered List**

   Inserts an ordered list at the cursor position. A child list item is also automatically inserted by default. You can also use this action to convert selected paragraphs or other types of lists to an ordered list.

8. **Insert Itemized List**

   Inserts an itemized list at the cursor position. A child list item is also automatically inserted by default. You can also use this action to convert selected paragraphs or other types of lists to an itemized list.
**Insert Variable List**

Inserts a DocBook variable list. A child list item is also inserted automatically by default. You can also use this action to convert selected paragraphs or other types of lists to a variable list.

**Insert Procedure List**

Inserts a DocBook `<procedure>` element. A `<step>` child element is also inserted automatically. You can also use this action to convert selected paragraphs or other types of lists to a procedure list.

**Sort**

Sorts cells or list items in a table.

**Insert Table**

Opens a dialog box that allows you to configure and insert a table. You can generate a header and footer, set the number of rows and columns of the table and decide how the table is framed. You can also use this action to convert selected paragraphs, lists, and inline content (mixed content, text plus markup, that is rendered inside a block element (on page 3216)) into a table, with the selected content inserted in the first column, starting from the first row after the header (if a header is inserted).

Note:

If the selection contains a mixture of elements that cannot be converted, you will receive an error message saying that **Only lists, paragraphs, or inline content can be converted to tables**.

**Insert Row**

Inserts a new table row with empty cells below the current row. This action is available when the cursor is positioned inside a table.

**Delete Row(s)**

Deletes the table row located at the cursor position or multiple rows in a selection.

**Insert Column**

Inserts a new table column with empty cells after the current column. This action is available when the cursor is positioned inside a table.

**Delete Column(s)**

Deletes the table column located at the cursor position or multiple columns in a selection.

**Table Properties**

Opens the Table properties dialog box that allows you to configure properties of a table (such as frame borders).

**Join Cells**

Joins the content of the selected cells (both horizontally and vertically).
**Split Cell**

Splits the cell at the cursor location. If Oxygen XML Editor detects more than one option to split the cell, a dialog box will be displayed that allows you to select the number of rows or columns to split the cell into.

**DocBook4 Contextual Menu Actions**

The following actions are available in the contextual menu when editing in **Author** mode (most of them are also available in the **DocBook4** menu at the top of the interface):

- **Add File to Review Task**:
  This action can be used to add the current document to a task in the **Content Fusion Tasks Manager** view. **Oxygen Content Fusion** is a flexible, intuitive collaboration platform designed to adapt to any type of documentation review workflow. This functionality is available through a pre-installed connector add-on *(on page 2527)*. To fully take advantage of all of the benefits and features of **Content Fusion**, your organization will need an **Oxygen Content Fusion Enterprise Server**. For more information, see the **Oxygen Content Fusion** website.

- **Edit Attributes**:
  Displays an **in-place attributes editor (on page 630)** that allows you to manage the attributes of an element.

- **Edit Profiling Attributes**:
  Allows you to change the **profiling attributes (on page 668)** defined on all selected elements.

- **Cut (Ctrl + X (Command + X on macOS))**:
  Removes the currently selected content from the document and places it in the clipboard.

- **Copy (Ctrl + C (Command + C on macOS))**:
  Places a copy of the currently selected content in the clipboard.

- **Paste (Ctrl + V (Command + V on macOS))**:
  Inserts the current clipboard content into the document at the cursor position.

- **Paste special submenu**:
  This submenu includes the following special paste actions:

  - **Paste As XInclude**:
    Allows you to create an `<xi:include>` element that references a DocBook element copied from **Author** mode. The operation fails if the copied element does not have a declared ID.

  - **Paste as link**
Allows you to create a `<link>` element that references a DocBook element copied from **Author** mode. The operation fails if the copied element does not have a declared ID.

**Paste as xref**

Allows you to create an `<xref>` element that references a DocBook element copied from **Author** mode. The operation fails if the copied element does not have a declared ID.

**Image Map Editor**

This action is available in the contextual menu when it is invoked on an image. This action applies an *image map* to the current image (if one does not already exist) and opens the **Image Map Editor** dialog box. This feature allows you to create hyperlinks in specific areas of an image that will link to various destinations.

**Insert submenu**

This submenu includes the following insert actions that are specific to the DocBook framework:

### Insert Table

Opens a dialog box that allows you to configure and insert a table. You can generate a header and footer, set the number of rows and columns of the table and decide how the table is framed. You can also use this action to convert selected paragraphs, lists, and inline content (mixed content, text plus markup, that is rendered inside a *block element ([on page 3216](#))*) into a table, with the selected content inserted in the first column, starting from the first row after the header (if a header is inserted).

**Note:**

If the selection contains a mixture of elements that cannot be converted, you will receive an error message saying that **Only lists, paragraphs, or inline content can be converted to tables.**

### Insert Image

Inserts an image reference ([on page 719](#)) at the cursor position. Depending on the current location, an image-type element is inserted.

### Insert Media Resource

Opens a **Choose Media** dialog box ([on page 747](#)) that allows you to select the URL of a media object to be inserted into a document at the cursor position. The result will be that a reference to the specified video, audio, or embedded HTML frame is inserted and rendered in **Author** mode so that it can be played directly from there.
Opens the XML Fragment Editor that allows you to insert and edit MathML notations (on page 749).

¶ Insert Paragraph

Inserts a new paragraph element at current cursor position.

§ Insert Section

Inserts a new section element in the document, depending on the current context.

✗ Insert XInclude

Opens a dialog box that allows you to browse and select content to be included and automatically generates the corresponding XInclude instruction.

Insert Entity

Allows you to insert a predefined entity or character entity. Surrogate character entities (range #x10000 to #x10FFFF) are also accepted. Character entities can be entered in one of the following forms:

- `#<decimal value>` - e.g. #65
- `&#<decimal value>` - e.g. &#65
- `#x<hexadecimal value>` - e.g. #x41
- `&#x<hexadecimal value>` - e.g. &#x41

§ Section submenu

The following actions are available in this submenu:

חמש Promote Section (Ctrl + Alt + LeftArrow (Command + Alt + LeftArrow on macOS))

Promotes the current node as a sibling of the parent node.

はず Demote Section (Ctrl + Alt + RightArrow (Command + Alt + RightArrow on macOS))

Demotes the current node a child of the previous node.

🔗 Link submenu

The following actions are available in this submenu:

Cross reference (link)

Opens a dialog box that allows you to select a target to insert as a hypertext link.

Cross reference (xref)

Inserts a cross reference to other parts of the document.

Web Link (ulink)

Inserts a link that addresses its target with a URL (Universal Resource Locator).

Insert OLink
Opens an **OLink** dialog box that allows you to insert a link that addresses its target indirectly, using the values of the `@targetdoc` and `@targetptr` attributes that are present in a **Targetset** *(on page 1336)* file.

**Figure 460. Insert OLink Dialog Box**

After you choose the **Targetset URL**, the structure of the target documents is presented. For each target document (`@targetdoc`), its content is displayed allowing you to easily identify the `@targetptr` for the `<olink>` element that will be inserted. You can also use the search fields to quickly identify a target. If you already know the values for `targetdoc` and `targetptr`, you can insert them directly in the corresponding fields. You can also edit an `<olink>` using the **Edit OLink** action that is available on the contextual menu. The last used **Targetset** URL will be used to identify the edited target.

To insert XREF text into the `<olink>`, enter the text in the **xreftext** field and make sure the **Insert xreftext in the OLink** option is selected.

**Insert URI**

Inserts a URI element. The URI identifies a Uniform Resource Identifier (URI) in content.

**Edit OLink**

Opens a dialog box that allows you edit an existing **OLink**. See the **Insert OLink** action for more information.

**Table actions**
The following table editing actions are available in the contextual menu when it is invoked on a table:

**Insert Rows**

Opens a dialog box that allows you to insert any number of rows and specify the position where they will be inserted (Above or Below the current row).

**Delete Row(s)**

Deletes the table row located at the cursor position or multiple rows in a selection.

**Insert Columns**

Opens a dialog box that allows you to insert any number of columns and specify the position where they will be inserted (Above or Below the current column).

**Delete Column(s)**

Deletes the table column located at the cursor position or multiple columns in a selection.

**Join Cells**

Joins the content of the selected cells (both horizontally and vertically).

**Split Cell**

Splits the cell at the cursor location. If Oxygen XML Editor detects more than one option to split the cell, a dialog box will be displayed that allows you to select the number of rows or columns to split the cell into.

**Sort**

Sorts cells or list items in a table.

**Table Properties**

Opens the Table properties dialog box that allows you to configure properties of a table (such as frame borders).

**Other Actions submenu**

This submenu give you access to all the usual contextual menu actions.

**Generate IDs**

Oxygen XML Editor generates unique IDs for the current element (or elements), depending on how the action is invoked:

- When invoked on a single selection, an ID is generated for the selected element at the cursor position.
- When invoked on a block of selected content, IDs are generated for all top-level elements and elements listed in the ID Options dialog box that are found in the current selection.
Note:
The Generate IDs action does not overwrite existing ID values. It only affects elements that do not already have an @id attribute.

Select submenu

This submenu allows you to select the following:

Element
Selects the entire element at the current cursor position.

Content
Selects the entire content of the element at the current cursor position, excluding the start and end tag. Performing this action repeatedly will result in the selection of the content of the ancestor of the currently selected element content.

Parent
Selects the entire parent element at the current cursor position.

Text submenu

This submenu contains the following actions:

To Lower Case
Converts the selected content to lower case characters.

To Upper Case
Converts the selected content to upper case characters.

Capitalize Sentences
Converts to upper case the first character of every selected sentence.

Capitalize Words
Converts to upper case the first character of every selected word.

Count Words
Counts the number of words and characters (no spaces) in the entire document or in the selection for regular content and read-only content.

Note:
The content marked as deleted with change tracking (on page 3224) is ignored when counting words.

Convert Hexadecimal Sequence to Character (Ctrl + Shift + X (Command + Shift + X on macOS))
Converts a sequence of hexadecimal characters to the corresponding Unicode character (on page 463). The action can be invoked if there is a selection containing a valid hexadecimal sequence or if the cursor is placed at the right side of a valid hexadecimal sequence. A valid hexadecimal sequence can be composed of 2 to 4 hexadecimal characters and may or may not be preceded by the `0x` or `0X` prefix. Examples of valid sequences and the characters they will be converted to:

- `0x0045` will be converted to `E`
- `0X0125` to `ĥ`
- `265` to `ɥ`
- `2190` to `←`

**Note:**
For more information about finding the hexadecimal value of a character, see Finding the Decimal, Hexadecimal, or Character Entity Equivalent (on page 466).

**Refactoring submenu**

Contains a series of actions designed to alter the XML structure of the document:

**_toggle_comment**

Encloses the currently selected text in an XML comment, or removes the comment if it is commented.

**Move Up (Alt + UpArrow)**

Moves the current node or selected nodes in front of the previous node.

**Move Down (Alt + DownArrow)**

Moves the current node or selected nodes after the subsequent node.

**Split Element (Alt + Shift + D (Ctrl + Alt + D on macOS))**

Splits the content of the closest element that contains the position of the cursor. Thus, if the cursor is positioned at the beginning or at the end of the element, the newly created sibling will be empty.

**Join Elements**

Joins two adjacent block elements (on page 3216) that have the same name. The action is available only when the cursor position is between the two adjacent block elements. Also, joining two block elements can be done by pressing the Delete or Backspace keys and the cursor is positioned between the boundaries of these two elements.

**Surround with Tags (Ctrl + E (Command + E on macOS))**
Allows you to choose a tag to enclose a selected portion of content. If there is no selection, the start and end tags are inserted at the cursor position.

- If the **Position cursor between tags option (on page 212)** is selected in the **Content Completion** preferences page, the cursor is placed between the start and end tag.
- If the **Position cursor between tags option (on page 212)** is not selected in the **Content Completion** preferences page, the cursor is placed at the end of the start tag, in an insert-attribute position.

**Surround with \*[tag\]** (Ctrl + ForwardSlash (Command + ForwardSlash on macOS))

Surround the selected content with the last tag used.

**Rename Element**

The element from the cursor position, and any elements with the same name, can be renamed according with the options from the **Rename** dialog box.

**Delete Element Tags**

Deletes the tags of the closest element that contains the position of the cursor. This operation is also executed if the start or end tags of an element are deleted by pressing the **Delete** or **Backspace** keys.

**Remove All Markup**

Removes all the XML markup inside the selected block of content and keeps only the text content.

**Remove Text**

Removes the text content of the selected block of content and keeps the markup intact with empty elements.

**Attributes Refactoring Actions**

Contains built-in XML refactoring operations that pertain to attributes with some of the information preconfigured based upon the current context.

- **Add/Change attribute**
  
  Allows you to change the value of an attribute or insert a new one.

- **Convert attribute to element**
  
  Allows you to change an attribute into an element.

- **Delete attribute**
  
  Allows you to remove one or more attributes.

- **Rename attribute**
  
  Allows you to rename an attribute.
Replace in attribute value
Allows you to search for a text fragment inside an attribute value and change the fragment to a new value.

Comments Refactoring Actions
Contains built-in XML refactoring operations that pertain to comments with some of the information preconfigured based upon the current context.

Delete comments
Allows you to delete comments found inside one or more elements.

Elements Refactoring Actions
Contains built-in XML refactoring operations that pertain to elements with some of the information preconfigured based upon the current context.

Delete element
Allows you to delete elements.

Delete element content
Allows you to delete the content of elements.

Insert element
Allows you to insert new elements.

Rename element
Allows you to rename elements.

Unwrap element
Allows you to remove the surrounding tags of elements, while keeping the content unchanged.

Wrap element
Allows you to surround elements with element tags.

Wrap element content
Allows you to surround the content of elements with element tags.

Fragments Refactoring Actions
Contains built-in XML refactoring operations that pertain to XML fragments with some of the information preconfigured based upon the current context.

Insert XML fragment
Allows you to insert an XML fragment.

Replace element content with XML fragment
Allows you to replace the content of elements with an XML fragment.
Replace element with XML fragment

 Allows you to replace elements with an XML fragment.

Review submenu

This submenu includes the following actions:

✔ Track Changes

 Enables or disables the Track Changes (on page 3224) support for the current document.

✔ Accept Change(s) and Move to Next

 Accepts the Tracked Change (on page 3224) located at the cursor position or all of the changes in a selection and then moves to the next change. If you select a part of a deletion or insertion change, only the selected content is accepted.

✔ Accept All Changes

 Accepts all Tracked Changes (on page 3224) in the current document.

 ✗ Reject Change(s) and Move to Next

 Rejects the Tracked Change (on page 3224) located at the cursor position or all of the changes in a selection and then moves to the next change. If you select a part of a deletion or insertion change, only the selected content is rejected.

 ✗ Reject All Changes

 Rejects all Tracked Changes (on page 3224) in the current document.

 ✔ Comment Change

 Opens a dialog box that allows you to add a comment to an existing Tracked Change (on page 3224). The comment will appear in a callout and a tooltip when hovering over the change. If the action is selected on an existing commented change, the dialog box will allow you to edit the comment.

 ✉ Highlight

 Enables the highlighting tool that allows you to mark text in your document.

Colors

 Allows you to select the color for highlighting text.

Stop highlighting

 Use this action to deactivate the highlighting tool.

Remove highlight(s)

 Use this action to remove highlighting from the document.

✉ Add Comment
Inserts a comment at the cursor position. The comment appears in a callout box and a tooltip (when hovering over the change).

**Show/Edit Comment**

Opens a dialog box that displays the discussion thread and allows the current user to edit comments that do not have replies. If you are not the author who inserted the original comment, the dialog box just displays the comment without the possibility of editing it.

**Remove Comment**

Removes a selected comment. If you remove a comment that contains replies, all of the replies will also be removed.

**Manage Reviews**

Opens the **Review view (on page 664)**.

**Folding submenu**

This submenu includes the following actions:

- **Toggle Fold**
  
  Toggles the state of the current fold.

- **Collapse Other Folds** *(Ctrl + NumPad/ (Command + NumPad/ on macOS))*
  
  Folds all the elements except the current element.

- **Collapse Child Folds** *(Ctrl + NumPad. (Command + NumPad. on macOS))*
  
  Folds the elements indented with one level inside the current element.

- **Expand Child Folds**
  
  Unfolds all child elements of the currently selected element.

- **Expand All** *(Ctrl + NumPad* (Command + NumPad* on macOS))*
  
  Unfolds all elements in the current document.

**About Element > ** Go to Definition

Moves the cursor to the definition of the current element.

**Inspect Styles**

Opens the **CSS Inspector view (on page 640)** that allows you to examine the CSS rules that match the currently selected element.

**Options**

Opens the **Author mode preferences page (on page 176)** where you can configure various options with regard to the Author editing mode.
Floating Contextual Toolbar for DocBook

Oxygen XML Editor includes a dynamic feature where certain editing contexts will trigger a floating toolbar with common actions that are available in the current editing context.

![Figure 461. DocBook Floating Contextual Toolbar](image)

The floating contextual toolbar is automatically displayed when editing DocBook documents in the following situations:

- When a `<para>` or `<listitem>` element has a selection inside, the floating toolbar includes actions such as **Bold**, *Italic*, _Underline_, _^Subscript_, and _^Superscript_.
- When an `<imagedata>` or `<videodata>` element is selected, the floating toolbar includes a URL chooser where you can select the appropriate target.
- When an `<olink>` element is selected, the floating toolbar includes an Edit OLink action.
- When a `<link>` or `<include>` element is selected, the floating toolbar includes a URL chooser where you can select the appropriate target.
- When a `<programlisting>` element is selected, the floating toolbar includes a drop-down control where you can select the value of the `@language` attribute.
- When an `<itemizedlist>`, `<orderedlist>`, `<variablelist>`, or `<procedure>` element is selected, the floating toolbar includes actions for converting it to a different type of list or sorting the list.
- When a `<listitem>`, `<varlistentry>`, or `<step>` element is selected, the floating toolbar includes actions for moving the item up or down in the list/procedure.
- When a `<row>` or `<tr>` element is selected in a table, the floating toolbar includes various table-related actions (such as actions for editing table properties, inserting rows, or deleting rows).
- When an `<entry>` or `<td>` element is selected in a table, the floating toolbar includes various table-related actions (such as actions for editing table properties, inserting/deleting rows, or inserting/deleting columns).
- When a `<table>` element is selected, the floating toolbar includes actions for editing table properties or sorting the table.

DocBook 4 Drag/Drop (or Copy/Paste) Actions

Dragging a file from the Project view (on page 402) or DITA Maps Manager view (on page 2897) and dropping it into a DocBook 4 document that is edited in Author mode, creates a link to the dragged file (the `<ulink>` DocBook element) at the drop location. Copy and paste actions work the same.

You can also drag images or media files from your system explorer or the Project view (on page 402) and drop them into a DocBook 4 document (or copy and paste). This will insert the appropriate element at the drop or paste location (for example, dropping/pasting an image will insert the `<inlinegraphic>` DocBook element with a `@fileref` attribute).
**Inserting an Olink in DocBook Documents**

The `<olink>` element is used for linking to resources outside the current DocBook document. The `<targetdoc>` attribute is used for the document ID that contains the target element and the `<targetptr>` attribute for the ID of the target element (the value of an `<id>` or `<xml:id>` attribute). The combination of those two attributes provides a unique identifier to locate cross references.

For example, a *Mail Administrator Guide* with the document ID `MailAdminGuide` might contain a chapter about user accounts, like this:

```xml
<chapter id="user_accounts">
  <title>Administering User Accounts</title>
  <para>blah blah</para>
</chapter>
```

You can form a cross reference to that chapter by adding an `<olink>`, as in the following example:

```xml
<olink targetdoc="MailAdminGuide" targetptr="user_accounts">user accounts</olink>
```

You may need to update your

```xml
<olink targetdoc="MailAdminGuide" targetptr="user_accounts">user accounts
</olink>
```

when you get a new machine.

To use an `<olink>` to create links between documents, follow these steps:

1. Decide which documents are to be included in the domain for cross referencing.
   
   A unique ID must be assigned to each document that will be referenced with an `<olink>`. It is usually added as an `<id>` (or `<xml:id>` for DocBook5) attribute to the root element of the document.

2. Decide on your output hierarchy.
   
   For creating links between documents, the relative locations of the output documents must be known. Before going further you must decide the names and locations of the output directories for all the documents from the domain. Each directory will be represented by an element `<dir name="directory_name">`, in the target database document.

3. Create the target database document.
   
   Each collection of documents has a main target database document that is used to resolve all `<olink>`s from that collection. The target database document is an XML file that is created once. It provides a means for pulling in the target data for each document. The database document is static and all the document data is pulled in dynamically.
Tip: Oxygen XML Editor includes a built-in new document template called DocBook Targetset Map available in the New document wizard (on page 368) that will help you get started.

Example: The following is an example of a target database document. It structures a collection of documents in a `<sitemap>` element that provides the relative locations of the outputs (HTML in this example). Then it pulls in the individual target data using system entity references to target data files that will be created in the next step.

```xml
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE targetset [ 
<!ENTITY ugtargets SYSTEM "file:///doc/userguide/target.db"> 
<!ENTITY agtargets SYSTEM "file:///doc/adminguide/target.db"> 
<!ENTITY reftargets SYSTEM "file:///doc/man/target.db"> ]>
<targetset>
  <targetsetinfo>
    Description of this target database document, which is for the examples in olink doc.
  </targetsetinfo>

  <!-- Site map for generating relative paths between documents -->
  <sitemap>
    <dir name="documentation">
      <dir name="guides">
        <dir name="mailuser">
          <document targetdoc="MailUserGuide" baseuri="userguide.html"> &ugtargets; </document>
        </dir>
      </dir>
    </dir>
    <dir name="mailadmin">
      <document targetdoc="MailAdminGuide"> &agtargets; </document>
    </dir>
    <dir name="reference">
      <dir name="mailref">
        <document targetdoc="MailReference"> &reftargets; </document>
      </dir>
    </dir>
  </sitemap>
</targetset>
```
4. Generate the target data files by executing a DocBook transformation scenario. 
Before applying the transformation, you need to edit the transformation scenario, go to the Parameters 
tab, and make sure the value of the collect.xref.targets parameter is set to yes. The default name of 
a target data file is target.db, but it can be changed by setting an absolute file path in the 
targets.filename parameter.

Example: An example of a target.db file:

```xml
<dir>
  </dir>
</sitemap>
</targetset>
```

When editing a DocBook XML document in Author mode, the Insert OLink action is available in the 
Link drop-down menu from the toolbar. This action opens the Insert OLink dialog box that 
allows you to select the target of an <olink> from the list of all possible targets from a specified target 
database document (specified in the Targetset URL field). Once a Targetset URL is selected, the 
structure of the target documents is presented. For each target document (@targetdoc), its content is 
displayed, allowing you to easily identify the appropriate @targetptr. You can also use the search fields 
to quickly identify a target. If you already know the values for the @targetdoc and @targetptr attributes, 
you can insert them directly in the corresponding fields.

Example: In the following image, the target database document is called target.xml, dbadmin is 
selected for the target document (@targetdoc), and bldinit is selected as the value for the @targetptr 
attribute. Notice that you can also add XREF text into the <olink> by using the xreftext field.
Oxygen XML Editor 24.1 | 9 - Built-in XML Frameworks (Document Types) | 1312

6. Process a DocBook transformation for each document to generate the output.
   a. Edit the transformation scenario and set the value of the `target.database.document` parameter to be the URL of the target database document.
   b. Apply the transformation scenario.

DocBook 5 Document Type (Framework)

DocBook is a very popular set of tags for describing books, articles, and other prose documents, particularly technical documentation. DocBook provides a vast number of semantic element tags, divided into three broad categories: structural, block-level, and inline. DocBook content can then be published in a variety of formats, including HTML, PDF, WebHelp, and EPUB.

File Definition

A file is considered to be a DocBook 5 document when the namespace is http://docbook.org/ns/docbook.

Default Document Templates

There are a variety of default DocBook 5 templates available when creating new documents from templates (on page 368) and they can be found in: Framework Templates > DocBook 5 > DocBook 5.0 and Framework Templates > DocBook 5 > DocBook 5.1.
New document templates for both DocBook 5 documents are located in the

\[OXYGEN\_INSTALL\_DIR]/frameworks/docbook/templates/Docbook5.0 folder.

New document templates for both DocBook 5.1 documents are located in the

\[OXYGEN\_INSTALL\_DIR]/frameworks/docbook/templates/Docbook5.1 folder.

**Default Schema for Validation and Content Completion**

The default schema that is used if one is not detected is docbookxi.rng and it is stored in
\[OXYGEN\_INSTALL\_DIR]/frameworks/docbook/5.0/rng/ (or for DocBook 5.1 in
\[OXYGEN\_INSTALL\_DIR]/frameworks/docbook/5.1/rng/). Other types of schemas for various
DocBook versions are also located in various folders inside the \[OXYGEN\_INSTALL\_DIR]/frameworks/ docbook/ directory.

**Default CSS**

The default CSS files used for rendering DocBook content in Author mode is stored in

\[OXYGEN\_INSTALL\_DIR]/frameworks/docbook/css/.

By default, these default CSS files are merged with any that are specified in the document. For more information, see Configuring and Managing Multiple CSS Styles for a Framework (on page 2145).

**Transformation Scenarios**

Oxygen XML Editor includes numerous built-in DocBook transformation scenarios that allow you to transform DocBook 5 documents to a variety of outputs, such as WebHelp, PDF, HTML, HTML Chunk, XHTML, XHTML Chunk, and EPUB. Oxygen XML Editor also includes a DocBook 5.1 transformation scenario for Assembly documents (on page 1334). All of them are listed in the DocBook 5 section in the Configure Transformation Scenario(s) dialog box (on page 1548).

For more information, see the DocBook Transformation Scenarios (on page 1458) section.

**Resources**

- Oxygen Video Tutorial: Editing DocBook Documents in Author Mode
- DocBook 5.0 (and older) Specifications
- DocBook 5.1 Specifications
- DocBook 5.1: The Definitive Guide

**Related Information:**

- Editing XML Documents in Author Mode (on page 588)
- Editing XML Documents in Text Mode (on page 517)
- Adding Tables in DocBook (on page 686)
DocBook 5 Author Mode Actions

A variety of actions are available for DocBook 5 documents in the DocBook5 menu, toolbar, contextual menu, and the Content Completion Assistant (on page 3217).

DocBook 5 Toolbar Actions

The following default actions are available on the DocBook toolbar when editing in Author mode (by default, most of them are also available in the DocBook5 menu and in various submenus of the contextual menu):

- **Bold**
  Emphasizes the selected text by surrounding it with a *bold* tag. You can use this action on multiple non-contiguous selections.

- **Italic**
  Emphasizes the selected text by surrounding it with an *italic* tag. You can use this action on multiple non-contiguous selections.

- **Underline**
  Emphasizes the selected text by surrounding it with an *underline* tag. You can use this action on multiple non-contiguous selections.

- **Link Actions Drop-Down Menu**

  The following link actions are available from this menu:

  - **Cross reference (link)**
    Opens a dialog box that allows you to select a target to insert as a hypertext link.

  - **Cross reference (xref)**
    Inserts a cross reference to other parts of the document.

  - **Web Link (ulink)**
    Inserts a link that addresses its target with a URL (Universal Resource Locator).

  - **Insert OLink**
    Opens an OLink dialog box that allows you to insert a link that addresses its target indirectly, using the values of the @targetdoc and @targetptr attributes that are present in a Targetset (on page 1336) file.
After you choose the Targetset URL, the structure of the target documents is presented. For each target document (@targetdoc), its content is displayed allowing you to easily identify the @targetptr for the <olink> element that will be inserted. You can also use the search fields to quickly identify a target. If you already know the values for targetdoc and targetptr, you can insert them directly in the corresponding fields. You can also edit an <olink> using the Edit OLink action that is available on the contextual menu. The last used Targetset URL will be used to identify the edited target.

To insert XREF text into the <olink>, enter the text in the xreftext field and make sure the Insert xreftext in the OLink option is selected.

**Insert URI**

Inserts a URI element. The URI identifies a Uniform Resource Identifier (URI) in content.

**Edit OLink**

Opens a dialog box that allows you edit an existing OLink. See the Insert OLink action for more information.

**Insert Image**

Opens a dialog box that allows you to select the path of an image to insert at the cursor position (on page 719). Depending on the current location, an image-type element is inserted. If the action is invoked between two block elements (such as paragraphs), the dialog box also allows you to provide a title.
Insert Media Resource

Opens a Choose Media dialog box (on page 747) that allows you to select the URL of a media object to be inserted into a document at the cursor position. The result will be that a reference to the specified video, audio, or embedded HTML frame is inserted and rendered in Author mode so that it can be played directly from there.

Insert XInclude

Opens a dialog box that allows you to browse and select content to be included and automatically generates the corresponding XInclude instruction.

Section Drop-Down Menu

The following actions are available from this menu:

Insert Section

Inserts a new section or subsection in the document, depending on the current context. For example, if the current context is `<sect1>`, then a `<sect2>` is inserted. By default, this action also inserts a `<para>` element as a child node. The `<para>` element can be deleted if it is not needed.

Promote Section (Ctrl + Alt + LeftArrow (Command + Alt + LeftArrow on macOS))

Promotes the current node as a sibling of the parent node.

Demote Section (Ctrl + Alt + RightArrow (Command + Alt + RightArrow on macOS))

Demotes the current node a child of the previous node.

Insert Paragraph

Insert a new paragraph element at current cursor position.

Insert Equation

Opens the XML Fragment Editor that allows you to insert and edit MathML notations (on page 749).

Insert List Item

Inserts a list item in the current list type.

Insert Ordered List

Inserts an ordered list at the cursor position. A child list item is also automatically inserted by default. You can also use this action to convert selected paragraphs or other types of lists to an ordered list.

Insert Itemized List

Inserts an itemized list at the cursor position. A child list item is also automatically inserted by default. You can also use this action to convert selected paragraphs or other types of lists to an itemized list.
Insert Variable List

Inserts a DocBook variable list. A child list item is also inserted automatically by default. You can also use this action to convert selected paragraphs or other types of lists to a variable list.

Insert Procedure List

Inserts a DocBook `<procedure>` element. A `<step>` child element is also inserted automatically. You can also use this action to convert selected paragraphs or other types of lists to a procedure list.

Sort

Sorts cells or list items in a table.

Insert Table

Opens a dialog box that allows you to configure and insert a table. You can generate a header and footer, set the number of rows and columns of the table and decide how the table is framed. You can also use this action to convert selected paragraphs, lists, and inline content (mixed content, text plus markup, that is rendered inside a block element (on page 3216)) into a table, with the selected content inserted in the first column, starting from the first row after the header (if a header is inserted).

Note:

If the selection contains a mixture of elements that cannot be converted, you will receive an error message saying that **Only lists, paragraphs, or inline content can be converted to tables.**

Insert Row

Inserts a new table row with empty cells below the current row. This action is available when the cursor is positioned inside a table.

Delete Row(s)

Deletes the table row located at the cursor position or multiple rows in a selection.

Insert Column

Inserts a new table column with empty cells after the current column. This action is available when the cursor is positioned inside a table.

Delete Column(s)

Deletes the table column located at the cursor position or multiple columns in a selection.

Table Properties

Opens the **Table properties** dialog box that allows you to configure properties of a table (such as frame borders).

Join Cells

Joins the content of the selected cells (both horizontally and vertically).
### Split Cell

Splits the cell at the cursor location. If Oxygen XML Editor detects more than one option to split the cell, a dialog box will be displayed that allows you to select the number of rows or columns to split the cell into.

### DocBook5 Contextual Menu Actions

The following actions are available in the contextual menu when editing in Author mode (most of them are also available in the DocBook5 menu at the top of the interface):

#### Add File to Review Task

This action can be used to add the current document to a task in the Content Fusion Tasks Manager view. Oxygen Content Fusion is a flexible, intuitive collaboration platform designed to adapt to any type of documentation review workflow. This functionality is available through a pre-installed connector add-on (on page 2527). To fully take advantage of all of the benefits and features of Content Fusion, your organization will need an Oxygen Content Fusion Enterprise Server. For more information, see the Oxygen Content Fusion website.

#### Edit Attributes

Displays an in-place attributes editor (on page 630) that allows you to manage the attributes of an element.

#### Edit Profiling Attributes

Allows you to change the profiling attributes (on page 668) defined on all selected elements.

#### Cut (Ctrl + X (Command + X on macOS))

Removes the currently selected content from the document and places it in the clipboard.

#### Copy (Ctrl + C (Command + C on macOS))

Places a copy of the currently selected content in the clipboard.

#### Paste (Ctrl + V (Command + V on macOS))

Inserts the current clipboard content into the document at the cursor position.

#### Paste special submenu

This submenu includes the following special paste actions:

- **Paste As XInclude**
  
  Allows you to create an `<xi:include>` element that references a DocBook element copied from Author mode. The operation fails if the copied element does not have a declared ID.

- **Paste as link**
Allows you to create a `<link>` element that references a DocBook element copied from Author mode. The operation fails if the copied element does not have a declared ID.

**Paste as xref**

Allows you to create an `<xref>` element that references a DocBook element copied from Author mode. The operation fails if the copied element does not have a declared ID.

**Image Map Editor**

This action is available in the contextual menu when it is invoked on an image. This action applies an image map to the current image (if one does not already exist) and opens the Image Map Editor dialog box. This feature allows you to create hyperlinks in specific areas of an image that will link to various destinations.

**Insert submenu**

This submenu includes the following insert actions that are specific to the DocBook framework:

---

**Insert Table**

Opens a dialog box that allows you to configure and insert a table. You can generate a header and footer, set the number of rows and columns of the table and decide how the table is framed. You can also use this action to convert selected paragraphs, lists, and inline content (mixed content, text plus markup, that is rendered inside a block element (on page 3216)) into a table, with the selected content inserted in the first column, starting from the first row after the header (if a header is inserted).

**Note:**

If the selection contains a mixture of elements that cannot be converted, you will receive an error message saying that **Only lists, paragraphs, or inline content can be converted to tables**.

---

**Insert Image**

Inserts an image reference (on page 719) at the cursor position. Depending on the current location, an image-type element is inserted.

---

**Insert Media Resource**

Opens a Choose Media dialog box (on page 747) that allows you to select the URL of a media object to be inserted into a document at the cursor position. The result will be that a reference to the specified video, audio, or embedded HTML frame is inserted and rendered in Author mode so that it can be played directly from there.
Opens the XML Fragment Editor that allows you to insert and edit MathML notations (on page 749).

Insert Paragraph
- Inserts a new paragraph element at current cursor position.

Insert Section
- Inserts a new section element in the document, depending on the current context.

Insert XInclude
- Opens a dialog box that allows you to browse and select content to be included and automatically generates the corresponding XInclude instruction.

Insert Entity
- Allows you to insert a predefined entity or character entity. Surrogate character entities (range #x10000 to #x10FFFF) are also accepted. Character entities can be entered in one of the following forms:
  
  - `#<decimal value>` - e.g. #65
  - `&#<decimal value>` - e.g. &#65
  - `#x<hexadecimal value>` - e.g. #x41
  - `&#x<hexadecimal value>` - e.g. &#x41

Style submenu
- This submenu includes the following text styling actions:

  - **B** Bold
    - Emphasizes the selected text by surrounding it with a bold tag. You can use this action on multiple non-contiguous selections.

  - **I** Italic
    - Emphasizes the selected text by surrounding it with an italic tag. You can use this action on multiple non-contiguous selections.

  - **U** Underline
    - Emphasizes the selected text by surrounding it with an underline tag. You can use this action on multiple non-contiguous selections.

  - **Tₜ** Subscript
    - Surrounds the selected text with a subscript tag, used for inserting a character (number, letter, or symbol) that will appear slightly below the baseline and slightly smaller than the rest of the text.

  - **Tʰ** Superscript
Surrounds the selected text with a *superscript* tag, used for inserting a character (number, letter, or symbol) that will appear slightly above the baseline and slightly smaller than the rest of the text.

§ **Section submenu**

The following actions are available in this submenu:

- **Promote Section** ([Ctrl + Alt + LeftArrow](Command + Alt + LeftArrow on macOS))
  Promotes the current node as a sibling of the parent node.

- **Demote Section** ([Ctrl + Alt + RightArrow](Command + Alt + RightArrow on macOS))
  Demotes the current node a child of the previous node.

¶ **Link submenu**

The following actions are available in this submenu:

- **Cross reference (link)**
  Opens a dialog box that allows you to select a target to insert as a hypertext link.

- **Cross reference (xref)**
  Inserts a cross reference to other parts of the document.

- **Web Link (ulink)**
  Inserts a link that addresses its target with a URL (Universal Resource Locator).

**Insert OLink**

Opens an OLink dialog box that allows you to insert a link that addresses its target indirectly, using the values of the [@targetdoc](#targetdoc) and [@targetptr](#targetptr) attributes that are present in a [Targetset](on page 1336) file.
After you choose the **Targetset URL**, the structure of the target documents is presented. For each target document (@targetdoc), its content is displayed allowing you to easily identify the @targetptr for the `<olink>` element that will be inserted. You can also use the search fields to quickly identify a target. If you already know the values for `targetdoc` and `targetptr`, you can insert them directly in the corresponding fields. You can also edit an `<olink>` using the **Edit OLink** action that is available on the contextual menu. The last used Targetset URL will be used to identify the edited target.

To insert XREF text into the `<olink>`, enter the text in the `xreftext` field and make sure the **Insert xreftext in the OLink** option is selected.

**Insert URI**

Inserts a URI element. The URI identifies a Uniform Resource Identifier (URI) in content.

**Edit OLink**

Opens a dialog box that allows you edit an existing `OLink`. See the **Insert OLink** action for more information.

**Table actions**

The following table editing actions are available in the contextual menu when it is invoked on a table:

**Insert Rows**
Opens a dialog box that allows you to insert any number of rows and specify the position where they will be inserted (Above or Below the current row).

**Delete Row(s)**

Deletes the table row located at the cursor position or multiple rows in a selection.

**Insert Columns**

Opens a dialog box that allows you to insert any number of columns and specify the position where they will be inserted (Above or Below the current column).

**Delete Column(s)**

Deletes the table column located at the cursor position or multiple columns in a selection.

**Join Cells**

Joins the content of the selected cells (both horizontally and vertically).

**Split Cell**

Splits the cell at the cursor location. If Oxygen XML Editor detects more than one option to split the cell, a dialog box will be displayed that allows you to select the number of rows or columns to split the cell into.

**Sort**

Sorts cells or list items in a table.

**Table Properties**

Opens the Table properties dialog box that allows you to configure properties of a table (such as frame borders).

**Other Actions submenu**

This submenu give you access to all the usual contextual menu actions.

**Generate IDs**

Oxygen XML Editor generates unique IDs for the current element (or elements), depending on how the action is invoked:

- When invoked on a single selection, an ID is generated for the selected element at the cursor position.
- When invoked on a block of selected content, IDs are generated for all top-level elements and elements listed in the ID Options dialog box that are found in the current selection.

**Note:**
The Generate IDs action does not overwrite existing ID values. It only affects elements that do not already have an @id attribute.
Select submenu

This submenu allows you to select the following:

Element
Selects the entire element at the current cursor position.

Content
Selects the entire content of the element at the current cursor position, excluding the start and end tag. Performing this action repeatedly will result in the selection of the content of the ancestor of the currently selected element content.

Parent
Selects the entire parent element at the current cursor position.

Text submenu

This submenu contains the following actions:

To Lower Case
Converts the selected content to lower case characters.

To Upper Case
Converts the selected content to upper case characters.

Capitalize Sentences
Converts to upper case the first character of every selected sentence.

Capitalize Words
Converts to upper case the first character of every selected word.

Count Words
Counts the number of words and characters (no spaces) in the entire document or in the selection for regular content and read-only content.

Note:
The content marked as deleted with change tracking (on page 3224) is ignored when counting words.

Convert Hexadecimal Sequence to Character (Ctrl + Shift + X (Command + Shift + X on macOS))

Converts a sequence of hexadecimal characters to the corresponding Unicode character (on page 463). The action can be invoked if there is a selection containing a valid hexadecimal sequence or if the cursor is placed at the right side of a valid hexadecimal sequence. A valid hexadecimal sequence can be composed
of 2 to 4 hexadecimal characters and may or may not be preceded by the \texttt{0x} or \texttt{0X} prefix. Examples of valid sequences and the characters they will be converted to:

- \texttt{0x0045} will be converted to \texttt{E}
- \texttt{0x0125} to \texttt{ĥ}
- \texttt{265} to \texttt{ɥ}
- \texttt{2190} to \texttt{←}

\begin{quote}
\textbf{Note:}
For more information about finding the hexadecimal value of a character, see \textit{Finding the Decimal, Hexadecimal, or Character Entity Equivalent (on page 466)}.
\end{quote}

Refactoring submenu

Contains a series of actions designed to alter the XML structure of the document:

- **Toggle Comment**
  
  Encloses the currently selected text in an XML comment, or removes the comment if it is commented.

- **Move Up (Alt + UpArrow)**
  
  Moves the current node or selected nodes in front of the previous node.

- **Move Down (Alt + DownArrow)**
  
  Moves the current node or selected nodes after the subsequent node.

- **Split Element (Alt + Shift + D (Ctrl + Alt + D on macOS))**
  
  Splits the content of the closest element that contains the position of the cursor. Thus, if the cursor is positioned at the beginning or at the end of the element, the newly created sibling will be empty.

- **Join Elements**
  
  Joins two adjacent \textit{block elements (on page 3216)} that have the same name. The action is available only when the cursor position is between the two adjacent \textit{block elements}. Also, joining two \textit{block elements} can be done by pressing the Delete or Backspace keys and the cursor is positioned between the boundaries of these two elements.

- **Surround with Tags (Ctrl + E (Command + E on macOS))**
  
  Allows you to choose a tag to enclose a selected portion of content. If there is no selection, the start and end tags are inserted at the cursor position.
• If the **Position cursor between tags** option (on page 212) is selected in the Content Completion preferences page, the cursor is placed between the start and end tag.

• If the **Position cursor between tags** option (on page 212) is not selected in the Content Completion preferences page, the cursor is placed at the end of the start tag, in an insert-attribute position.

![Surround with 'tag'] (Ctrl + ForwardSlash (Command + ForwardSlash on macOS))

Surround the selected content with the last tag used.

**Rename Element**

The element from the cursor position, and any elements with the same name, can be renamed according with the options from the **Rename** dialog box.

![Delete Element Tags]

Deletes the tags of the closest element that contains the position of the cursor. This operation is also executed if the start or end tags of an element are deleted by pressing the **Delete** or **Backspace** keys.

![Remove All Markup]

Removes all the XML markup inside the selected block of content and keeps only the text content.

![Remove Text]

Removes the text content of the selected block of content and keeps the markup intact with empty elements.

**Attributes Refactoring Actions**

Contains built-in XML refactoring operations that pertain to attributes with some of the information preconfigured based upon the current context.

**Add/Change attribute**

Allows you to change the value of an attribute or insert a new one.

**Convert attribute to element**

Allows you to change an attribute into an element.

**Delete attribute**

Allows you to remove one or more attributes.

**Rename attribute**

Allows you to rename an attribute.

**Replace in attribute value**
Allows you to search for a text fragment inside an attribute value and change the fragment to a new value.

**Comments Refactoring Actions**

Contains built-in XML refactoring operations that pertain to comments with some of the information preconfigured based upon the current context.

**Delete comments**

Allows you to delete comments found inside one or more elements.

**Elements Refactoring Actions**

Contains built-in XML refactoring operations that pertain to elements with some of the information preconfigured based upon the current context.

**Delete element**

Allows you to delete elements.

**Delete element content**

Allows you to delete the content of elements.

**Insert element**

Allows you to insert new elements.

**Rename element**

Allows you to rename elements.

**Unwrap element**

Allows you to remove the surrounding tags of elements, while keeping the content unchanged.

**Wrap element**

Allows you to surround elements with element tags.

**Wrap element content**

Allows you to surround the content of elements with element tags.

**Fragments Refactoring Actions**

Contains built-in XML refactoring operations that pertain to XML fragments with some of the information preconfigured based upon the current context.

**Insert XML fragment**

Allows you to insert an XML fragment.

**Replace element content with XML fragment**

Allows you to replace the content of elements with an XML fragment.

**Replace element with XML fragment**
Allows you to replace elements with an XML fragment.

**Review submenu**

This submenu includes the following actions:

- **Track Changes**
  Enables or disables the *Track Changes* support for the current document.

- **Accept Change(s) and Move to Next**
  Accepts the *Tracked Change* located at the cursor position or all of the changes in a selection and then moves to the next change. If you select a part of a *deletion or insertion* change, only the selected content is accepted.

- **Accept All Changes**
  Accepts all *Tracked Changes* in the current document.

- **Reject Change(s) and Move to Next**
  Rejects the *Tracked Change* located at the cursor position or all of the changes in a selection and then moves to the next change. If you select a part of a *deletion or insertion* change, only the selected content is rejected.

- **Reject All Changes**
  Rejects all *Tracked Changes* in the current document.

- **Comment Change**
  Opens a dialog box that allows you to add a comment to an existing *Tracked Change*. The comment will appear in a callout and a tooltip when hovering over the change. If the action is selected on an existing commented change, the dialog box will allow you to edit the comment.

- **Highlight**
  Enables the highlighting tool that allows you to mark text in your document.

  - **Colors**
    Allows you to select the color for highlighting text.

  - **Stop highlighting**
    Use this action to deactivate the highlighting tool.

  - **Remove highlight(s)**
    Use this action to remove highlighting from the document.

- **Add Comment**
  Inserts a comment at the cursor position. The comment appears in a callout box and a tooltip (when hovering over the change).
Show/Edit Comment

Opens a dialog box that displays the discussion thread and allows the current user to edit comments that do not have replies. If you are not the author who inserted the original comment, the dialog box just displays the comment without the possibility of editing it.

Remove Comment

Removes a selected comment. If you remove a comment that contains replies, all of the replies will also be removed.

Manage Reviews

Opens the Review view (on page 664).

Folding submenu

This submenu includes the following actions:

- **Toggle Fold**
  Toggles the state of the current fold.

- **Collapse Other Folds (Ctrl + NumPad/ (Command + NumPad/ on macOS))**
  Folds all the elements except the current element.

- **Collapse Child Folds (Ctrl + NumPad. (Command + NumPad. on macOS))**
  Folds the elements indented with one level inside the current element.

- **Expand Child Folds**
  Unfolds all child elements of the currently selected element.

- **Expand All (Ctrl + NumPad* (Command + NumPad* on macOS))**
  Unfolds all elements in the current document.

About Element > Go to Definition

Moves the cursor to the definition of the current element.

Inspect Styles

Opens the CSS Inspector view (on page 640) that allows you to examine the CSS rules that match the currently selected element.

Options

Opens the Author mode preferences page (on page 176) where you can configure various options with regard to the Author editing mode.
Floating Contextual Toolbar for DocBook

Oxygen XML Editor includes a dynamic feature where certain editing contexts will trigger a floating toolbar with common actions that are available in the current editing context.

![Figure 465. DocBook Floating Contextual Toolbar](image)

The floating contextual toolbar is automatically displayed when editing DocBook documents in the following situations:

- When an `<para>` or `<listitem>` element has a selection inside, the floating toolbar includes actions such as **Bold**, *Italic*, _Underline_, ↑_Subscript_, and ↑^Superscript_.
- When an `<imagedata>` or `<videodata>` element is selected, the floating toolbar includes a URL chooser where you can select the appropriate target.
- When an `<olink>` element is selected, the floating toolbar includes an Edit OLink action.
- When a `<link>` or `<include>` element is selected, the floating toolbar includes a URL chooser where you can select the appropriate target.
- When a `<programlisting>` element is selected, the floating toolbar includes a drop-down control where you can select the value of the @language attribute.
- When an `<itemizedlist>`, `<orderdlist>`, `<variablelist>`, or `<procedure>` element is selected, the floating toolbar includes actions for converting it to a different type of list or sorting the list.
- When a `<listitem>`, `<varlistentry>`, or `<step>` element is selected, the floating toolbar includes actions for moving the item up or down in the list/procedure.
- When a `<row>` or `<tr>` element is selected in a table, the floating toolbar includes various table-related actions (such as actions for editing table properties, inserting rows, or deleting rows).
- When an `<entry>` or `<td>` element is selected in a table, the floating toolbar includes various table-related actions (such as actions for editing table properties, inserting/deleting rows, or inserting/deleting columns).
- When a `<table>` element is selected, the floating toolbar includes actions for editing table properties or sorting the table.

DocBook 5 Drag/Drop (or Copy/Paste) Actions

Dragging a file from the **Project view (on page 402)** and dropping it into a DocBook 5 document that is edited in **Author** mode, creates a link to the dragged file (the `<link>` DocBook element) at the drop location. Copy and paste actions work the same.

You can also drag images or media files from your system explorer or the **Project view (on page 402)** and drop them into a DocBook 5 document (or copy and paste). This will insert the appropriate element at the drop or paste location (for example, dropping/pasting an image will insert the `<imageobject>` DocBook element with an `<imagedata>` child element and a @fileref attribute).
Tip:
For information about customizing Author mode actions for a particular framework (on page 3219) (document type), see the Customizing the Author Mode Editing Experience for a Framework (on page 2145) section.

Related Information:
Customizing the Author Mode Editing Experience for a Framework (on page 2145)

Inserting an Olink in DocBook Documents

The `<olink>` element is used for linking to resources outside the current DocBook document. The `@targetdoc` attribute is used for the document ID that contains the target element and the `@targetptr` attribute for the ID of the target element (the value of an `@id` or `@xml:id` attribute). The combination of those two attributes provides a unique identifier to locate cross references.

For example, a *Mail Administrator Guide* with the document ID `MailAdminGuide` might contain a chapter about user accounts, like this:

```xml
<chapter id="user_accounts">
  <title>Administering User Accounts</title>
  <para>blah blah</para>
</chapter>
```

You can form a cross reference to that chapter by adding an `<olink>`, as in the following example:

```xml
<olink targetdoc="MailAdminGuide" targetptr="user_accounts">user accounts
</olink>
```

You may need to update your

```xml
<olink targetdoc="MailAdminGuide" targetptr="user_accounts">user accounts
</olink>
```

when you get a new machine.

To use an `<olink>` to create links between documents, follow these steps:

1. Decide which documents are to be included in the domain for cross referencing.
   A unique ID must be assigned to each document that will be referenced with an `<olink>`. It is usually added as an `@id` (or `@xml:id` for DocBook5) attribute to the root element of the document.

2. Decide on your output hierarchy.
   For creating links between documents, the relative locations of the output documents must be known. Before going further you must decide the names and locations of the output directories for all the documents from the domain. Each directory will be represented by an element `<dir name="directory_name">`, in the target database document.

3. Create the target database document.
   Each collection of documents has a main target database document that is used to resolve all `olinks` from that collection. The target database document is an XML file that is created once. It provides a means for pulling in the target data for each document. The database document is static and all the document data is pulled in dynamically.
Tip:
Oxygen XML Editor includes a built-in new document template called **DocBook Targetset Map** available in the **New document wizard (on page 368)** that will help you get started.

**Example:** The following is an example of a target database document. It structures a collection of documents in a `<sitemap>` element that provides the relative locations of the outputs (HTML in this example). Then it pulls in the individual target data using system entity references to target data files that will be created in the next step.

```xml
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE targetset [ 
<!ENTITY ugtargets SYSTEM "file:///doc/userguide/target.db">
<!ENTITY agtargets SYSTEM "file:///doc/adminguide/target.db">
<!ENTITY reftargets SYSTEM "file:///doc/man/target.db">
]>
<targetset>
  <targetsetinfo>
    Description of this target database document, which is for the examples in olink doc.
  </targetsetinfo>

  <!-- Site map for generating relative paths between documents -->
  <sitemap>
    <dir name="documentation">
      <dir name="guides">
        <dir name="mailuser">
          <document targetdoc="MailUserGuide" baseuri="userguide.html">
            &ugtargets;
          </document>
        </dir>
      </dir>
    </dir>
    <dir name="mailadmin">
      <document targetdoc="MailAdminGuide">
        &agttargets;
      </document>
    </dir>
  </sitemap>
</targetset>
```
4. Generate the target data files by executing a DocBook transformation scenario.

Before applying the transformation, you need to edit the transformation scenario, go to the Parameters tab, and make sure the value of the `collect.xref.targets` parameter is set to `yes`. The default name of a target data file is `target.db`, but it can be changed by setting an absolute file path in the `targets.filename` parameter.

**Example:** An example of a `target.db` file:

```
<book href="#MailAdminGuide" number="1" targetptr="user_accounts">
  <title>Administering User Accounts</title>
  <xref>How to administer user accounts</xref>
  <part href="#d5e4" number="I">
    <title>First Part</title>
    <xref>Part I, "First Part"</xref>
  </part>
  <chapter href="#d5e6" number="1">
    <title>Chapter Title</title>
    <xref>Chapter 1, Chapter Title</xref>
  </chapter>
  <sect1 href="#src_chapter" number="1" targetptr="src_chapter">
    <xref>Section1 Title</xref>
  </sect1>
</book>
```


When editing a DocBook XML document in Author mode, the Insert OLink action is available in the Link drop-down menu from the toolbar. This action opens the Insert OLink dialog box that allows you to select the target of an `<olink>` from the list of all possible targets from a specified target database document (specified in the Targetset URL field). Once a Targetset URL is selected, the structure of the target documents is presented. For each target document (`@targetdoc`), its content is displayed, allowing you to easily identify the appropriate `@targetptr`. You can also use the search fields to quickly identify a target. If you already know the values for the `@targetdoc` and `@targetptr` attributes, you can insert them directly in the corresponding fields.

**Example:** In the following image, the target database document is called `target.xml`, `dbadmin` is selected for the target document (`@targetdoc`), and `bldinit` is selected as the value for the `@targetptr` attribute. Notice that you can also add XREF text into the `<olink>` by using the `xreftext` field.
Process a DocBook transformation for each document to generate the output.

a. Edit the transformation scenario and set the value of the `target.database.document` parameter to be the URL of the target database document.

b. Apply the transformation scenario.

**DocBook 5.1 Assembly**

The DocBook Assembly document type was introduced with DocBook 5.1 and it is used to define the hierarchy and relationships for a collection of resources. It is especially helpful for topic-oriented authoring scenarios since it assembles a set of resources (such as DocBook 5.1 topics (on page 1335)) to form a hierarchical structure for a larger publication.

An Assembly document usually has four major parts:

- **Resources** - Identifies a collection of resources (such as topics). An Assembly may identify one or more collections.

- **Structure** - Identifies an artifact to be assembled. A document in this case is the particular collection of resources (such as topics) that forms the documentation. Within the `<structure>` element, an `<output>` element can be used to identify the type of output to be generated and `<module>` elements can be used to identify the resources to be included. An Assembly may identify one or more structures.
• **Relationships** - Identifies relationships between resources. These relationships may be manifested in any number of *structures* during assembly. An *Assembly* may identify any number of relationships.

• **Transformations** - Identifies transformations that can be applied during assembly. An *Assembly* may identify any number of transformations.

For detailed information about the DocBook *Assembly* document type, see *DocBook 5.1: The Definitive Guide - DocBook Assemblies*.

**File Definition**

A file is considered to be an *Assembly* when the root name is `assembly`.

**Default Document Templates**

A default *Assembly* document template is available when creating new documents from templates (on page 368) and it can be found in: Framework Templates > DocBook 5 > DocBook 5.1.

The default template for DocBook Assembly documents is located in the 
`{OXYGEN_INSTALL_DIR}/frameworks/docbook/templates/Docbook5.1` folder.

**Default Schema for Validation and Content Completion**

The default schema that is used if one is not detected is `docbookxi.rng` and it is stored in

`{OXYGEN_INSTALL_DIR}/frameworks/docbook/5.1/rng/`.

**Transformation Scenarios**

Oxygen XML Editor includes a built-in transformation scenario that can be applied on an *Assembly* file to generate an *assembled* (merged) DocBook file. The scenario is called *DocBook Assembly* and is found in the DocBook 5 section in the Configure Transformation Scenario(s) dialog box (on page 1548).

**Resources**

- *DocBook 5.1: The Definitive Guide - DocBook Assemblies*
- *DocBook 5.1 Specifications*
- Sample files: `{OXYGEN_INSTALL_DIR}/samples/docbook/v5/assembly`/

**DocBook 5.1 Topic**

The DocBook *Topic* document type was introduced with DocBook 5.1 and it is used as a modular unit of documentation. It is similar to the concept of the DITA *Topic* and can be used as modular resources in conjunction with DocBook 5.1 *Assembly* (on page 1334) documents.

For detailed information about the DocBook *Topic* document type, see *DocBook 5.1: The Definitive Guide - Topic*.
File Definition

A DocBook file is considered to be a Topic when the root name is `topic`.

Default Document Templates

A default Topic document template is available when creating new documents from templates (on page 368) and it can be found in: Framework Templates > DocBook 5 > DocBook 5.1.

The default template for DocBook Assembly documents is located in the `\{OXYGEN_INSTALL_DIR\}/frameworks/docbook/templates/Docbook5.1` folder.

Default Schema for Validation and Content Completion

The default schema that is used if one is not detected is `docbookxi.rng` and it is stored in `\{OXYGEN_INSTALL_DIR\}/frameworks/docbook/5.1/rng/`.

Transformation Scenarios

Since DocBook Topics are modular resources, they are assembled and transformed in the DocBook Assembly transformation process (on page 1335). You can also use any of the built-in DocBook transformation scenarios to transform individual DocBook Topics to a variety of outputs, such as PDF, HTML, EPUB, and more. They are found in the DocBook 5 section in the Configure Transformation Scenario(s) dialog box (on page 1548).

Resources

- DocBook 5.1: The Definitive Guide - Topic
- DocBook 5.1 Specifications
- Sample files: `\{OXYGEN_INSTALL_DIR\}/samples/docbook/v5/assembly/`

Related Information:

DocBook 5.1 Assembly (on page 1334)

DocBook Targetset Document Type (Framework)

DocBook Targetset documents are used to resolve cross references with the DocBook Olink.

File Definition

A file is considered to be a Targetset when the root name is `targetset`.

Default Document Templates

A default DocBook Targetset Map document template is available when creating new documents from templates (on page 368) and it can be found in: Framework Templates > DocBook Targetset.
The default template for DocBook Targetset documents is located in the

\[OXYGEN_INSTALL_DIR]/frameworks/docbook/templates/Targetset folder.

Default Schema for Validation and Content Completion

The default schema, `targetdatabase.dtd`, for this type of document is stored in

\[OXYGEN_INSTALL_DIR]/frameworks/docbook/xsl/common/.

Related Information:
DocBook Specifications

DITA Topics Document Type (Framework)

The Darwin Information Typing Architecture (DITA) is an XML-based architecture for authoring, producing, and delivering technical information. It divides content into small, self-contained topics that you can reuse in various deliverables. The extensibility of DITA permits organizations to define specific information structures while still using standard tools to work with them. DITA content is created as topics, each an individual XML file. Typically, each topic has a defined primary objective and structure, and DITA also includes several specialized topic types (task, concept, reference, glossary entry).

For much more detailed information, resources, and instructions, see the DITA Authoring (on page 2886) chapter.

File Definition

A file is considered to be a DITA topic document when one of the following conditions are true:

- The root element name is one of the following: `<concept>`, `<task>`, `<reference>`, `<dita>`, or `<topic>`.
- The PUBLIC ID of the document is a PUBLIC ID for the elements listed above.
- The root element of the file has a `@DITAArchVersion` attribute for the “http://dita.oasis-open.org/architecture/2005/” namespace. This enhanced case of matching is only applied when the Enable DTD/XML Schema processing in document type detection option (on page 139) is selected from the Document Type Association preferences page (on page 138).

Default Document Templates

There are a variety of default DITA topic templates available when creating new documents from templates (on page 368) and they can be found in various folders inside: Framework Templates > DITA.

The default templates for DITA topic documents are located in the \[OXYGEN_INSTALL_DIR]/frameworks/dita/templates/topic folder.

Default Schema for Validation and Content Completion

Default schemas that are used if one is not detected in the DITA documents are stored in the various folders inside DITA-OT-DIR/dtd/ or DITA-OT-DIR/schema/.
Default CSS

The default CSS files used for rendering DITA content in Author mode are stored in the various folders inside: 

\[OXYGEN\_INSTALL\_DIR]/frameworks/dita/css/.

By default, these default CSS files are merged with any that are specified in the document. For more information, see Configuring and Managing Multiple CSS Styles for a Framework (on page 2145).

Default XML Catalogs

The default XML Catalogs (on page 3225) for the DITA topic document type are as follows:

- \[DITA\_OT\_DIR]/catalog-dita.xml
- \[OXYGEN\_INSTALL\_DIR]/frameworks/dita/catalog.xml
- \[OXYGEN\_INSTALL\_DIR]/frameworks/dita/plugin/catalog.xml
- \[OXYGEN\_INSTALL\_DIR]/frameworks/dita/styleguide/catalog.xml

Transformation Scenarios

Oxygen XML Editor includes built-in transformation scenarios for transforming individual DITA Topics to HTML5, XHTML, or PDF output. They can be found in the DITA section in the Configure Transformation Scenario(s) dialog box (on page 1548).

Resources

- DITA Specifications
- DITA Style Guide Best Practices for Authors
- Oxygen Video Tutorial: DITA Editing

Related Information:
DITA Authoring (on page 2886)
Getting Started with DITA (on page 2887)
Editing XML Documents in Author Mode (on page 588)
Editing XML Documents in Text Mode (on page 517)

DITA Topic Author Mode Actions

A variety of actions are available for DITA documents that can be found in DITA menu, toolbar, contextual menu, and the Content Completion Assistant (on page 3217).

DITA Toolbar Actions

The following default actions are available on the DITA toolbar when editing in Author mode (by default, most of them are also available in the DITA menu and in various submenus of the contextual menu):
**Bold**

Surrounds the selected text with a `<b>` tag. You can use this action on multiple non-contiguous selections.

**Italic**

Surrounds the selected text with an `<i>` tag. You can use this action on multiple non-contiguous selections.

**Underline**

Surrounds the selected text with a `<u>` tag. You can use this action on multiple non-contiguous selections.

🔗 **Link Actions Drop-Down Menu**

The following link actions are available from this menu:

**Cross Reference**

Opens the [Cross Reference (xref) dialog box](on page 3073) that allows you to insert a link to a target DITA resource at the current location within a document. The target resource can be the location of a file or a key that is already defined in your [DITA map](on page 3218) structure. Once the target resource has been selected, you can also target specific elements within that resource. For more information, see [Linking in DITA Topics](on page 3072).

**File Reference**

Opens the [File Reference dialog box](on page 3073) that allows you to insert a link to a target non-DITA file resource at the current location within a document. The target resource can be the location of a file or a key that is already defined in your [DITA map](on page 3218) structure. For more information, see [Linking in DITA Topics](on page 3072).

**Web Link**

Opens the [Web Link dialog box](on page 3073) that allows you to insert a link to a target web-related resource at the current location within a document. The target resource can be a URL or a key that is already defined in your [DITA map](on page 3218) structure. For more information, see [Linking in DITA Topics](on page 3072).

**Related Link to Topic**

Opens the [Cross Reference (xref) dialog box](on page 3074) that allows you to insert a link to a target DITA resource in a related links section at the bottom of the current document. The target resource can be the location of a file or a key that is already defined in your [DITA map](on page 3218) structure. Once the target resource has been selected, you can also target specific elements within that resource. If a related links section does not already exist, this action creates one. For more information, see [Linking in DITA Topics](on page 3072).
Tip:
You can use the Find Similar Topics action (available in the contextual menu or DITA menu) to quickly find related topics that can be added as related links. It opens the Open/Find Resource view and performs a search using text content from the <title>, <shortdesc>, <keyword>, and <indexterm> elements.

Related Link to File
Opens the File Reference dialog box (on page 3074) that allows you to insert a link to a target non-DITA file resource in a related links section at the bottom of the current document. The target resource can be the location of a file or a key that is already defined in your DITA map structure. If a related links section does not already exist, this action creates one. For more information, see Linking in DITA Topics (on page 3072).

Related Link to Web Page
Opens the Web Link dialog box (on page 3074) that allows you to insert a link to a target web-related resource in a related links section at the bottom of the current document. The target resource can be a URL or a key that is already defined in your DITA map structure. If a related links section does not already exist, this action creates one. For more information, see Linking in DITA Topics (on page 3072).

Insert Image
Opens the Insert Image dialog box (on page 2973) that allows you to configure the properties of an image to be inserted into a DITA document at the cursor position.

Insert Media Resource
Opens the Insert Media dialog box (on page 2977) that allows you to select and configure the properties of a media object to be inserted into a DITA document at the cursor position. The result will be that a reference to the specified video, audio, or embedded HTML frame is inserted in an <object> element and it is rendered in Author mode so that it can be played directly from there.

Insert Section Drop-Down Menu
The following insert actions are available from this menu:

- **Insert Section**
  Inserts a new <section> element in the document, depending on the current context.

- **Insert Concept**
  Inserts a new <concept> element, depending on the current context. Concepts provide background information that users must know before they can successfully work with a product or interface.
**Insert Task**

Inserts a new `<task>` element, depending on the current context. Tasks are the main building blocks for task-oriented user assistance. They generally provide step-by-step instructions that will help a user to perform a task.

**Insert Topic**

Inserts a new `<topic>` element, depending on the current context. Topics are the basic units of DITA content and are usually organized around a single subject.

**Insert Reference**

Inserts a new `<reference>` element, depending on the current context. A reference is a top-level container for a reference topic.

**Insert Paragraph**

Inserts a new paragraph at current cursor position.

**Reuse Content**

This action provides a mechanism for reusing content fragments. It opens the Reuse Content dialog box (on page 3043) that allows you to insert several types of references to reusable content at the cursor position. The types of references that you can insert using this dialog box include content references (@conref) (on page 3044), content key references (@conkeyref) (on page 3046), or key references to metadata (@keyref) (on page 3049).

**Insert step or list item**

Inserts a new list or step item in the current list type.

**Insert Unordered List**

Inserts an unordered list at the cursor position. A child list item is also automatically inserted by default. You can also use this action to convert selected paragraphs or other types of lists to an unordered list.

**Insert Ordered List**

Inserts an ordered list at the cursor position. A child list item is also automatically inserted by default. You can also use this action to convert selected paragraphs or other types of lists to an ordered list.

**Sort**

Sorts cells or list items in a table.

**Insert Table**

Opens a dialog box that allows you to configure and insert a table. You can generate a header and footer, set the number of rows and columns of the table and decide how the table is framed. You can also use this action to convert selected paragraphs, lists, and inline content (mixed content, text plus markup, that is rendered inside a block element (on page 3216)) into a table,
with the selected content inserted in the first column, starting from the first row after the header (if a header is inserted).

**Note:**
If the selection contains a mixture of elements that cannot be converted, you will receive an error message saying that **Only lists, paragraphs, or inline content can be converted to tables**.

- **Insert Row**
  Inserts a new table row with empty cells below the current row. This action is available when the cursor is positioned inside a table.

- **Delete Row(s)**
  Deletes the table row located at the cursor position or multiple rows in a selection.

- **Insert Column**
  Inserts a new table column with empty cells after the current column. This action is available when the cursor is positioned inside a table.

- **Delete Column(s)**
  Deletes the table column located at the cursor position or multiple columns in a selection.

- **Table Properties**
  Opens the **Table properties** dialog box that allows you to configure properties of a table (such as frame borders).

- **Join Cells**
  Joins the content of the selected cells (both horizontally and vertically).

- **Split Cell**
  Splits the cell at the cursor location. If Oxygen XML Editor detects more than one option to split the cell, a dialog box will be displayed that allows you to select the number of rows or columns to split the cell into.

**DITA Contextual Menu Actions**

The following actions are available in the contextual menu when editing in **Author** mode (most of them are also available in the **DITA** menu at the top of the interface):

- **Add File to Review Task**
  This action can be used to add the current document to a task in the **Content Fusion Tasks Manager** view. **Oxygen Content Fusion** is a flexible, intuitive collaboration platform designed to adapt to any type of documentation review workflow. This functionality is available through a pre-installed **connector add-on (on page 2527)**. To fully take advantage of all of the benefits and
features of Content Fusion, your organization will need an Oxygen Content Fusion Enterprise Server. For more information, see the Oxygen Content Fusion website.

Edit Attributes

Displays an in-place attributes editor (on page 630) that allows you to manage the attributes of an element.

Edit Profiling Attributes

Allows you to change the profiling attributes (on page 668) defined on all selected elements.

Cut (Ctrl + X (Command + X on macOS))

Removes the currently selected content from the document and places it in the clipboard.

Copy (Ctrl + C (Command + C on macOS))

Places a copy of the currently selected content in the clipboard.

Paste (Ctrl + V (Command + V on macOS))

Inserts the current clipboard content into the document at the cursor position.

Paste special submenu

This submenu includes the following special paste actions that are specific to the DITA framework:

Paste as content reference

Inserts a content reference (a DITA element with a @conref attribute) to the DITA XML element from the clipboard. An entire DITA XML element with an ID attribute must be present in the clipboard when the action is invoked. The conref attribute will point to this ID value.

Paste as content key reference

Allows you to indirectly reference content using the @conkeyref attribute. When the DITA content is processed, the key references are resolved using key definitions from DITA maps (on page 3218). To use this action, you must first do the following:

1. Make sure the DITA element that contains the copied content has an ID attribute assigned to it.
2. In the DITA Maps Manager view, make sure that the Context combo box (on page 2901) points to the correct map that stores the keys.
3. Make sure the topic that contains the content you want to reference has a key assigned to it. To assign a key, right-click the topic with its parent map opened in the DITA Maps Manager, select Edit Properties, and enter a value in the Keys field.

Paste as link
Looks for the first element with an ID value in the clipboard and inserts an `<xref>` that points to that element. If no elements with an ID value are found, a message will appear that informs you that to use this action, the clipboard contents must include at least one element with a declared ID.

**Paste as link (keyref)**

Inserts a link to the element that you want to reference. To use this action, you must first do the following:

1. Make sure the DITA element that contains the copied content has an ID attribute assigned to it.
2. In the **DITA Maps Manager** view, make sure that the **Context combo box (on page 2901)** points to the correct map that stores the keys.
3. Make sure the topic that contains the content you want to reference has a key assigned to it. To assign a key, right-click the topic with its parent map opened in the **DITA Maps Manager**, select **Edit Properties**, and enter a value in the **Keys** field.

**Insert submenu**

This submenu includes the following insert actions that are specific to the DITA framework:

- **Insert Table**
  
  Opens a dialog box that allows you to configure and insert a table. You can generate a header and footer, set the number of rows and columns of the table and decide how the table is framed. You can also use this action to convert selected paragraphs, lists, and inline content (mixed content, text plus markup, that is rendered inside a **block element (on page 3216)**) into a table, with the selected content inserted in the first column, starting from the first row after the header (if a header is inserted).

  **Note:**
  
  If the selection contains a mixture of elements that cannot be converted, you will receive an error message saying that **Only lists, paragraphs, or inline content can be converted to tables**.

- **Insert Image**
  
  Inserts an image reference (on page 719) at the cursor position. Depending on the current location, an image-type element is inserted.

- **Insert Media Resource**
  
  Opens a **Choose Media dialog box (on page 747)** that allows you to select the URL of a media object to be inserted into a document at the cursor position. The result
will be that a reference to the specified video, audio, or embedded HTML frame is inserted and rendered in Author mode so that it can be played directly from there.

∑ Insert Equation

Opens the XML Fragment Editor that allows you to insert and edit MathML notations (on page 749).

📖 Insert Note

Inserts a new <note> element at the current cursor position.

{} Insert Code Block

Inserts a new <codeblock> element at current cursor position.

📦 Insert Menu Cascade

Inserts a new <menucascade> element at current cursor position.

¶ Insert Paragraph

Inserts a new <p> (paragraph) element at current cursor position.

§ Insert Section

Inserts a new <section> element in the document, depending on the current context.

🔥 Insert Topic

Inserts a new <topic> element, depending on the current context. Topics are the basic units of DITA content and are usually organized around a single subject.

Insert Entity

Allows you to insert a predefined entity or character entity. Surrogate character entities (range #x10000 to #x10FFFF) are also accepted. Character entities can be entered in one of the following forms:

- #<decimal value> - e.g. #65
- &#<decimal value> - e.g. &#65
- #x<hexadecimal value> - e.g. #x41
- &#x<hexadecimal value> - e.g. &#x41

Style submenu

This submenu includes the following text styling actions:

**Bold**

Emphasizes the selected text by surrounding it with a <b> (bold) tag. You can use this action on multiple non-contiguous selections.

*Italic*
Emphasizes the selected text by surrounding it with an `<i>` (italic) tag. You can use this action on multiple non-contiguous selections.

**Underline**

Emphasizes the selected text by surrounding it with a `<u>` (underline) tag. You can use this action on multiple non-contiguous selections.

**Subscript**

Surrounds the selected text with a `<sub>` (subscript) tag, used for inserting a character (number, letter, or symbol) that will appear slightly below the baseline and slightly smaller than the rest of the text.

**Superscript**

Surrounds the selected text with a `<sup>` (superscript) tag, used for inserting a character (number, letter, or symbol) that will appear slightly above the baseline and slightly smaller than the rest of the text.

**Code**

Surrounds the selected text with a `<codeph>` tag.

**Ui Control**

Surrounds the selected text with a `<uicontrol>` tag, used to mark up names of buttons, entry fields, menu items, or other interface objects.

**Filepath**

Surrounds the selected text with a `<filepath>` tag, used to indicate the name, and optionally the location of a referenced file. You can specify the directory that contains the file and other directories that may precede it in the system hierarchy.

**Image Map Editor**

This action is available in the contextual menu when it is invoked on an image. This action applies an image map to the current image (if one does not already exist) and opens the Image Map Editor dialog box. This feature allows you to create hyperlinks in specific areas of an image that will link to various destinations.

**Table Actions**

A variety of table editing actions are available in the contextual menu when it is invoked on a table (depending on the context, the table-related actions are promoted to the top level of the contextual menu and the Other Actions submenu provides access to the other actions):

**Insert Rows**

Opens a dialog box that allows you to insert any number of rows and specify the position where they will be inserted (Above or Below the current row).

**Delete Row(s)**
Deletes the table row located at the cursor position or multiple rows in a selection.

**Insert Columns**

Opens a dialog box that allows you to insert any number of columns and specify the position where they will be inserted (Above or Below the current column).

**Delete Column(s)**

Deletes the table column located at the cursor position or multiple columns in a selection.

**Join Cells**

Joins the content of the selected cells (both horizontally and vertically).

**Split Cell**

Splits the cell at the cursor location. If Oxygen XML Editor detects more than one option to split the cell, a dialog box will be displayed that allows you to select the number of rows or columns to split the cell into.

**Sort**

Sorts cells or list items in a table.

**Table Properties**

Opens the Table properties dialog box that allows you to configure properties of a table (such as frame borders).

**Other Actions submenu**

This submenu give you access to all the usual contextual menu actions.

**Link submenu**

The following link actions are available from this submenu:

**Cross Reference**

Opens the Cross Reference (xref) dialog box (on page 3073) that allows you to insert a link to a target DITA resource at the current location within a document. The target resource can be the location of a file or a key that is already defined in your DITA map (on page 3218) structure. Once the target resource has been selected, you can also target specific elements within that resource. For more information, see Linking in DITA Topics (on page 3072).

**File Reference**

Opens the File Reference dialog box (on page 3073) that allows you to insert a link to a target non-DITA file resource at the current location within a document. The target resource can be the location of a file or a key that is already defined in your DITA map structure. For more information, see Linking in DITA Topics (on page 3072).
Web Link

Opens the **Web Link** dialog box (on page 3073) that allows you to insert a link to a target web-related resource at the current location within a document. The target resource can be a URL or a key that is already defined in your **DITA map** structure. For more information, see Linking in DITA Topics (on page 3072).

Related Link to Topic

Opens the **Cross Reference (xref)** dialog box (on page 3074) that allows you to insert a link to a target DITA resource in a related links section at the bottom of the current document. The target resource can be the location of a file or a key that is already defined in your **DITA map** structure. Once the target resource has been selected, you can also target specific elements within that resource. If a related links section does not already exist, this action creates one. For more information, see Linking in DITA Topics (on page 3072).

Tip:

You can use the **Find Similar Topics** action (available in the contextual menu or **DITA** menu) to quickly find related topics that can be added as related links. It opens the **Open/Find Resource** view and performs a search using text content from the `<title>`, `<shortdesc>`, `<keyword>`, and `<indexterm>` elements.

Related Link to File

Opens the **File Reference** dialog box (on page 3074) that allows you to insert a link to a target non-DITA file resource in a related links section at the bottom of the current document. The target resource can be the location of a file or a key that is already defined in your **DITA map** structure. If a related links section does not already exist, this action creates one. For more information, see Linking in DITA Topics (on page 3072).

Related Link to Web Page

Opens the **Web Link** dialog box (on page 3074) that allows you to insert a link to a target web-related resource in a related links section at the bottom of the current document. The target resource can be a URL or a key that is already defined in your **DITA map** structure. If a related links section does not already exist, this action creates one. For more information, see Linking in DITA Topics (on page 3072).

Sort

Available when invoked on a list, it opens a dialog box where you can configure a sorting operation for an entire list or a selection of list items.

Generate IDs
Oxygen XML Editor generates unique IDs for the current element (or elements), depending on how the action is invoked:

- When invoked on a single selection, an ID is generated for the selected element at the cursor position.
- When invoked on a block of selected content, IDs are generated for all top-level elements and elements listed in the ID Options dialog box that are found in the current selection.

**Note:**
The Generate IDs action does not overwrite existing ID values. It only affects elements that do not already have an @id attribute.

**Reuse submenu**

This submenu includes the following actions regarding reusing content in DITA:

- **Reuse Content**
  This action provides a mechanism for reusing content fragments. It opens the Reuse Content dialog box (on page 3043) that allows you to insert several types of references to reusable content at the cursor position. The types of references that you can insert using this dialog box include content references (@conref) (on page 3044), content key references (@conkeyref) (on page 3046), or key references to metadata (@keyref) (on page 3049).

- **Push Current Element**
  Opens the Push current element dialog box (on page 3052) that allows content from a source topic to be inserted into another topic without any special coding in the topic where the content will be re-used.

- **Edit Content Reference**
  This action is available for elements with a @conref or @conkeyref attribute. It opens the Edit Content Reference dialog box that allows you to edit the source location (or key) and source element of a content reference (or content key reference), and the reference details (@conref/@conkeyref and @conrefend attributes). For more information, see Reuse Content Dialog Box (on page 3043).

- **Replace Reference with Content**
  Replaces the referenced fragment (@conref or @conkeyref) at the cursor position with its content from its source. This action is useful if you want to make changes to the content in the currently edited document without changing the referenced fragment in its source location. If the source content includes references to other topics/resources (hrefs), the operation also resolves those references relative to the new location. Attributes are preserved according to the following priority:
1. Attributes from the elements in the current document that reference other content are preserved except for attributes with a `-dita-use-conref-target` value.

2. Attributes from the referenced content are brought into the replaced elements in the current document except for `@id` attributes.

**Replace All References with Content**

Replaces all referenced fragments (`@keyref`, `@conref`, or `@conkeyref`) in the current document with the content. Attributes are preserved according to the following priority:

1. Attributes from the elements in the current document that reference other content are preserved except for attributes with a `-dita-use-conref-target` value.

2. Attributes from the referenced content are brought into the replaced elements in the current document except for `@id` attributes.

For `keyrefs` inside `<xref>` or `<link>` elements, the `@keyref` attribute is changed to an `@href` attribute, while the rest of the content for the `keyref` is replaced with its source content.

If the source content includes references to other topics/resources (`hrefs`), the operation also resolves those references relative to the new location.

**Remove Content Reference**

Removes the content reference (`@conref` or `@conkeyref`) inside the element at the cursor position.

**Create Reusable Component**

Opens a dialog box that helps you to create a reusable component from the current element or selection of elements. If the **Replace selection with content reference** option is selected in the dialog box, the selection will be replaced with a content reference (`@conref`). If multiple elements are selected (for example, multiple steps or list items), the selection will be replaced with a content reference range (`@conref` and `@conrefend`). For more information, see Creating a Reusable Content Component (on page 3055).

**Insert Reusable Component**

Inserts a reusable component at cursor location. For more information, see Inserting a Reusable Content Component (on page 3056).

**Search References (Ctrl + Shift + G (Command + Shift + G on macOS))**

Finds the references to the `@id` attribute value for the element at the current cursor position, in all the topics contained in the current DITA map (on page 3218) (opened in the DITA Maps...
Manager view (on page 2897)). If no references are found for the current element, a dialog box will be displayed that offers you the option of searching for references to its ancestor elements.

Figure 467. Search References to Ancestors Dialog Box

![Search References to Ancestors Dialog Box](image)

Tip:
If you are invoking the action on an image, see Searching for References to Images (on page 2976) for details about what will be reported.

Find Similar Topics
Opens the Open/Find Resource view and performs a search using text content from the `<title>`, `<shortdesc>`, `<keywords>`, and `<indexterm>` elements. It is helpful for quickly finding related topics that can be added as related links.

Show Key Definition
Available for elements that have a `@conkeyref` or `@keyref` attribute set (or elements with an ancestor element that has a `@conkeyref` or `@keyref` attribute). It computes the key name and opens the DITA map (on page 3218) that contains the definition of the key with the element that defines that key selected.

About Element submenu
This submenu includes the following actions:

Style Guide
Opens the DITA Style Guide Best Practices for Authors in your browser and displays a topic that is relevant to the element at the cursor position. When editing DITA documents, this action is available in the contextual menu of the editing area (under the About Element sub-menu), in the DITA menu, and in some of the documentation tips that are displayed by the Content Completion Assistant (on page 3217).

Browse reference manual
Opens a reference to the documentation of the XML element closest to the cursor position in a web browser.

Go to Definition
Moves the cursor to the definition of the current element.
Select submenu

This submenu allows you to select the following:

**Element**

Selects the entire element at the current cursor position.

**Content**

Selects the entire content of the element at the current cursor position, excluding the start and end tag. Performing this action repeatedly will result in the selection of the content of the ancestor of the currently selected element content.

**Parent**

Selects the entire parent element at the current cursor position.

Text submenu

This submenu contains the following actions:

**To Lower Case**

Converts the selected content to lower case characters.

**To Upper Case**

Converts the selected content to upper case characters.

**Capitalize Sentences**

Converts to upper case the first character of every selected sentence.

**Capitalize Words**

Converts to upper case the first character of every selected word.

**Count Words**

Counts the number of words and characters (no spaces) in the entire document or in the selection for regular content and read-only content.

**Note:**

The content marked as deleted with change tracking (on page 3224) is ignored when counting words.

**Convert Hexadecimal Sequence to Character (Ctrl + Shift + X (Command + Shift + X on macOS))**

Converts a sequence of hexadecimal characters to the corresponding Unicode character (on page 463). The action can be invoked if there is a selection containing a valid hexadecimal sequence or if the cursor is placed at the right side of a valid hexadecimal sequence. A valid hexadecimal sequence can be composed
of 2 to 4 hexadecimal characters and may or may not be preceded by the `0x` or `0X` prefix. Examples of valid sequences and the characters they will be converted to:

- `0x0045` will be converted to `E`
- `0x0125` to `ĥ`
- `265` to `ɥ`
- `2190` to `←`

Note:
For more information about finding the hexadecimal value of a character, see Finding the Decimal, Hexadecimal, or Character Entity Equivalent (on page 466).

Refactoring submenu

Contains a series of actions designed to alter the XML structure of the document:

- **Toggle Comment**

  Encloses the currently selected text in an XML comment, or removes the comment if it is commented.

- **Move Up (Alt + UpArrow)**

  Moves the current node or selected nodes in front of the previous node.

- **Move Down (Alt + DownArrow)**

  Moves the current node or selected nodes after the subsequent node.

- **Split Element (Alt + Shift + D (Ctrl + Alt + D on macOS))**

  Splits the content of the closest element that contains the position of the cursor. Thus, if the cursor is positioned at the beginning or at the end of the element, the newly created sibling will be empty.

- **Join Elements**

  Joins two adjacent block elements (on page 3216) that have the same name. The action is available only when the cursor position is between the two adjacent block elements. Also, joining two block elements can be done by pressing the Delete or Backspace keys and the cursor is positioned between the boundaries of these two elements.

- **Surround with Tags (Ctrl + E (Command + E on macOS))**

  Allows you to choose a tag to enclose a selected portion of content. If there is no selection, the start and end tags are inserted at the cursor position.
• If the Position cursor between tags option (on page 212) is selected in the Content Completion preferences page, the cursor is placed between the start and end tag.
• If the Position cursor between tags option (on page 212) is not selected in the Content Completion preferences page, the cursor is placed at the end of the start tag, in an insert-attribute position.

Surround with '[tag]' (Ctrl + ForwardSlash (Command + ForwardSlash on macOS))
Surround the selected content with the last tag used.

Rename Element
The element from the cursor position, and any elements with the same name, can be renamed according with the options from the Rename dialog box.

Delete Element Tags
Deletes the tags of the closest element that contains the position of the cursor. This operation is also executed if the start or end tags of an element are deleted by pressing the Delete or Backspace keys.

Remove All Markup
Removes all the XML markup inside the selected block of content and keeps only the text content.

Remove Text
Removes the text content of the selected block of content and keeps the markup intact with empty elements.

DITA-related Refactoring Actions
A variety of built-in XML refactoring operations that pertain to DITA documents with some of the information preconfigured based upon the current context.

Change Topic ID to File Name
Use this operation to change the ID of a topic to be the same as its file name.

Convert CALS Tables to Simple Tables
Use this operation to convert DITA CALS tables to simple tables. If you invoke this operation from a nested table (a table inside a table), only the nested table will be affected. If it is invoked on a parent table that contains nested tables, all of the contained tables will be converted.

Convert conrefs to conkeyrefs
Use this operation to convert @conref attributes to @conkeyref attributes.

**Convert Simple Tables to CALS Tables**

Use this operation to convert DITA simple tables to CALS tables. If you invoke this operation from a nested table (a table inside a table), only the nested table will be affected. If it is invoked on a parent table that contains nested tables, all of the contained tables will be converted.

**Convert to Concept**

Use this operation to convert a DITA topic (of any type) to a DITA Concept topic type (for example, Topic to Concept).

**Convert to General Task**

Use this operation to convert a DITA topic (of any type) to a DITA General Task topic type (for example, Task to General Task). A DITA *General Task* is a less restrictive alternative to the *Strict Task* information type.

**Convert to Reference**

Use this operation to convert a DITA topic (of any type) to a DITA Reference topic type (for example, Topic to Reference).

**Convert to Task**

Use this operation to convert a DITA topic (of any type) to a DITA Task topic type (for example, Topic to Task).

**Convert to Topic**

Use this operation to convert a DITA topic (of any type) to a DITA Topic (for example, Task to Topic).

**Convert to Troubleshooting**

Use this operation to convert a DITA topic (of any type) to a DITA Troubleshooting topic type (for example, Topic to Troubleshooting).

**Rename Key**

Available when invoked on a key, and can be used to quickly rename a key. It also updates all references to it. Note that it does not work on DITA 1.3 key scopes.

**Generate IDs**

Use this operation to automatically generate unique IDs for elements.

*Attributes Refactoring Actions*
Contains built-in XML refactoring operations that pertain to attributes with some of the information preconfigured based upon the current context.

**Add/Change attribute**
Allows you to change the value of an attribute or insert a new one.

**Convert attribute to element**
Allows you to change an attribute into an element.

**Delete attribute**
Allows you to remove one or more attributes.

**Rename attribute**
Allows you to rename an attribute.

**Replace in attribute value**
Allows you to search for a text fragment inside an attribute value and change the fragment to a new value.

**Comments Refactoring Actions**
Contains built-in XML refactoring operations that pertain to comments with some of the information preconfigured based upon the current context.

**Delete comments**
Allows you to delete comments found inside one or more elements.

**Elements Refactoring Actions**
Contains built-in XML refactoring operations that pertain to elements with some of the information preconfigured based upon the current context.

**Delete element**
Allows you to delete elements.

**Delete element content**
Allows you to delete the content of elements.

**Insert element**
Allows you to insert new elements.

**Rename element**
Allows you to rename elements.

**Unwrap element**
Allows you to remove the surrounding tags of elements, while keeping the content unchanged.

**Wrap element**
Allows you to surround elements with element tags.

**Wrap element content**
Allows you to surround the content of elements with element tags.

**Fragments Refactoring Actions**
Contains built-in XML refactoring operations that pertain to XML fragments with some of the information preconfigured based upon the current context.

**Insert XML fragment**
Allows you to insert an XML fragment.

**Replace element content with XML fragment**
Allows you to replace the content of elements with an XML fragment.

**Replace element with XML fragment**
Allows you to replace elements with an XML fragment.

**Review submenu**
This submenu includes the following actions:

**Track Changes**
Enables or disables the *Track Changes* support for the current document.

**Accept Change(s) and Move to Next**
Accepts the *Tracked Change* located at the cursor position or all of the changes in a selection and then moves to the next change. If you select a part of a *deletion* or *insertion* change, only the selected content is accepted.

**Accept All Changes**
Accepts all *Tracked Changes* in the current document.

**Reject Change(s) and Move to Next**
Rejects the *Tracked Change* located at the cursor position or all of the changes in a selection and then moves to the next change. If you select a part of a *deletion* or *insertion* change, only the selected content is rejected.

**Reject All Changes**
Rejects all *Tracked Changes* in the current document.

**Comment Change**
Opens a dialog box that allows you to add a comment to an existing *Tracked Change*. The comment will appear in a callout and a tooltip when
hovering over the change. If the action is selected on an existing commented change, the dialog box will allow you to edit the comment.

Highlight

Enables the highlighting tool that allows you to mark text in your document.

Colors

Allows you to select the color for highlighting text.

Stop highlighting

Use this action to deactivate the highlighting tool.

Remove highlight(s)

Use this action to remove highlighting from the document.

Add Comment

Inserts a comment at the cursor position. The comment appears in a callout box and a tooltip (when hovering over the change).

Show/Edit Comment

Opens a dialog box that displays the discussion thread and allows the current user to edit comments that do not have replies. If you are not the author who inserted the original comment, the dialog box just displays the comment without the possibility of editing it.

Remove Comment

Removes a selected comment. If you remove a comment that contains replies, all of the replies will also be removed.

Manage Reviews

Opens the Review view (on page 664).

Manage IDs submenu

This submenu is available for topics that have an associated DTD or schema. It includes the following actions:

Rename in

Renames the ID and all its occurrences. Selecting this action opens the Rename XML ID dialog box. This dialog box lets you insert the new ID value and choose the scope of the rename operation.

Search References

Searches for the references of the ID. By default, the scope of this action is the current project. If you configure a scope using the Select the scope for the Search and Refactor operations (on page 831) dialog box, this scope will be used instead.
Search References in

Searches for the references of the ID. Selecting this action opens the Select the scope for the Search and Refactor operations (on page 831).

Search Occurrences in file

Searches for the occurrences of the ID in the current document.

Folding submenu

This submenu includes the following actions:

- **Toggle Fold**
  Toggles the state of the current fold.

- **Collapse Other Folds (Ctrl + NumPad/ (Command + NumPad/ on macOS))**
  Folds all the elements except the current element.

- **Collapse Child Folds (Ctrl + NumPad. (Command + NumPad. on macOS))**
  Folds the elements indented with one level inside the current element.

- **Expand Child Folds**
  Unfolds all child elements of the currently selected element.

- **Expand All (Ctrl + NumPad* (Command + NumPad* on macOS))**
  Unfolds all elements in the current document.

Inspect Styles

Opens the CSS Inspector view (on page 640) that allows you to examine the CSS rules that match the currently selected element.

Options

Opens the Author mode preferences page (on page 176) where you can configure various options with regard to the Author editing mode.

Floating Contextual Toolbar for DITA

Oxygen XML Editor includes a dynamic feature where certain editing contexts will trigger a floating toolbar with common actions that are available in the current editing context.

**Figure 468. DITA Floating Contextual Toolbar**

The floating contextual toolbar is automatically displayed when editing DITA documents in various situations, including:
• When a `<p>`, `<li>`, or `<shortdesc>` element has a selection inside, the floating toolbar includes actions such as **Bold**, *Italic*, _Underline_, a **Link** submenu, and more.

• When an `<image>` or `<xref>` element is selected:
  ◦ If the element has an `@href` attribute, the floating toolbar includes a URL chooser where you can select the appropriate target.
  ◦ If the element has a `@keyref` attribute, the floating toolbar includes a drop-down control where you can select the appropriate target key reference.

• When an `<object>` element is selected:
  ◦ If the element has a `@data` attribute, the floating toolbar includes a URL chooser where you can select the appropriate target.
  ◦ If the element has a `@datakeyref` attribute, the floating toolbar includes a drop-down control where you can select the appropriate target key reference.

• When an element with a `@conref` attribute is selected, the floating toolbar includes actions for editing, removing, or replacing content references.

• When a `<codeblock>` element is selected, the floating toolbar includes a drop-down control where you can select the value of the `@outputclass` attribute.

• When a `<ol>` element is selected, the floating toolbar includes actions for converting it to an ordered list or sorting the list.

• When an `<ul>` element is selected, the floating toolbar includes actions for converting it to an unordered list or sorting the list.

• When an `<li>` or `<step>` element is selected, the floating toolbar includes actions for moving the item up or down in the list/procedure.

• When a `<row>` or `<strow>` element is selected in a table, the floating toolbar includes various table-related actions (such as actions for editing table properties, inserting rows, or deleting rows).

• When an `<entry>` or `<stentry>` element is selected in a table, the floating toolbar includes various table-related actions (such as actions for editing table properties, inserting/deleting rows, or inserting/deleting columns).

• When an `<table>` or `<simpletable>` element is selected, the floating toolbar includes actions for editing table properties or sorting the table.

**DITA Drag/Drop (or Copy/Paste) Actions**

Dragging a file from the Project view (on page 402) or DITA Maps Manager view (on page 2897) and dropping it into a DITA document that is edited in Author mode, creates a link to the dragged file (the `<xref>` DITA element with the `@href` attribute) at the drop location. Copy and paste actions work the same.

You can also drag images or media files from your system explorer or the Project view (on page 402) and drop them into a DITA document (or copy and paste). This will insert the appropriate element at the drop or paste location (for example, dropping/pasting an image will insert the DITA `<image>` element with an `@href` attribute).
DITA Map Document Type (Framework)

*DITA maps (on page 3218)* are documents that collect and organize references to DITA topics to indicate the relationships between the topics. They can be used as a container for topics used to transform a collection of content into a publication and they offer a sequence and structure to the topics. They can also serve as outlines or tables of contents for DITA deliverables and as build manifests for DITA projects. *DITA maps* allow scalable reuse of content across multiple contexts. Maps can reference topics or other maps, and can contain a variety of content types and metadata.

For much more detailed information, resources, and instructions, see the DITA Authoring (on page 2886) chapter.

**File Definition**

A file is considered to be a *DITA map* document when one of the following conditions are true:

- The root element name is one of the following: `<map>`, `<bookmap>`.
- The public ID of the document is `-//OASIS//DTD DITA Map` or `-//OASIS//DTD DITA BookMap`.
- The root element of the file has a `@class` attribute that contains the value `map/map` and a `@DITAArchVersion` attribute from the `http://dita.oasis-open.org/architecture/2005/` namespace. This enhanced case of matching is only applied when the Enable DTD/XML Schema processing in document type detection option (on page 139) from the Document Type Association preferences page (on page 138) is selected.

**Default Document Templates**

There are a variety of default *DITA map* templates available when creating new documents from templates (on page 368) and they can be found in various folders inside: Framework Templates > DITA Map.

The default templates for *DITA map* documents are located in the `{OXYGEN_INSTALL_DIR}/frameworks/sita/templates/map` folder.

**Default Schema for Validation and Content Completion**

Default schemas that are used if one is not detected in the *DITA map* document are stored in the various folders inside `DITA-OT-DIR/dtd/` or `DITA-OT-DIR/schema/`. 

**Tip:**

For information about customizing Author mode actions for a particular framework (on page 3219) (document type), see the Customizing the Author Mode Editing Experience for a Framework (on page 2145) section.

Related Information:

Customizing the Author Mode Editing Experience for a Framework (on page 2145)
Default CSS

The default CSS files used for rendering DITA content in **Author** mode are stored in the various folders inside: 

`{OXYGEN_INSTALL_DIR}/frameworks/dita/css/`

By default, these default CSS files are merged with any that are specified in the document. For more information, see [Configuring and Managing Multiple CSS Styles for a Framework](on page 2145).

Default XML Catalogs

The default **XML Catalogs** (on page 3225) for the **DITA map** document type are as follows:

- `{OXYGEN_INSTALL_DIR}/frameworks/dita/catalog.xml`
- `DITA-OT-DIR/catalog-dita.xml`

Transformation Scenarios

Oxygen XML Editor includes numerous built-in transformation scenarios that allow you to transform **DITA maps** to a variety of outputs, such as WebHelp, PDF, ODF, XHTML, EPUB, and CHM. All of them are listed in the **DITA Map** section in the [Configure Transformation Scenario(s)](on page 1548) dialog box.

For more information, see the **DITA Map Transformation Scenarios** (on page 3081) section.

Resources

- DITA Specifications
- DITA Style Guide Best Practices for Authors
- Oxygen Video Tutorial: DITA Maps Manager

Related Information:

- [Selecting a Root Map](on page 2914)
- DITA Authoring (on page 2886)
- Getting Started with DITA (on page 2887)
- Editing XML Documents in Author Mode (on page 588)
- Editing XML Documents in Text Mode (on page 517)

DITA Map Author Mode Actions

A variety of actions are available for DITA map documents that can be found in **DITA** menu, toolbar, contextual menu, and the **Content Completion Assistant** (on page 3217).

DITA Map Toolbar and Menu Actions

When a **DITA map** is opened in **Author** mode, the following default actions are available on the DITA Map toolbar (by default, they are also available in the **DITA** menu and in various submenus of the contextual menu):
Insert New DITA Resource

Opens a New DITA file dialog box (on page 2961) where you can choose the type of DITA document to create and inserts a reference to it at the current position within the map.

Insert Topic Reference

Opens the Insert Reference dialog box (on page 2923) where you can configure a topic reference and inserts it at the current position within the map.

Insert Key Definition with Keyword

Opens a dialog box where you can choose the name of a key and its keyword value and inserts the key definition at the current position within the map.

Reuse Content

Opens the Reuse Content dialog box (on page 3043) that allows you to insert and configure a content reference (@conref), or a content key reference (@conkeyref) at the cursor position.

Insert Topic Heading

Opens the Insert Reference dialog box (on page 2923) that allows you to insert a topic heading at the cursor position.

Insert Topic Group

Opens the Insert Reference dialog box (on page 2923) that allows you to insert a topic group at the cursor position.

Insert Relationship Table

Opens a dialog box that allows you to configure the relationship table to be inserted. The dialog box allows you to configure the number of rows and columns of the relationship table, if the header will be generated and if the title will be added.

Relationship Table Properties

Allows you to change the properties of rows in relationship tables.

Insert Relationship Row

Inserts a new table row with empty cells. The action is available when the cursor position is inside a table.

Insert Relationship Column

Inserts a new table column with empty cells after the current column. The action is available when the cursor position is inside a table.

Delete Relationship Column

Deletes the table column where the cursor is located.

Delete Relationship Row
Deletes the table row where the cursor is located.

†↑ Move Up

Moves the selected node up one position on its same level.

↓↓ Move Down

Moves the selected node down one position on its same level.

← Promote (Alt + LeftArrow)

Moves the selected node up one level to the level of its parent node.

→ Demote (Alt + RightArrow)

Moves the selected node down one level to the level of its child nodes.

DITA Map Contextual Menu Actions

The following actions are available in the contextual menu when editing in Author mode (most of them are also available in the DITA menu at the top of the interface):

Add File to Review Task

This action can be used to add the current document to a task in the Content Fusion Tasks Manager view. Oxygen Content Fusion is a flexible, intuitive collaboration platform designed to adapt to any type of documentation review workflow. This functionality is available through a pre-installed connector add-on (on page 2527). To fully take advantage of all of the benefits and features of Content Fusion, your organization will need an Oxygen Content Fusion Enterprise Server. For more information, see the Oxygen Content Fusion website.

Edit Properties

Opens the Edit Properties dialog box that allows you to configure the properties of a selected node. For more details about this dialog box, see Edit Properties Dialog Box (on page 2932).

Cut (Ctrl + X (Command + X on macOS))

Removes the currently selected content from the document and places it in the clipboard.

Copy (Ctrl + C (Command + C on macOS))

Places a copy of the currently selected content in the clipboard.

Paste (Ctrl + V (Command + V on macOS))

Inserts the current clipboard content into the document at the cursor position.

Paste special submenu

This submenu includes the following special paste actions that are specific to the DITA framework:

Paste as content reference
Inserts a content reference (a DITA element with a `@conref` attribute) to the DITA XML element from the clipboard. An entire DITA XML element with an ID attribute must be present in the clipboard when the action is invoked. The `conref` attribute will point to this ID value.

**Paste as content key reference**

Allows you to indirectly reference content using the `@conkeyref` attribute. When the DITA content is processed, the key references are resolved using key definitions from *DITA maps (on page 3218)*. To use this action, you must first do the following:

1. Make sure the DITA element that contains the copied content has an ID attribute assigned to it.
2. In the *DITA Maps Manager* view, make sure that the *Context combo box (on page 2901)* points to the correct map that stores the keys.
3. Make sure the topic that contains the content you want to reference has a key assigned to it. To assign a key, right-click the topic with its parent map opened in the *DITA Maps Manager*, select *Edit Properties*, and enter a value in the *Keys* field.

**Paste as link**

Looks for the first element with an ID value in the clipboard and inserts an `<xref>` that points to that element. If no elements with an ID value are found, a message will appear that informs you that to use this action, the clipboard contents must include at least one element with a declared ID.

**Paste as link (keyref)**

Inserts a link to the element that you want to reference. To use this action, you must first do the following:

1. Make sure the DITA element that contains the copied content has an ID attribute assigned to it.
2. In the *DITA Maps Manager* view, make sure that the *Context combo box (on page 2901)* points to the correct map that stores the keys.
3. Make sure the topic that contains the content you want to reference has a key assigned to it. To assign a key, right-click the topic with its parent map opened in the *DITA Maps Manager*, select *Edit Properties*, and enter a value in the *Keys* field.

**Insert submenu**

This submenu includes the following insert actions that are specific to the DITA Map framework:

- **Insert New DITA Resource**
Opens a New DITA file dialog box *(on page 2961)* where you can choose the type
of DITA document to create and inserts a reference to it at the current position
within the map.

**Insert Topic Reference**

Opens the Insert Reference dialog box *(on page 2923)* where you can configure a
topic reference and inserts it at the current position within the map.

**Insert Key Definition with Keyword**

Opens a dialog box where you can choose the name of a key and its keyword value
and inserts the key definition at the current position within the map.

**Reuse Content**

Opens the Reuse Content dialog box *(on page 3043)* that allows you to insert and
configure a content reference (@conref), or a content key reference (@conkeyref) at
the cursor position.

**Insert Topic Heading**

Opens the Insert Reference dialog box *(on page 2923)* that allows you to insert a
topic heading at the cursor position.

**Insert Topic Group**

Opens the Insert Reference dialog box *(on page 2923)* that allows you to insert a
topic group at the cursor position.

**Insert Entity**

Allows you to insert a predefined entity or character entity. Surrogate character
entities (range #x10000 to #x10FFFF) are also accepted. Character entities can be
entered in one of the following forms:

- `<decimal value>` - e.g. #65
- `&<decimal value>` - e.g. &#65
- `#x<hexadecimal value>` - e.g. #x41
- `&#x<hexadecimal value>` - e.g. &#x41

**Relationship Table > Insert Relationship Table**

Opens a dialog box that allows you to configure the relationship table to be inserted. The dialog
box allows you to configure the number of rows and columns of the relationship table, if the
header will be generated and if the title will be added.

**Generate IDs**

Oxygen XML Editor generates unique IDs for the current element (or elements), depending on
how the action is invoked:
• When invoked on a single selection, an ID is generated for the selected element at the cursor position.
• When invoked on a block of selected content, IDs are generated for all top-level elements and elements listed in the ID Options dialog box that are found in the current selection.

Note:
The Generate IDs action does not overwrite existing ID values. It only affects elements that do not already have an @id attribute.

Search References

Finds the references to the @href or @keys attribute value of the topic/map reference element at the current cursor position, in all the topics from the current DITA map (opened in the DITA Maps Manager view (on page 2897)). The current topic/map reference element must have an @href or @keys attribute defined to complete the search.

Show Key Definition

Available for elements that have a @conkeyref or @keyref attribute set (or elements with an ancestor element that has a @conkeyref or @keyref attribute). It computes the key name and opens the DITA map (on page 3218) that contains the definition of the key with the element that defines that key selected.

Select submenu

This submenu allows you to select the following:

Element

Selects the entire element at the current cursor position.

Content

Selects the entire content of the element at the current cursor position, excluding the start and end tag. Performing this action repeatedly will result in the selection of the content of the ancestor of the currently selected element content.

Parent

Selects the entire parent element at the current cursor position.

Text submenu

This submenu contains the following actions:

To Lower Case

Converts the selected content to lower case characters.

To Upper Case

Converts the selected content to upper case characters.
Capitalize Sentences

Converts to upper case the first character of every selected sentence.

Capitalize Words

Converts to upper case the first character of every selected word.

Count Words

Counts the number of words and characters (no spaces) in the entire document or in the selection for regular content and read-only content.

Note:
The content marked as deleted with change tracking (on page 3224) is ignored when counting words.

Convert Hexadecimal Sequence to Character (Ctrl + Shift + X (Command + Shift + X on macOS))

Converts a sequence of hexadecimal characters to the corresponding Unicode character (on page 463). The action can be invoked if there is a selection containing a valid hexadecimal sequence or if the cursor is placed at the right side of a valid hexadecimal sequence. A valid hexadecimal sequence can be composed of 2 to 4 hexadecimal characters and may or may not be preceded by the 0x or 0X prefix. Examples of valid sequences and the characters they will be converted to:

- \0x0045 will be converted to E
- \0x125 to ĥ
- \265 to ɥ
- \2190 to ←

Note:
For more information about finding the hexadecimal value of a character, see Finding the Decimal, Hexadecimal, or Character Entity Equivalent (on page 466).

Refactoring submenu

Contains a series of actions designed to alter the XML structure of the document:

- Toggle Comment
  Encloses the currently selected text in an XML comment, or removes the comment if it is commented.

- Move Up (Alt + UpArrow)
  Moves the current node or selected nodes in front of the previous node.
Move Down (Alt + DownArrow)

Moves the current node or selected nodes after the subsequent node.

Split Element (Alt + Shift + D (Ctrl + Alt + D on macOS))

Splits the content of the closest element that contains the position of the cursor. Thus, if the cursor is positioned at the beginning or at the end of the element, the newly created sibling will be empty.

Join Elements

Joins two adjacent block elements (on page 3216) that have the same name. The action is available only when the cursor position is between the two adjacent block elements. Also, joining two block elements can be done by pressing the Delete or Backspace keys and the cursor is positioned between the boundaries of these two elements.

Surround with Tags (Ctrl + E (Command + E on macOS))

Allows you to choose a tag to enclose a selected portion of content. If there is no selection, the start and end tags are inserted at the cursor position.

- If the Position cursor between tags option (on page 212) is selected in the Content Completion preferences page, the cursor is placed between the start and end tag.
- If the Position cursor between tags option (on page 212) is not selected in the Content Completion preferences page, the cursor is placed at the end of the start tag, in an insert-attribute position.

Surround with '[tag]' (Ctrl + ForwardSlash (Command + ForwardSlash on macOS))

Surround the selected content with the last tag used.

Rename Element

The element from the cursor position, and any elements with the same name, can be renamed according with the options from the Rename dialog box.

Delete Element Tags

Deletes the tags of the closest element that contains the position of the cursor. This operation is also executed if the start or end tags of an element are deleted by pressing the Delete or Backspace keys.

Remove All Markup

Removes all the XML markup inside the selected block of content and keeps only the text content.

Remove Text
Removes the text content of the selected block of content and keeps the markup intact with empty elements.

**Attributes Refactoring Actions**

Contains built-in XML refactoring operations that pertain to attributes with some of the information preconfigured based upon the current context.

- **Add/Change attribute**
  Allows you to change the value of an attribute or insert a new one.

- **Convert attribute to element**
  Allows you to change an attribute into an element.

- **Delete attribute**
  Allows you to remove one or more attributes.

- **Rename attribute**
  Allows you to rename an attribute.

- **Replace in attribute value**
  Allows you to search for a text fragment inside an attribute value and change the fragment to a new value.

**Comments Refactoring Actions**

Contains built-in XML refactoring operations that pertain to comments with some of the information preconfigured based upon the current context.

- **Delete comments**
  Allows you to delete comments found inside one or more elements.

**Elements Refactoring Actions**

Contains built-in XML refactoring operations that pertain to elements with some of the information preconfigured based upon the current context.

- **Delete element**
  Allows you to delete elements.

- **Delete element content**
  Allows you to delete the content of elements.

- **Insert element**
  Allows you to insert new elements.

- **Rename element**
  Allows you to rename elements.

- **Unwrap element**
Allows you to remove the surrounding tags of elements, while keeping the content unchanged.

Wrap element
Allows you to surround elements with element tags.

Wrap element content
Allows you to surround the content of elements with element tags.

Fragments Refactoring Actions
Contains built-in XML refactoring operations that pertain to XML fragments with some of the information preconfigured based upon the current context.

Insert XML fragment
Allows you to insert an XML fragment.

Replace element content with XML fragment
Allows you to replace the content of elements with an XML fragment.

Replace element with XML fragment
Allows you to replace elements with an XML fragment.

Review submenu
This submenu includes the following actions:

✔ Track Changes
   Enables or disables the Track Changes (on page 3224) support for the current document.

✔ Accept Change(s) and Move to Next
   Accepts the Tracked Change (on page 3224) located at the cursor position or all of the changes in a selection and then moves to the next change. If you select a part of a deletion or insertion change, only the selected content is accepted.

✔ Accept All Changes
   Accepts all Tracked Changes (on page 3224) in the current document.

❌ Reject Change(s) and Move to Next
   Rejects the Tracked Change (on page 3224) located at the cursor position or all of the changes in a selection and then moves to the next change. If you select a part of a deletion or insertion change, only the selected content is rejected.

❌ Reject All Changes
   Rejects all Tracked Changes (on page 3224) in the current document.

💬 Comment Change
Opens a dialog box that allows you to add a comment to an existing Tracked Change (on page 3224). The comment will appear in a callout and a tooltip when hovering over the change. If the action is selected on an existing commented change, the dialog box will allow you to edit the comment.

Highlight

Enables the highlighting tool that allows you to mark text in your document.

Colors

Allows you to select the color for highlighting text.

Stop highlighting

Use this action to deactivate the highlighting tool.

Remove highlight(s)

Use this action to remove highlighting from the document.

Add Comment

Inserts a comment at the cursor position. The comment appears in a callout box and a tooltip (when hovering over the change).

Show/Edit Comment

Opens a dialog box that displays the discussion thread and allows the current user to edit comments that do not have replies. If you are not the author who inserted the original comment, the dialog box just displays the comment without the possibility of editing it.

Remove Comment

Removes a selected comment. If you remove a comment that contains replies, all of the replies will also be removed.

Manage Reviews

Opens the Review view (on page 664).

Folding submenu

This submenu includes the following actions:

Toggle Fold

Toggles the state of the current fold.

Collapse Other Folds (Ctrl + NumPad/ (Command + NumPad/ on macOS))

Folds all the elements except the current element.

Collapse Child Folds (Ctrl + NumPad. (Command + NumPad. on macOS))

Folds the elements indented with one level inside the current element.
Expand Child Folds

Unfolds all child elements of the currently selected element.

Expand All (Ctrl + NumPad* (Command + NumPad* on macOS))

Unfolds all elements in the current document.

About Element > Go to Definition

Moves the cursor to the definition of the current element.

Inspect Styles

Opens the CSS Inspector view (on page 640) that allows you to examine the CSS rules that match the currently selected element.

Options

Opens the Author mode preferences page (on page 176) where you can configure various options with regard to the Author editing mode.

Floating Contextual Toolbar for DITA

Oxygen XML Editor includes a dynamic feature where certain editing contexts will trigger a floating toolbar with common actions that are available in the current editing context.

The floating contextual toolbar is automatically displayed when editing DITA map documents when a <topicref> element is selected and it includes actions for moving the topic reference node up or down (or promoting/demoting the node).

DITA Map Drag/Drop Actions

Dragging a file from the Project view (on page 402) or DITA Maps Manager view (on page 2897) and dropping it into a DITA map document that is edited in Author mode creates a link to the dragged file (a <topicref> element, <chapter>, <part>, etc.) at the drop location.

Opening a Topic from a DITA Map in Author Mode

If a DITA map is open in the Author visual editing mode, you can open a referenced topic by clicking the icon to the left of the particular topic. The source topic is opened in a new tab in the main editor.

Tip:

For information about customizing Author mode actions for a particular framework (on page 3219) (document type), see the Customizing the Author Mode Editing Experience for a Framework (on page 2145) section.

Related Information:

Customizing the Author Mode Editing Experience for a Framework (on page 2145)
XHTML Document Type (Framework)

The Extensible HyperText Markup Language (XHTML), is a markup language that has the same depth of expression as HTML, but also conforms to XML syntax.

File Definition

A file is considered to be an XHTML document when the root element is `<html>`.

Default Document Templates

There are a variety of default XHTML templates available when creating new documents from templates (on page 368) and they can be found in: Framework Templates > XHTML.

The default templates for XHTML documents are located in the `{OXYGEN_INSTALL_DIR}/frameworks/xhtml/templates/` folder.

Default Schema for Validation and Content Completion

Default schemas that are used if one is not detected in the XHTML file are stored in the following locations:

- **XHTML 1.0** - `{OXYGEN_INSTALL_DIR}/frameworks/xhtml/dtd/` or `{OXYGEN_INSTALL_DIR}/frameworks/xhtml/nvdl/`.
- **XHTML 1.1** - `{OXYGEN_INSTALL_DIR}/frameworks/xhtml11/dtd/` or `{OXYGEN_INSTALL_DIR}/frameworks/xhtml11/schema/`.
- **XHTML 5** - `{OXYGEN_INSTALL_DIR}/frameworks/xhtml/xhtml5 (epub3)/`.

Default CSS

The default CSS files used for rendering XHTML content in Author mode are stored in `{OXYGEN_INSTALL_DIR}/frameworks/xhtml/css/`.

By default, these default CSS files are merged with any that are specified in the document. For more information, see Configuring and Managing Multiple CSS Styles for a Framework (on page 2145).

Default XML Catalogs

The default XML Catalogs (on page 3225) for the XHTML document type are as follows:

- `{OXYGEN_INSTALL_DIR}/frameworks/xhtml/dtd/xhtmlcatalog.xml`
- `{OXYGEN_INSTALL_DIR}/frameworks/relaxng/catalog.xml`
- `{OXYGEN_INSTALL_DIR}/frameworks/nvdl/catalog.xml`
- `{OXYGEN_INSTALL_DIR}/frameworks/xhtml11/dtd/xhtmlcatalog.xml`
- `{OXYGEN_INSTALL_DIR}/frameworks/xhtml11/schema/xhtmlcatalog.xml`
- `{OXYGEN_INSTALL_DIR}/xhtml5 (epub3)/catalog-compat.xml`
Transformation Scenarios

Oxygen XML Editor includes built-in transformation scenarios that allow you to transform XHTML documents to several types of DITA document types (topic, task, concept, reference). They can be found in the XHTML section in the Configure Transformation Scenario(s) dialog box (on page 1548).

Related Information:
- Editing HTML Documents (on page 1255)
- Editing XML Documents in Text Mode (on page 517)
- Editing XML Documents in Author Mode (on page 588)
- Adding Tables in XHTML Documents (on page 710)
- XHTML Specifications

XHTML Validation

XHTML documents can be validated in Oxygen XML Editor using the same validation features as with any other XML document (on page 772). In addition, Oxygen XML Editor includes a built-in validator engine (W3C XHTML Validator) based upon the W3C Nu HTML Checker that can be used to validate HTML or XHTML documents.

To use the W3C XHTML Validator engine:

1. Create or edit a validation scenario (on page 787) (e.g. select the Configure Validation Scenario(s) from the Validation toolbar drop-down menu).
2. Change the File type column to XML Document and select W3C XHTML Validator in the Validation engine column.
3. Click OK and Apply Associated to run the validation.

Related Information:
- W3C Nu HTML Checker
- Validating XML Documents (on page 772)
- Batch Validation and Transformation (on page 414)

XHTML Author Mode Actions

A variety of actions are available for XHTML documents that can be found in XHTML menu, toolbar, contextual menu, and the Content Completion Assistant (on page 3217).
The following default actions are available on the XHTML toolbar when editing in Author mode (by default, they are also available in the XHTML menu and some of them are in various submenus of the contextual menu):

- **Bold**
  Changes the style of the selected text to *bold* by surrounding it with the `<b>` tag. You can use this action on multiple non-contiguous selections.

- **Italic**
  Changes the style of the selected text to *italic* by surrounding it with `<i>` tag. You can use this action on multiple non-contiguous selections.

- **Underline**
  Changes the style of the selected text to *underline* by surrounding it with `<u>` tag. You can use this action on multiple non-contiguous selections.

- **Link**
  Inserts an `<a>` element with an `@href` attribute at the cursor position. You can type the URL of the reference you want to insert or use the browsing actions in the **Browse** drop-down menu.

- **Insert Image**
  Inserts a graphic object at the cursor position. This is done by inserting an `<img>` element regardless of the current context. The following graphical formats are supported: GIF, JPG, JPEG, BMP, PNG, SVG.

- **Insert Media Resource**
  Opens a **Choose Media** dialog box (on page 747) that allows you to select the URL of a media object to be inserted into a document at the cursor position. The result will be that a reference to the specified video, audio, or embedded HTML frame is inserted and rendered in Author mode so that it can be played directly from there.

- **Headings Drop-down Menu**
  A drop-down menu that includes actions for inserting `<h1>`, `<h2>`, `<h3>`, `<h4>`, `<h5>`, `<h6>` elements.

- **Insert Paragraph**
  Insert a new paragraph element at current cursor position.

- **Insert Equation**
  Opens the **XML Fragment Editor** that allows you to insert and edit MathML notations (on page 749).

- **Insert List Item**
  Inserts a list item in the current list type.
**Insert Unordered List**

Inserts an unordered list at the cursor position. A child list item is also automatically inserted by default. You can also use this action to convert selected paragraphs or other types of lists to an unordered list.

**Insert Ordered List**

Inserts an ordered list at the cursor position. A child list item is also automatically inserted by default. You can also use this action to convert selected paragraphs or other types of lists to an ordered list.

**Insert a definition list at the cursor position**

Inserts a definition list (\texttt{<dl> element}) with one list item (a \texttt{<dt>} child element and a \texttt{<dd>} child element). You can also use this action to convert selected paragraphs or other types of lists to a definition list.

**Sort**

Sorts cells or list items in a table.

**Insert Table**

Opens a dialog box that allows you to configure and insert a table. You can generate a header and footer, set the number of rows and columns of the table and decide how the table is framed. You can also use this action to convert selected paragraphs, lists, and inline content (mixed content, text plus markup, that is rendered inside a \texttt{block element (on page 3216)}) into a table, with the selected content inserted in the first column, starting from the first row after the header (if a header is inserted).

**Note:**

If the selection contains a mixture of elements that cannot be converted, you will receive an error message saying that \textbf{Only lists, paragraphs, or inline content can be converted to tables.}

**Insert Row**

Inserts a new table row with empty cells below the current row. This action is available when the cursor is positioned inside a table.

**Insert Row Above**

Inserts a new table row with empty cells above the current row. This action is available when the cursor is positioned inside a table.

**Insert Column**

Inserts a new table column with empty cells after the current column. This action is available when the cursor is positioned inside a table.

**Insert Cell**
Inserts a new empty cell depending on the current context. If the cursor is positioned between two cells, Oxygen XML Editor a new cell at the cursor position. If the cursor is inside a cell, the new cell is created after the current cell.

**Delete Column(s)**

Deletes the table column located at the cursor position or multiple columns in a selection.

**Delete Row(s)**

Deletes the table row located at the cursor position or multiple rows in a selection.

**Join Cells**

Joins the content of the selected cells (both horizontally and vertically).

**Split Cell**

Splits the cell at the cursor location. If Oxygen XML Editor detects more than one option to split the cell, a dialog box will be displayed that allows you to select the number of rows or columns to split the cell into.

### XHTML Contextual Menu Actions

The following actions are available in the contextual menu when editing in Author mode (most of them are also available in the XHTML menu at the top of the interface):

**Add File to Review Task**

This action can be used to add the current document to a task in the Content Fusion Tasks Manager view. Oxygen Content Fusion is a flexible, intuitive collaboration platform designed to adapt to any type of documentation review workflow. This functionality is available through a pre-installed connector add-on (on page 2527). To fully take advantage of all of the benefits and features of Content Fusion, your organization will need an Oxygen Content Fusion Enterprise Server. For more information, see the Oxygen Content Fusion website.

**Edit Attributes**

Displays an in-place attributes editor (on page 630) that allows you to manage the attributes of an element.

**Edit Profiling Attributes**

Allows you to change the profiling attributes (on page 668) defined on all selected elements.

**Cut (Ctrl + X (Command + X on macOS))**

Removes the currently selected content from the document and places it in the clipboard.

**Copy (Ctrl + C (Command + C on macOS))**

Places a copy of the currently selected content in the clipboard.

**Paste (Ctrl + V (Command + V on macOS))**

Inserts the current clipboard content into the document at the cursor position.
Image Map Editor

This action is available in the contextual menu when it is invoked on an image. This action applies an image map to the current image (if one does not already exist) and opens the **Image Map Editor** dialog box. This feature allows you to create hyperlinks in specific areas of an image that will link to various destinations.

Insert submenu

This submenu includes the following insert actions:

- **Insert Table**
  
  Opens a dialog box that allows you to configure and insert a table. You can generate a header and footer, set the number of rows and columns of the table and decide how the table is framed. You can also use this action to convert selected paragraphs, lists, and inline content (mixed content, text plus markup, that is rendered inside a block element (on page 3216)) into a table, with the selected content inserted in the first column, starting from the first row after the header (if a header is inserted).

  **Note:**
  
  If the selection contains a mixture of elements that cannot be converted, you will receive an error message saying that Only lists, paragraphs, or inline content can be converted to tables.

- **Insert Link**
  
  Inserts an `<a>` element with an `@href` attribute at the cursor position. You can type the URL of the reference you want to insert or use the browsing actions in the `Browse` drop-down menu.

- **Insert Image**
  
  Inserts an image reference (on page 719) at the cursor position. Depending on the current location, an image-type element is inserted.

- **Insert Media Resource**
  
  Opens a Choose Media dialog box (on page 747) that allows you to select the URL of a media object to be inserted into a document at the cursor position. The result will be that a reference to the specified video, audio, or embedded HTML frame is inserted and rendered in Author mode so that it can be played directly from there.

- **Insert Equation**
  
  Opens the XML Fragment Editor that allows you to insert and edit MathML notations (on page 749).

- **Insert Paragraph**
Inserts a new paragraph element at current cursor position.

**Headings Drop-down Menu**

A drop-down menu that includes actions for inserting `<h1>`, `<h2>`, `<h3>`, `<h4>`, `<h5>`, `<h6>` elements.

**Insert Entity**

Allows you to insert a predefined entity or character entity. Surrogate character entities (range #x10000 to #x10FFFF) are also accepted. Character entities can be entered in one of the following forms:

- `#<decimal value>` - e.g. #65
- `&#<decimal value>` - e.g. &#65
- `#x<hexadecimal value>` - e.g. #x41
- `&#x<hexadecimal value>` - e.g. &#x41

**Style submenu**

This submenu includes the following text styling actions:

- **Bold**
  Emphasizes the selected text by surrounding it with a `bold` tag. You can use this action on multiple non-contiguous selections.

- **Italic**
  Emphasizes the selected text by surrounding it with an `italic` tag. You can use this action on multiple non-contiguous selections.

- **Underline**
  Emphasizes the selected text by surrounding it with an `underline` tag. You can use this action on multiple non-contiguous selections.

- **Subscript**
  Surrounds the selected text with a `subscript` tag, used for inserting a character (number, letter, or symbol) that will appear slightly below the baseline and slightly smaller than the rest of the text.

- **Superscript**
  Surrounds the selected text with a `superscript` tag, used for inserting a character (number, letter, or symbol) that will appear slightly above the baseline and slightly smaller than the rest of the text.

**Table actions**

The following table editing actions are available in the contextual menu when it is invoked on a table:
Insert Rows

Opens a dialog box that allows you to insert any number of rows and specify the position where they will be inserted (Above or Below the current row).

Delete Row(s)

Deletes the table row located at the cursor position or multiple rows in a selection.

Insert Columns

Opens a dialog box that allows you to insert any number of columns and specify the position where they will be inserted (Above or Below the current column).

Delete Column(s)

Deletes the table column located at the cursor position or multiple columns in a selection.

Join Cells

Joins the content of the selected cells (both horizontally and vertically).

Split Cell

Splits the cell at the cursor location. If Oxygen XML Editor detects more than one option to split the cell, a dialog box will be displayed that allows you to select the number of rows or columns to split the cell into.

Sort

Sorts cells or list items in a table.

Table Properties

Opens the Table properties dialog box that allows you to configure properties of a table (such as frame borders).

Other Actions submenu

This submenu gives you access to all the usual contextual menu actions.

Select submenu

This submenu allows you to select the following:

Element

Selects the entire element at the current cursor position.

Content

Selects the entire content of the element at the current cursor position, excluding the start and end tag. Performing this action repeatedly will result in the selection of the content of the ancestor of the currently selected element content.

Parent

Selects the entire parent element at the current cursor position.
Text submenu

This submenu contains the following actions:

**To Lower Case**

Converts the selected content to lower case characters.

**To Upper Case**

Converts the selected content to upper case characters.

**Capitalize Sentences**

Converts to upper case the first character of every selected sentence.

**Capitalize Words**

Converts to upper case the first character of every selected word.

**Count Words**

Counts the number of words and characters (no spaces) in the entire document or in the selection for regular content and read-only content.

**Note:**

The content marked as deleted with change tracking (on page 3224) is ignored when counting words.

**Convert Hexadecimal Sequence to Character (Ctrl + Shift + X (Command + Shift + X on macOS))**

Converts a sequence of hexadecimal characters to the corresponding Unicode character (on page 463). The action can be invoked if there is a selection containing a valid hexadecimal sequence or if the cursor is placed at the right side of a valid hexadecimal sequence. A valid hexadecimal sequence can be composed of 2 to 4 hexadecimal characters and may or may not be preceded by the 0x or 0X prefix. Examples of valid sequences and the characters they will be converted to:

- 0x0045 will be converted to E
- 0x0125 to ĥ
- 265 to ɥ
- 2190 to ←

**Note:**

For more information about finding the hexadecimal value of a character, see Finding the Decimal, Hexadecimal, or Character Entity Equivalent (on page 466).

Refactoring submenu
Contains a series of actions designed to alter the XML structure of the document:

- **Toggle Comment**
  
  Encloses the currently selected text in an XML comment, or removes the comment if it is commented.

- **Move Up (Alt + UpArrow)**
  
  Moves the current node or selected nodes in front of the previous node.

- **Move Down (Alt + DownArrow)**
  
  Moves the current node or selected nodes after the subsequent node.

- **Split Element (Alt + Shift + D (Ctrl + Alt + D on macOS))**
  
  Splits the content of the closest element that contains the position of the cursor. Thus, if the cursor is positioned at the beginning or at the end of the element, the newly created sibling will be empty.

- **Join Elements**
  
  Joins two adjacent block elements (on page 3216) that have the same name. The action is available only when the cursor position is between the two adjacent block elements. Also, joining two block elements can be done by pressing the **Delete** or **Backspace** keys and the cursor is positioned between the boundaries of these two elements.

- **Surround with Tags (Ctrl + E (Command + E on macOS))**
  
  Allows you to choose a tag to enclose a selected portion of content. If there is no selection, the start and end tags are inserted at the cursor position.

  - If the **Position cursor between tags** option (on page 212) is selected in the **Content Completion** preferences page, the cursor is placed between the start and end tag.
  - If the **Position cursor between tags** option (on page 212) is not selected in the **Content Completion** preferences page, the cursor is placed at the end of the start tag, in an insert-attribute position.

- **Surround with ‘[tag]’ (Ctrl + ForwardSlash (Command + ForwardSlash on macOS))**
  
  Surround the selected content with the last tag used.

- **Rename Element**
  
  The element from the cursor position, and any elements with the same name, can be renamed according with the options from the **Rename** dialog box.

- **Delete Element Tags**
Deletes the tags of the closest element that contains the position of the cursor. This operation is also executed if the start or end tags of an element are deleted by pressing the **Delete** or **Backspace** keys.

Remove All Markup

Removes all the XML markup inside the selected block of content and keeps only the text content.

Remove Text

Removes the text content of the selected block of content and keeps the markup intact with empty elements.

Attributes Refactoring Actions

Contains built-in XML refactoring operations that pertain to attributes with some of the information preconfigured based upon the current context.

- **Add/Change attribute**
  Allows you to change the value of an attribute or insert a new one.

- **Convert attribute to element**
  Allows you to change an attribute into an element.

- **Delete attribute**
  Allows you to remove one or more attributes.

- **Rename attribute**
  Allows you to rename an attribute.

- **Replace in attribute value**
  Allows you to search for a text fragment inside an attribute value and change the fragment to a new value.

Comments Refactoring Actions

Contains built-in XML refactoring operations that pertain to comments with some of the information preconfigured based upon the current context.

- **Delete comments**
  Allows you to delete comments found inside one or more elements.

Elements Refactoring Actions

Contains built-in XML refactoring operations that pertain to elements with some of the information preconfigured based upon the current context.

- **Delete element**
  Allows you to delete elements.
Delete element content
Allows you to delete the content of elements.

Insert element
Allows you to insert new elements.

Rename element
Allows you to rename elements.

Unwrap element
Allows you to remove the surrounding tags of elements, while keeping the content unchanged.

Wrap element
Allows you to surround elements with element tags.

Wrap element content
Allows you to surround the content of elements with element tags.

Fragments Refactoring Actions
Contains built-in XML refactoring operations that pertain to XML fragments with some of the information preconfigured based upon the current context.

Insert XML fragment
Allows you to insert an XML fragment.

Replace element content with XML fragment
Allows you to replace the content of elements with an XML fragment.

Replace element with XML fragment
Allows you to replace elements with an XML fragment.

Review submenu
This submenu includes the following actions:

Track Changes
Enables or disables the Track Changes (on page 3224) support for the current document.

Accept Change(s) and Move to Next
Accepts the Tracked Change (on page 3224) located at the cursor position or all of the changes in a selection and then moves to the next change. If you select a part of a deletion or insertion change, only the selected content is accepted.

Accept All Changes
Accepts all Tracked Changes (on page 3224) in the current document.
Reject Change(s) and Move to Next

Rejects the Tracked Change (on page 3224) located at the cursor position or all of the changes in a selection and then moves to the next change. If you select a part of a deletion or insertion change, only the selected content is rejected.

Reject All Changes

Rejects all Tracked Changes (on page 3224) in the current document.

Comment Change

Opens a dialog box that allows you to add a comment to an existing Tracked Change (on page 3224). The comment will appear in a callout and a tooltip when hovering over the change. If the action is selected on an existing commented change, the dialog box will allow you to edit the comment.

Highlight

Enables the highlighting tool that allows you to mark text in your document.

Colors

Allows you to select the color for highlighting text.

Stop highlighting

Use this action to deactivate the highlighting tool.

Remove highlight(s)

Use this action to remove highlighting from the document.

Add Comment

Inserts a comment at the cursor position. The comment appears in a callout box and a tooltip (when hovering over the change).

Show/Edit Comment

Opens a dialog box that displays the discussion thread and allows the current user to edit comments that do not have replies. If you are not the author who inserted the original comment, the dialog box just displays the comment without the possibility of editing it.

Remove Comment

Removes a selected comment. If you remove a comment that contains replies, all of the replies will also be removed.

Manage Reviews

Opens the Review view (on page 664).

Folding submenu

This submenu includes the following actions:
Toggle Fold
Toggles the state of the current fold.

Collapse Other Folds (Ctrl + NumPad/ (Command + NumPad/ on macOS))
Folds all the elements except the current element.

Collapse Child Folds (Ctrl + NumPad. (Command + NumPad. on macOS))
Folds the elements indented with one level inside the current element.

Expand Child Folds
Unfolds all child elements of the currently selected element.

Expand All (Ctrl + NumPad* (Command + NumPad* on macOS))
Unfolds all elements in the current document.

About Element > Go to Definition
Moves the cursor to the definition of the current element.

Inspect Styles
Opens the CSS Inspector view (on page 640) that allows you to examine the CSS rules that match the currently selected element.

Options
Opens the Author mode preferences page (on page 176) where you can configure various options with regard to the Author editing mode.

XHTML Drag/Drop (or Copy/Paste) Actions
Dragging a file from the Project view (on page 402) and dropping it into an XHTML document that is edited in Author mode, creates a link to the dragged file (the `<a>` element with the `@href` attribute) at the drop location. Copy and paste actions work the same.

You can also drag images or media files from your system explorer or the Project view (on page 402) and drop them into an XHTML document (or copy and paste). This will insert the appropriate element at the drop or paste location (for example, dropping/pasting an image will insert the `<img>` element with the `@src` attribute).

Tip:
For information about customizing Author mode actions for a particular framework (on page 3219) (document type), see the Customizing the Author Mode Editing Experience for a Framework (on page 2145) section.

Related Information:
Customizing the Author Mode Editing Experience for a Framework (on page 2145)
TEI P5 Document Type (Framework)

The TEI (Text Encoding Initiative) document type is an international and interdisciplinary standard that enables libraries, museums, publishers, and individual scholars to represent a variety of literary and linguistic texts for online research, teaching, and preservation.

File Definition

A file is considered to be a TEI P5 document when one of the following conditions are true:

- The document namespace is http://www.tei-c.org/ns/1.0.
- The public ID of the document is -//TEI P5.

Default Document Templates

There are a variety of default TEI P5 templates available when creating new documents from templates (on page 368) and they can be found in: Framework Templates > TEI P5.

The default templates for TEI P5 documents are located in the [OXYGEN_INSTALL_DIR]/frameworks/tei/templates/TEI P5 folder.

Default Schema for Validation and Content Completion

The default schema that is used if one is not detected in the TEI P5 document is tei_all.rng and it is stored in [OXYGEN_INSTALL_DIR]/frameworks/tei/xml/tei/custom/schema/relaxng/.

Default CSS

The default CSS files used for rendering TEI content in Author mode are stored in [OXYGEN_INSTALL_DIR]/frameworks/tei/xml/tei/css/.

By default, these default CSS files are merged with any that are specified in the document. For more information, see Configuring and Managing Multiple CSS Styles for a Framework (on page 2145).

Default XML Catalogs

The default XML Catalogs (on page 3225) for the TEI P5 document type are as follows:

- [OXYGEN_INSTALL_DIR]/frameworks/tei/xml/tei/schema/dtd/catalog.xml
- [OXYGEN_INSTALL_DIR]/frameworks/tei/xml/tei/custom/schema/dtd/catalog.xml
- [OXYGEN_INSTALL_DIR]/frameworks/tei/xml/tei/stylesheet/catalog.xml

Transformation Scenarios

Oxygen XML Editor includes built-in transformation scenarios that allow you to transform TEI P5 documents to a variety of outputs, such as PDF, XHTML, EPUB, DOCX, and ODT. They can be found in the TEI P5 section in the Configure Transformation Scenario(s) dialog box (on page 1548).
Resources

- Oxygen Video Tutorial: Editing TEI Documents in Author Mode
- TEI: P5 Guidelines

Related Information:
- Editing XML Documents in Author Mode (on page 588)
- Editing XML Documents in Text Mode (on page 517)
- Adding Tables in TEI Documents (on page 713)

TEI P5 Author Mode Actions

A variety of actions are available for TEI P5 documents that can be found in **TEI P5** menu, toolbar, contextual menu, and the *Content Completion Assistant (on page 3217).*

**TEI P5 Toolbar Actions**

The following default actions are available on the TEI P5 toolbar when editing in **Author** mode (by default, they are also available in the **TEI P5** menu and some of them are in various submenus of the contextual menu):

- **Bold**
  Changes the style of the selected text to *bold* by surrounding it with the `<hi>` tag and setting the `@rend` attribute to *bold*. You can use this action on multiple non-contiguous selections.

- **Italic**
  Changes the style of the selected text to *italic* by surrounding it with the `<hi>` tag and setting the `@rend` attribute to *italic*. You can use this action on multiple non-contiguous selections.

- **Underline**
  Changes the style of the selected text to *underline* by surrounding it with the `<hi>` tag and setting the `@rend` attribute to *ul*. You can use this action on multiple non-contiguous selections.

- **Insert Section**
  Inserts a new section or subsection, depending on the current context. For example, if the current context is `div1`, then a `div2` is inserted. By default, this action also inserts a paragraph element as a child node.

- **Insert Paragraph**
  Insert a new paragraph element at current cursor position.

- **Insert Image**
  Inserts an image reference (on page 719) at the cursor position. Depending on the current location, an image-type element is inserted.

- **Insert List Item**
Inserts a list item in the current list type.

Insert Ordered List

Inserts an ordered list at the cursor position. A child list item is also automatically inserted by default. You can also use this action to convert selected paragraphs or other types of lists to an ordered list.

Insert Itemized List

Inserts an itemized list at the cursor position. A child list item is also automatically inserted by default. You can also use this action to convert selected paragraphs or other types of lists to an itemized list.

Sort

Sorts cells or list items in a table.

Insert Table

Opens a dialog box that allows you to configure and insert a table. You can generate a header and footer, set the number of rows and columns of the table and decide how the table is framed. You can also use this action to convert selected paragraphs, lists, and inline content (mixed content, text plus markup, that is rendered inside a block element (on page 3216)) into a table, with the selected content inserted in the first column, starting from the first row after the header (if a header is inserted).

Note:

If the selection contains a mixture of elements that cannot be converted, you will receive an error message saying that Only lists, paragraphs, or inline content can be converted to tables.

Insert Row

Inserts a new table row with empty cells below the current row. This action is available when the cursor is positioned inside a table.

Insert Column

Inserts a new table column with empty cells after the current column. This action is available when the cursor is positioned inside a table.

Insert Cell

Inserts a new empty cell depending on the current context. If the cursor is positioned between two cells, Oxygen XML Editor a new cell at the cursor position. If the cursor is inside a cell, the new cell is created after the current cell.

Delete Column(s)

Deletes the table column located at the cursor position or multiple columns in a selection.
Delete Row(s)

Deletes the table row located at the cursor position or multiple rows in a selection.

Join Cells

Joins the content of the selected cells (both horizontally and vertically).

Split Cell

Splits the cell at the cursor location. If Oxygen XML Editor detects more than one option to split the cell, a dialog box will be displayed that allows you to select the number of rows or columns to split the cell into.

TEI Contextual Menu Actions

The following actions are available in the contextual menu when editing in Author mode (most of them are also available in the TEI P5 menu at the top of the interface):

Add File to Review Task

This action can be used to add the current document to a task in the Content Fusion Tasks Manager view. Oxygen Content Fusion is a flexible, intuitive collaboration platform designed to adapt to any type of documentation review workflow. This functionality is available through a pre-installed connector add-on (on page 2527). To fully take advantage of all of the benefits and features of Content Fusion, your organization will need an Oxygen Content Fusion Enterprise Server. For more information, see the Oxygen Content Fusion website.

Edit Attributes

Displays an in-place attributes editor (on page 630) that allows you to manage the attributes of an element.

Edit Profiling Attributes

Allows you to change the profiling attributes (on page 668) defined on all selected elements.

Cut (Ctrl + X (Command + X on macOS))

Removes the currently selected content from the document and places it in the clipboard.

Copy (Ctrl + C (Command + C on macOS))

Places a copy of the currently selected content in the clipboard.

Paste (Ctrl + V (Command + V on macOS))

Inserts the current clipboard content into the document at the cursor position.

Image Map Editor

This action is available in the contextual menu when it is invoked on an image. This action applies an image map to the current image (if one does not already exist) and opens the Image Map Editor dialog box. This feature allows you to create hyperlinks in specific areas of an image that will link to various destinations.
Insert submenu

This submenu includes the following insert actions:

- **Insert Table**
  Opens a dialog box that allows you to configure and insert a table. You can generate a header and footer, set the number of rows and columns of the table and decide how the table is framed. You can also use this action to convert selected paragraphs, lists, and inline content (mixed content, text plus markup, that is rendered inside a block element (on page 3216)) into a table, with the selected content inserted in the first column, starting from the first row after the header (if a header is inserted).

  **Note:**
  If the selection contains a mixture of elements that cannot be converted, you will receive an error message saying that **Only lists, paragraphs, or inline content can be converted to tables.**

- **Insert Image**
  Inserts an image reference (on page 719) at the cursor position. Depending on the current location, an image-type element is inserted.

- **Insert Paragraph**
  Inserts a new paragraph element at current cursor position.

- **Insert Section**
  Inserts a new section element in the document, depending on the current context.

- **Insert Entity**
  Allows you to insert a predefined entity or character entity. Surrogate character entities (range #x10000 to #x10FFFF) are also accepted. Character entities can be entered in one of the following forms:

  - `#<decimal value>` - e.g. #65
  - `&<decimal value>` - e.g. &#65
  - `#x<hexadecimal value>` - e.g. #x41
  - `&#x<hexadecimal value>` - e.g. &#x41

- **Generate IDs**
  Oxygen XML Editor generates unique IDs for the current element (or elements), depending on how the action is invoked:
• When invoked on a single selection, an ID is generated for the selected element at the cursor position.
• When invoked on a block of selected content, IDs are generated for all top-level elements and elements listed in the **ID Options** dialog box that are found in the current selection.

**Note:**

The **Generate IDs** action does not overwrite existing ID values. It only affects elements that do not already have an `@id` attribute.

### Table actions

The following table editing actions are available in the contextual menu when it is invoked on a table:

**Insert Rows**

Opens a dialog box that allows you to insert any number of rows and specify the position where they will be inserted (**Above** or **Below** the current row).

**Delete Row(s)**

Deletes the table row located at the cursor position or multiple rows in a selection.

**Insert Columns**

Opens a dialog box that allows you to insert any number of columns and specify the position where they will be inserted (**Above** or **Below** the current column).

**Delete Column(s)**

Deletes the table column located at the cursor position or multiple columns in a selection.

**Join Cells**

Joins the content of the selected cells (both horizontally and vertically).

**Split Cell**

Splits the cell at the cursor location. If Oxygen XML Editor detects more than one option to split the cell, a dialog box will be displayed that allows you to select the number of rows or columns to split the cell into.

**Sort**

Sorts cells or list items in a table.

**Table Properties**

Opens the **Table properties** dialog box that allows you to configure properties of a table (such as frame borders).

### Other Actions submenu
This submenu gives you access to all the usual contextual menu actions.

Select submenu

This submenu allows you to select the following:

Element
- Selects the entire element at the current cursor position.

Content
- Selects the entire content of the element at the current cursor position, excluding the start and end tag. Performing this action repeatedly will result in the selection of the content of the ancestor of the currently selected element content.

Parent
- Selects the entire parent element at the current cursor position.

Text submenu

This submenu contains the following actions:

To Lower Case
- Converts the selected content to lower case characters.

To Upper Case
- Converts the selected content to upper case characters.

Capitalize Sentences
- Converts to upper case the first character of every selected sentence.

Capitalize Words
- Converts to upper case the first character of every selected word.

Count Words
- Counts the number of words and characters (no spaces) in the entire document or in the selection for regular content and read-only content.

Note:
The content marked as deleted with change tracking (on page 3224) is ignored when counting words.

Convert Hexadecimal Sequence to Character (Ctrl + Shift + X (Command + Shift + X on macOS))

Converts a sequence of hexadecimal characters to the corresponding Unicode character (on page 463). The action can be invoked if there is a selection containing a valid hexadecimal sequence or if the cursor is placed at the right side of a valid hexadecimal sequence. A valid hexadecimal sequence can be composed
of 2 to 4 hexadecimal characters and may or may not be preceded by the 0x or 0X prefix. Examples of valid sequences and the characters they will be converted to:

- 0x0045 will be converted to Æ
- 0x0125 to ĥ
- 265 to ť
- 2190 to ←

Note:
For more information about finding the hexadecimal value of a character, see Finding the Decimal, Hexadecimal, or Character Entity Equivalent (on page 466).

Refactoring submenu

Contains a series of actions designed to alter the XML structure of the document:

- **Toggle Comment**
  Encloses the currently selected text in an XML comment, or removes the comment if it is commented.

- **Move Up (Alt + UpArrow)**
  Moves the current node or selected nodes in front of the previous node.

- **Move Down (Alt + DownArrow)**
  Moves the current node or selected nodes after the subsequent node.

- **Split Element (Alt + Shift + D (Ctrl + Alt + D on macOS))**
  Splits the content of the closest element that contains the position of the cursor. Thus, if the cursor is positioned at the beginning or at the end of the element, the newly created sibling will be empty.

- **Join Elements**
  Joins two adjacent block elements (on page 3216) that have the same name. The action is available only when the cursor position is between the two adjacent block elements. Also, joining two block elements can be done by pressing the Delete or Backspace keys and the cursor is positioned between the boundaries of these two elements.

- **Surround with Tags (Ctrl + E (Command + E on macOS))**
  Allows you to choose a tag to enclose a selected portion of content. If there is no selection, the start and end tags are inserted at the cursor position.
• If the **Position cursor between tags** option *(on page 212)* is selected in the **Content Completion** preferences page, the cursor is placed between the start and end tag.

• If the **Position cursor between tags** option *(on page 212)* is not selected in the **Content Completion** preferences page, the cursor is placed at the end of the start tag, in an insert-attribute position.

**Surround with [tag]** *(Ctrl + ForwardSlash (Command + ForwardSlash on macOS))*

Surround the selected content with the last tag used.

**Rename Element**

The element from the cursor position, and any elements with the same name, can be renamed according with the options from the **Rename** dialog box.

**Delete Element Tags**

Deletes the tags of the closest element that contains the position of the cursor. This operation is also executed if the start or end tags of an element are deleted by pressing the **Delete** or **Backspace** keys.

**Remove All Markup**

Removes all the XML markup inside the selected block of content and keeps only the text content.

**Remove Text**

Removes the text content of the selected block of content and keeps the markup intact with empty elements.

**Attributes Refactoring Actions**

Contains built-in XML refactoring operations that pertain to attributes with some of the information preconfigured based upon the current context.

**Add/Change attribute**

Allows you to change the value of an attribute or insert a new one.

**Convert attribute to element**

Allows you to change an attribute into an element.

**Delete attribute**

Allows you to remove one or more attributes.

**Rename attribute**

Allows you to rename an attribute.

**Replace in attribute value**
Allows you to search for a text fragment inside an attribute value and change the fragment to a new value.

**Comments Refactoring Actions**

Contains built-in XML refactoring operations that pertain to comments with some of the information preconfigured based upon the current context.

**Delete comments**

Allows you to delete comments found inside one or more elements.

**Elements Refactoring Actions**

Contains built-in XML refactoring operations that pertain to elements with some of the information preconfigured based upon the current context.

**Delete element**

Allows you to delete elements.

**Delete element content**

Allows you to delete the content of elements.

**Insert element**

Allows you to insert new elements.

**Rename element**

Allows you to rename elements.

**Unwrap element**

Allows you to remove the surrounding tags of elements, while keeping the content unchanged.

**Wrap element**

Allows you to surround elements with element tags.

**Wrap element content**

Allows you to surround the content of elements with element tags.

**Fragments Refactoring Actions**

Contains built-in XML refactoring operations that pertain to XML fragments with some of the information preconfigured based upon the current context.

**Insert XML fragment**

Allows you to insert an XML fragment.

**Replace element content with XML fragment**

Allows you to replace the content of elements with an XML fragment.

**Replace element with XML fragment**
Allows you to replace elements with an XML fragment.

**Review submenu**

This submenu includes the following actions:

- **Track Changes**
  Enables or disables the *Track Changes* support for the current document.

- **Accept Change(s) and Move to Next**
  Accepts the *Tracked Change* located at the cursor position or all of the changes in a selection and then moves to the next change. If you select a part of a *deletion or insertion* change, only the selected content is accepted.

- **Accept All Changes**
  Accepts all *Tracked Changes* in the current document.

- **Reject Change(s) and Move to Next**
  Rejects the *Tracked Change* located at the cursor position or all of the changes in a selection and then moves to the next change. If you select a part of a *deletion or insertion* change, only the selected content is rejected.

- **Reject All Changes**
  Rejects all *Tracked Changes* in the current document.

- **Comment Change**
  Opens a dialog box that allows you to add a comment to an existing *Tracked Change*. The comment will appear in a callout and a tooltip when hovering over the change. If the action is selected on an existing commented change, the dialog box will allow you to edit the comment.

- **Highlight**
  Enables the highlighting tool that allows you to mark text in your document.

**Colors**

Allows you to select the color for highlighting text.

**Stop highlighting**

Use this action to deactivate the highlighting tool.

**Remove highlight(s)**

Use this action to remove highlighting from the document.

- **Add Comment**
  Inserts a comment at the cursor position. The comment appears in a callout box and a tooltip (when hovering over the change).
Show/Edit Comment

Opens a dialog box that displays the discussion thread and allows the current user to edit comments that do not have replies. If you are not the author who inserted the original comment, the dialog box just displays the comment without the possibility of editing it.

Remove Comment

Removes a selected comment. If you remove a comment that contains replies, all of the replies will also be removed.

Manage Reviews

Opens the Review view (on page 664).

Folding submenu

This submenu includes the following actions:

- Toggle Fold
  Toggles the state of the current fold.

- Collapse Other Folds (Ctrl + NumPad/ (Command + NumPad/ on macOS))
  Folds all the elements except the current element.

- Collapse Child Folds (Ctrl + NumPad. (Command + NumPad. on macOS))
  Folds the elements indented with one level inside the current element.

- Expand Child Folds
  Unfolds all child elements of the currently selected element.

- Expand All (Ctrl + NumPad* (Command + NumPad* on macOS))
  Unfolds all elements in the current document.

About Element > Go to Definition

Moves the cursor to the definition of the current element.

Inspect Styles

Opens the CSS Inspector view (on page 640) that allows you to examine the CSS rules that match the currently selected element.

Options

Opens the Author mode preferences page (on page 176) where you can configure various options with regard to the Author editing mode.
TEI P5 Drag/Drop Actions

Dragging a file from the Project view (on page 402) and dropping it into a TEI P5 document that is edited in Author mode, creates a link to the dragged file (the `<ptr>` element with the `@target` attribute) at the drop location. Dragging an image file from the default file system application (Windows Explorer on Windows or Finder on macOS, for example) and dropping it into a TEI P5 document inserts a graphic element (the `<graphic>` element with the `@url` attribute) at the drop location, similar to the Insert Image toolbar action.

Tip:
For information about customizing Author mode actions for a particular framework (on page 3219) (document type), see the Customizing the Author Mode Editing Experience for a Framework (on page 2145) section.

Related Information:
Customizing the Author Mode Editing Experience for a Framework (on page 2145)

Customization of TEI Frameworks Using the Latest Sources

The TEI P5 (on page 1388) framework (on page 3219) is available as a public project at the following SVN repository: https://github.com/TEIC/oxygen-tei. This project is the base for customizing a TEI framework.

To customize a TEI framework, follow this procedure:

1. Check out the project on a local computer from the SVN repository.
   This action is done with an SVN client application that creates a working copy of the SVN repository on a local computer.
2. Customize the TEI framework in Oxygen XML Editor.
   a. Set the Oxygen XML Editor frameworks folder to the oxygen/frameworks subfolder of the folder of the SVN working copy.
      Open the Preferences dialog box (Options > Preferences) (on page 124), go to Global, and set the path of the SVN working copy in the Use custom frameworks option.
   b. Open the Preferences dialog box (Options > Preferences) (on page 124), go to Document Type Association > Locations, and select Custom.
3. Build a JAR (on page 3220) file with the TEI framework.
   The SVN project includes a build.xml file that can be used for building a JAR file using the Ant tool. The command that should be used:
   ```
   ant -f build.xml
   ```
4. Distribute the JAR file to the users that need the customized TEI framework.
   The command from the above step creates a file `tei.zip` in the `dist` subfolder of the SVN project. Each user that needs the customized TEI framework will receive the `tei.zip` file and will unzip it in the `frameworks` folder of the Oxygen XML Editor install folder.

**Customization of TEI Frameworks Using the Compiled Sources**

The following procedure describes how to update to the latest stable version of TEI Schema and TEI XSL, already integrated in the TEI framework (on page 3219) for Oxygen XML Editor.

2. Go to Downloads;
3. Download the latest uploaded `.zip` file;
4. Unpack the `.zip` file and copy its content in the Oxygen XML Editor frameworks folder.

**TEI ODD Document Type (Framework)**

The TEI ODD (Text Encoding Initiative - One Document Does it all) document type is a TEI XML-conformant specification format that allows you to create a custom TEI P5 schema in a literate programming fashion. A system of XSLT stylesheets called Roma was created by the TEI Consortium for manipulating the ODD files.

**File Definition**

A file is considered to be a TEI ODD document when the following conditions are true:

- The file extension is `.odd`.
- The document namespace is `http://www.tei-c.org/ns/1.0`.

**Default Document Templates**

There is a default TEI ODD document template available when creating new documents from templates (on page 368) and they can be found in: Framework Templates > TEI ODD.

The default template is located in the `[OXYGEN_INSTALL_DIR]/frameworks/tei/templates/TEI ODD` folder.

**Default Schema for Validation and Content Completion**

The default schema that is used if one is not detected in the TEI ODD document is `tei_odds.rng` and it is stored in `[OXYGEN_INSTALL_DIR]/frameworks/tei/xml/tei/custom/schema/relaxng/`.

**Default CSS**

The default CSS files used for rendering TEI ODD content in Author mode are stored in `[OXYGEN_INSTALL_DIR]/frameworks/tei/xml/tei/css/`. 
By default, these default CSS files are merged with any that are specified in the document. For more information, see Configuring and Managing Multiple CSS Styles for a Framework (on page 2145).

**Default XML Catalogs**

The default XML Catalogs (on page 3225) for the TEI ODD document type are as follows:

- `[OXYGEN_INSTALL_DIR]/frameworks/tei/xml/tei/custom/schema/catalog.xml`
- `[OXYGEN_INSTALL_DIR]/frameworks/tei/xml/tei/schema/catalog.xml`

**Transformation Scenarios**

Oxygen XML Editor includes built-in transformation scenarios that allow you to transform TEI ODD documents to a variety of outputs, such as PDF, XHTML, EPUB, DOCX, ODT, RNG, DTD, and XML Schema. They can be found in the TEI ODD section in the Configure Transformation Scenario(s) dialog box (on page 1548).

**Resources**

- Oxygen Video Tutorial: Editing TEI Documents in Author Mode
- TEI: Getting Started with ODD

**Related Information:**

Editing XML Documents in Author Mode (on page 588)
Editing XML Documents in Text Mode (on page 517)

**TEI ODD Author Mode Actions**

A variety of actions are available for TEI ODD documents that can be found in TEI ODD menu, toolbar, contextual menu, and the Content Completion Assistant (on page 3217).

**TEI ODD Toolbar Actions**

The following default actions are available on the TEI ODD toolbar when editing in Author mode (by default, they are also available in the TEI ODD menu and some of them are in various submenus of the contextual menu):

- **Bold**
  Changes the style of the selected text to bold by surrounding it with the `<hi>` tag and setting the `@rend` attribute to `bold`. You can use this action on multiple non-contiguous selections.

- **Italic**
  Changes the style of the selected text to italic by surrounding it with the `<hi>` tag and setting the `@rend` attribute to `italic`. You can use this action on multiple non-contiguous selections.

- **Underline**
Changes the style of the selected text to **underline** by surrounding it with the `<hi>` tag and setting the `rend` attribute to `ul`. You can use this action on multiple non-contiguous selections.

### Insert Section

Inserts a new section or subsection, depending on the current context. For example, if the current context is `div1`, then a `div2` is inserted. By default, this action also inserts a paragraph element as a child node.

### Insert Paragraph

Insert a new paragraph element at current cursor position.

### Insert Image

Inserts an image reference *(on page 719)* at the cursor position. Depending on the current location, an image-type element is inserted.

### Insert List Item

Inserts a list item in the current list type.

### Insert Ordered List

Inserts an ordered list at the cursor position. A child list item is also automatically inserted by default. You can also use this action to convert selected paragraphs or other types of lists to an ordered list.

### Insert Itemized List

Inserts an itemized list at the cursor position. A child list item is also automatically inserted by default. You can also use this action to convert selected paragraphs or other types of lists to an itemized list.

### Insert Table

Opens a dialog box that allows you to configure and insert a table. You can generate a header and footer, set the number of rows and columns of the table and decide how the table is framed. You can also use this action to convert selected paragraphs, lists, and inline content (mixed content, text plus markup, that is rendered inside a block element *(on page 3216)*) into a table, with the selected content inserted in the first column, starting from the first row after the header (if a header is inserted).

#### Note:

If the selection contains a mixture of elements that cannot be converted, you will receive an error message saying that **Only lists, paragraphs, or inline content can be converted to tables.**

### Insert Row

Inserts a new table row with empty cells below the current row. This action is available when the cursor is positioned inside a table.
Insert Column

Inserts a new table column with empty cells after the current column. This action is available when the cursor is positioned inside a table.

Insert Cell

Inserts a new empty cell depending on the current context. If the cursor is positioned between two cells, Oxygen XML Editor a new cell at the cursor position. If the cursor is inside a cell, the new cell is created after the current cell.

Delete Column(s)

Deletes the table column located at the cursor position or multiple columns in a selection.

Delete Row(s)

Deletes the table row located at the cursor position or multiple rows in a selection.

Join Cells

Joins the content of the selected cells (both horizontally and vertically).

Split Cell

Splits the cell at the cursor location. If Oxygen XML Editor detects more than one option to split the cell, a dialog box will be displayed that allows you to select the number of rows or columns to split the cell into.

**TEI Contextual Menu Actions**

The following actions are available in the contextual menu when editing in Author mode (most of them are also available in the TEI ODD menu at the top of the interface):

Add File to Review Task

This action can be used to add the current document to a task in the Content Fusion Tasks Manager view. Oxygen Content Fusion is a flexible, intuitive collaboration platform designed to adapt to any type of documentation review workflow. This functionality is available through a pre-installed connector add-on (on page 2527). To fully take advantage of all of the benefits and features of Content Fusion, your organization will need an Oxygen Content Fusion Enterprise Server. For more information, see the Oxygen Content Fusion website.

Edit Attributes

Displays an in-place attributes editor (on page 630) that allows you to manage the attributes of an element.

Edit Profiling Attributes

Allows you to change the profiling attributes (on page 668) defined on all selected elements.

Cut (Ctrl + X (Command + X on macOS))

Removes the currently selected content from the document and places it in the clipboard.
Copy (Ctrl + C (Command + C on macOS))

Places a copy of the currently selected content in the clipboard.

Paste (Ctrl + V (Command + V on macOS))

Inserts the current clipboard content into the document at the cursor position.

Image Map Editor

This action is available in the contextual menu when it is invoked on an image. This action applies an image map to the current image (if one does not already exist) and opens the Image Map Editor dialog box. This feature allows you to create hyperlinks in specific areas of an image that will link to various destinations.

Insert submenu

This submenu includes the following insert actions that are specific to the DocBook framework:

Insert Table

Opens a dialog box that allows you to configure and insert a table. You can generate a header and footer, set the number of rows and columns of the table and decide how the table is framed. You can also use this action to convert selected paragraphs, lists, and inline content (mixed content, text plus markup, that is rendered inside a block element (on page 3216)) into a table, with the selected content inserted in the first column, starting from the first row after the header (if a header is inserted).

Note:

If the selection contains a mixture of elements that cannot be converted, you will receive an error message saying that Only lists, paragraphs, or inline content can be converted to tables.

Insert Image

Inserts an image reference (on page 719) at the cursor position. Depending on the current location, an image-type element is inserted.

Insert Paragraph

Inserts a new paragraph element at current cursor position.

Insert Section

Inserts a new section element in the document, depending on the current context.

Insert Entity

Allows you to insert a predefined entity or character entity. Surrogate character entities (range #x10000 to #x10FFFF) are also accepted. Character entities can be entered in one of the following forms:
• \#<decimal value>- e.g. \#65
• &\#<decimal value>- e.g. \&\#65
• \#x<hexadecimal value>- e.g. \#x41
• &\#x<hexadecimal value>- e.g. \&\#x41

**Generate IDs**

Oxygen XML Editor generates unique IDs for the current element (or elements), depending on how the action is invoked:

• When invoked on a single selection, an ID is generated for the selected element at the cursor position.
• When invoked on a block of selected content, IDs are generated for all top-level elements and elements listed in the **ID Options** dialog box that are found in the current selection.

**Note:**
The **Generate IDs** action does not overwrite existing ID values. It only affects elements that do not already have an `@id` attribute.

**Table actions**

The following table editing actions are available in the contextual menu when it is invoked on a table:

**Insert Rows**

Opens a dialog box that allows you to insert any number of rows and specify the position where they will be inserted (Above or Below the current row).

**Delete Row(s)**

Deletes the table row located at the cursor position or multiple rows in a selection.

**Insert Columns**

Opens a dialog box that allows you to insert any number of columns and specify the position where they will be inserted (Above or Below the current column).

**Delete Column(s)**

Deletes the table column located at the cursor position or multiple columns in a selection.

**Join Cells**

Joins the content of the selected cells (both horizontally and vertically).

**Split Cell**
Splits the cell at the cursor location. If Oxygen XML Editor detects more than one option to split the cell, a dialog box will be displayed that allows you to select the number of rows or columns to split the cell into.

**Sort**

Sorts cells or list items in a table.

**Table Properties**

Opens the Table properties dialog box that allows you to configure properties of a table (such as frame borders).

**Other Actions submenu**

This submenu gives you access to all the usual contextual menu actions.

**Select submenu**

This submenu allows you to select the following:

**Element**

Selects the entire element at the current cursor position.

**Content**

Selects the entire content of the element at the current cursor position, excluding the start and end tag. Performing this action repeatedly will result in the selection of the content of the ancestor of the currently selected element content.

**Parent**

Selects the entire parent element at the current cursor position.

**Text submenu**

This submenu contains the following actions:

**To Lower Case**

Converts the selected content to lower case characters.

**To Upper Case**

Converts the selected content to upper case characters.

**Capitalize Sentences**

Converts to upper case the first character of every selected sentence.

**Capitalize Words**

Converts to upper case the first character of every selected word.

**Count Words**

Counts the number of words and characters (no spaces) in the entire document or in the selection for regular content and read-only content.
Convert Hexadecimal Sequence to Character (Ctrl + Shift + X (Command + Shift + X on macOS))

Converts a sequence of hexadecimal characters to the corresponding Unicode character (on page 463). The action can be invoked if there is a selection containing a valid hexadecimal sequence or if the cursor is placed at the right side of a valid hexadecimal sequence. A valid hexadecimal sequence can be composed of 2 to 4 hexadecimal characters and may or may not be preceded by the 0x or 0X prefix. Examples of valid sequences and the characters they will be converted to:

- \x0045 will be converted to E
- \x0125 to ĥ
- 265 to ɥ
- 2190 to ←

Note: For more information about finding the hexadecimal value of a character, see Finding the Decimal, Hexadecimal, or Character Entity Equivalent (on page 466).

Refactoring submenu

Contains a series of actions designed to alter the XML structure of the document:

- **Toggle Comment**
  
  Encloses the currently selected text in an XML comment, or removes the comment if it is commented.

- **Move Up (Alt + UpArrow)**
  
  Moves the current node or selected nodes in front of the previous node.

- **Move Down (Alt + DownArrow)**
  
  Moves the current node or selected nodes after the subsequent node.

- **Split Element (Alt + Shift + D (Ctrl + Alt + D on macOS))**
  
  Splits the content of the closest element that contains the position of the cursor. Thus, if the cursor is positioned at the beginning or at the end of the element, the newly created sibling will be empty.

- **Join Elements**
Joins two adjacent block elements (on page 3216) that have the same name. The action is available only when the cursor position is between the two adjacent block elements. Also, joining two block elements can be done by pressing the Delete or Backspace keys and the cursor is positioned between the boundaries of these two elements.

**Surround with Tags (Ctrl + E (Command + E on macOS))**

Allows you to choose a tag to enclose a selected portion of content. If there is no selection, the start and end tags are inserted at the cursor position.

- If the Position cursor between tags option (on page 212) is selected in the Content Completion preferences page, the cursor is placed between the start and end tag.
- If the Position cursor between tags option (on page 212) is not selected in the Content Completion preferences page, the cursor is placed at the end of the start tag, in an insert-attribute position.

**Surround with ‘[tag]’ (Ctrl + ForwardSlash (Command + ForwardSlash on macOS))**

Surround the selected content with the last tag used.

**Rename Element**

The element from the cursor position, and any elements with the same name, can be renamed according with the options from the Rename dialog box.

**Delete Element Tags**

Deletes the tags of the closest element that contains the position of the cursor. This operation is also executed if the start or end tags of an element are deleted by pressing the Delete or Backspace keys.

**Remove All Markup**

Removes all the XML markup inside the selected block of content and keeps only the text content.

**Remove Text**

Removes the text content of the selected block of content and keeps the markup intact with empty elements.

**Attributes Refactoring Actions**

Contains built-in XML refactoring operations that pertain to attributes with some of the information preconfigured based upon the current context.

**Add/Change attribute**

Allows you to change the value of an attribute or insert a new one.

**Convert attribute to element**
Allows you to change an attribute into an element.

Delete attribute
Allows you to remove one or more attributes.

Rename attribute
Allows you to rename an attribute.

Replace in attribute value
Allows you to search for a text fragment inside an attribute value and change the fragment to a new value.

Comments Refactoring Actions
Contains built-in XML refactoring operations that pertain to comments with some of the information preconfigured based upon the current context.

Delete comments
Allows you to delete comments found inside one or more elements.

Elements Refactoring Actions
Contains built-in XML refactoring operations that pertain to elements with some of the information preconfigured based upon the current context.

Delete element
Allows you to delete elements.

Delete element content
Allows you to delete the content of elements.

Insert element
Allows you to insert new elements.

Rename element
Allows you to rename elements.

Unwrap element
Allows you to remove the surrounding tags of elements, while keeping the content unchanged.

Wrap element
Allows you to surround elements with element tags.

Wrap element content
Allows you to surround the content of elements with element tags.

Fragments Refactoring Actions
Contains built-in XML refactoring operations that pertain to XML fragments with some of the information preconfigured based upon the current context.

**Insert XML fragment**

Allows you to insert an XML fragment.

**Replace element content with XML fragment**

Allows you to replace the content of elements with an XML fragment.

**Replace element with XML fragment**

Allows you to replace elements with an XML fragment.

**Review submenu**

This submenu includes the following actions:

- **Track Changes**
  
  Enables or disables the Track Changes (on page 3224) support for the current document.

- **Accept Change(s) and Move to Next**
  
  Accepts the Tracked Change (on page 3224) located at the cursor position or all of the changes in a selection and then moves to the next change. If you select a part of a deletion or insertion change, only the selected content is accepted.

- **Accept All Changes**
  
  Accepts all Tracked Changes (on page 3224) in the current document.

- **Reject Change(s) and Move to Next**
  
  Rejects the Tracked Change (on page 3224) located at the cursor position or all of the changes in a selection and then moves to the next change. If you select a part of a deletion or insertion change, only the selected content is rejected.

- **Reject All Changes**
  
  Rejects all Tracked Changes (on page 3224) in the current document.

- **Comment Change**
  
  Opens a dialog box that allows you to add a comment to an existing Tracked Change (on page 3224). The comment will appear in a callout and a tooltip when hovering over the change. If the action is selected on an existing commented change, the dialog box will allow you to edit the comment.

- **Highlight**
  
  Enables the highlighting tool that allows you to mark text in your document.

**Colors**

Allows you to select the color for highlighting text.
Stop highlighting

Use this action to deactivate the highlighting tool.

Remove highlight(s)

Use this action to remove highlighting from the document.

Add Comment

Inserts a comment at the cursor position. The comment appears in a callout box and a tooltip (when hovering over the change).

Show/Edit Comment

Opens a dialog box that displays the discussion thread and allows the current user to edit comments that do not have replies. If you are not the author who inserted the original comment, the dialog box just displays the comment without the possibility of editing it.

Remove Comment

Removes a selected comment. If you remove a comment that contains replies, all of the replies will also be removed.

Manage Reviews

Opens the Review view (on page 664).

Folding submenu

This submenu includes the following actions:

Toggle Fold

Toggles the state of the current fold.

Collapse Other Folds (Ctrl + NumPad/ (Command + NumPad/ on macOS))

Folds all the elements except the current element.

Collapse Child Folds (Ctrl + NumPad. (Command + NumPad. on macOS))

Folds the elements indented with one level inside the current element.

Expand Child Folds

Unfolds all child elements of the currently selected element.

Expand All (Ctrl + NumPad* (Command + NumPad* on macOS))

Unfolds all elements in the current document.

About Element > Go to Definition

Moves the cursor to the definition of the current element.

Inspect Styles
Opens the CSS Inspector view (on page 640) that allows you to examine the CSS rules that match the currently selected element.

**Options**

Opens the Author mode preferences page (on page 176) where you can configure various options with regard to the Author editing mode.

**TEI ODD Drag/Drop Actions**

Dragging a file from the Project view (on page 402) and dropping it into a TEI ODD document that is edited in Author mode, creates a link to the dragged file (the `ptr` element with the `@target` attribute) at the drop location.

**Tip:**

For information about customizing Author mode actions for a particular framework (on page 3219) (document type), see the Customizing the Author Mode Editing Experience for a Framework (on page 2145) section.

**jTEI Document Type (Framework)**

The jTEI (Journal of the Text Encoding Initiative) document type is a highly restrictive customization (only about 80 elements are included) of the TEI P5 framework.

**File Definition**

A file is considered to be a jTEI document when the root element is named `TEI`, it is in the namespace `http://www.tei-c.org/ns/1.0`, and the `@rend` attribute is set to "jTEI".

**Default Document Templates**

There is a default jTEI Article template available when creating new documents from templates (on page 368) and they can be found in: Framework Templates > TEI JTEI.

The default template is located in the `\[OXYGEN_INSTALL_DIR\]/frameworks/tei/templates/TEI jTEI` folder.

**Default Schema for Validation and Content Completion**

The default schema that is used if one is not detected is `tei_jtei.rng` and it is stored in `\[OXYGEN_INSTALL_DIR\]/frameworks/tei/xml/tei/custom/schema/relaxng/`. 
Default CSS

The default CSS file (`jtei.css`) that is used for rendering jTEI in Author mode is stored in `{OXYGEN_INSTALL_DIR}/frameworks/tei/xml/tei/css/`.

By default, these default CSS files are merged with any that are specified in the document. For more information, see Configuring and Managing Multiple CSS Styles for a Framework (on page 2145).

Default XML Catalogs

The default XML Catalogs (on page 3225) for jTEI are as follows:

- `{OXYGEN_INSTALL_DIR}/frameworks/tei/xml/tei/schema/dtd/catalog.xml`
- `{OXYGEN_INSTALL_DIR}/frameworks/tei/xml/tei/custom/schema/dtd/catalog.xml`
- `{OXYGEN_INSTALL_DIR}/frameworks/tei/xml/tei/stylesheet/catalog.xml`

Transformation Scenarios

Oxygen XML Editor includes built-in transformation scenarios that allow you to transform jTEI documents to PDF and ODT. They can be found in the TEI JTEI section in the Configure Transformation Scenario(s) dialog box (on page 1548).

Resources

- Oxygen Video Tutorial: Editing TEI Documents in Author Mode
- jTEI Article Guidelines

Related Information:

Editing XML Documents in Author Mode (on page 588)
Editing XML Documents in Text Mode (on page 517)
Adding Tables in TEI Documents (on page 713)

JATS Document Type (Framework)

The JATS (NISO Journal Article Tag Suite) document type is a technical standard that defines an XML format for scientific literature.

File Definition

A file is considered to be a JATS document when the PUBLIC ID of the document contains the string `//NLM//`.

Default Document Templates

There are some default JATS templates available when creating new documents from templates (on page 368) and they can be found in: Framework Templates > JATKit - NISO JATS and NLM BITS
The default templates for JATS documents are located in the 
\[OXYGEN_INSTALL_DIR\]/frameworks/jats/templates/ folder.

**Default Schema for Validation and Content Completion**

Default schemas that are used if one is not detected in the JATS document are stored in 
\[OXYGEN_INSTALL_DIR\]/frameworks/jats/lib/schemas/.

**Default CSS**

The default CSS files used for rendering JATS content in Author mode are stored in 
\[OXYGEN_INSTALL_DIR\]/frameworks/jats/lib/author-css/.

By default, these default CSS files are merged with any that are specified in the document. For more information, see Configuring and Managing Multiple CSS Styles for a Framework (on page 2145).

**Default XML Catalog**

The default XML Catalog (on page 3225), jatskit-catalog.xml, is stored in 
\[OXYGEN_INSTALL_DIR\]/frameworks/jats/lib/schemas/.

**Transformation Scenarios**

Oxygen XML Editor includes built-in transformation scenarios that allow you to transform JATS documents to a variety of outputs, such as PDF, HTML, and EPUB. They can be found in the JATSKit section in the Configure Transformation Scenario(s) dialog box (on page 1548).

**Resources**

- Oxygen Video Tutorial: Configuring a JATS Framework
- NLM Journal Archiving and Interchange Tag Suite

**Related Information:**

- Editing XML Documents in Author Mode (on page 588)
- Editing XML Documents in Text Mode (on page 517)

**JATS Author Mode Actions**

A variety of actions are available for JATS documents that can be found in JATS menu, toolbar, contextual menu, and the **Content Completion Assistant (on page 3217)**.

**JATS Toolbar Actions**

The following default actions are available on the JATS toolbar when editing in Author mode (by default, they are also available in the JATS menu and in various submenus of the contextual menu):

- Paragraph Level Drop-Down Menu
Insert Paragraph

Insert a new paragraph element at current cursor position.

Insert Unordered List

Inserts an unordered list at the cursor position. A child list item is also automatically inserted by default. You can also use this action to convert selected paragraphs or other types of lists to an unordered list.

Insert Ordered List

Inserts an ordered list at the cursor position. A child list item is also automatically inserted by default. You can also use this action to convert selected paragraphs or other types of lists to an ordered list.

Boxed Text

Inserts or wraps content in a box with a shaded background.

Code

Inserts or wraps content in a `<code>` element.

Display Quote

Inserts or wraps content in a `<disp-quote>` element.

Figure

Inserts a `<fig>` element with a `<title>` (inside a `<caption>` element). This action opens a dialog box that allows you to enter the text for the title for the figure.

Graphic Figure

Inserts a `<fig>` element with a `<title>` (inside a `<caption>` element), and a `<graphic>` element. A dialog box is displayed that allows you to enter the title for the figure, followed by a dialog box that allows you to select the URL of the graphic to be inserted.

Bold

Surrounds the selected text with a `<bold>` tag. You can use this action on multiple non-contiguous selections.

Italic

Surrounds the selected text with an `<italic>` tag. You can use this action on multiple non-contiguous selections.

Underline

Surrounds the selected text with an `<underline>` tag. You can use this action on multiple non-contiguous selections.

Monospace
Inserts or wraps content with a `<monospace>` element.

## Insert Table
Opens a dialog box that allows you to configure and insert a table. You can generate a header and footer, set the number of rows and columns of the table and decide how the table is framed. You can also use this action to convert selected paragraphs, lists, and inline content (mixed content, text plus markup, that is rendered inside a block element (on page 3216)) into a table, with the selected content inserted in the first column, starting from the first row after the header (if a header is inserted).

**Note:**
If the selection contains a mixture of elements that cannot be converted, you will receive an error message saying that Only lists, paragraphs, or inline content can be converted to tables.

## Insert Row
Inserts a new table row with empty cells below the current row. This action is available when the cursor is positioned inside a table.

## Delete Row(s)
Deletes the table row located at the cursor position or multiple rows in a selection.

## Insert Column
Inserts a new table column with empty cells after the current column. This action is available when the cursor is positioned inside a table.

## Delete Column(s)
Deletes the table column located at the cursor position or multiple columns in a selection.

## Join Cells
Joins the content of the selected cells (both horizontally and vertically).

## Split Cell
Splits the cell at the cursor location. If Oxygen XML Editor detects more than one option to split the cell, a dialog box will be displayed that allows you to select the number of rows or columns to split the cell into.

## Insert Image
Inserts an image reference (on page 719) at the cursor position. Depending on the current location, an image-type element is inserted.

## Insert List Item
Inserts a list item in the current list type.

## Insert MathML
Opens the XML Fragment Editor that allows you to insert and edit MathML notations (on page 749).

Subscript

Surrounds the selected text with a subscript tag, used for inserting a character (number, letter, or symbol) that will appear slightly below the baseline and slightly smaller than the rest of the text.

Superscript

Surrounds the selected text with a superscript tag, used for inserting a character (number, letter, or symbol) that will appear slightly above the baseline and slightly smaller than the rest of the text.

JATS Contextual Menu Actions

The following actions are available in the contextual menu when editing in Author mode (most of them are also available in the JATS menu at the top of the interface):

Add File to Review Task

This action can be used to add the current document to a task in the Content Fusion Tasks Manager view. Oxygen Content Fusion is a flexible, intuitive collaboration platform designed to adapt to any type of documentation review workflow. This functionality is available through a pre-installed connector add-on (on page 2527). To fully take advantage of all of the benefits and features of Content Fusion, your organization will need an Oxygen Content Fusion Enterprise Server. For more information, see the Oxygen Content Fusion website.

Edit Attributes

Displays an in-place attributes editor (on page 630) that allows you to manage the attributes of an element.

Edit Profiling Attributes

Allows you to change the profiling attributes (on page 668) defined on all selected elements.

Cut (Ctrl + X (Command + X on macOS))

Removes the currently selected content from the document and places it in the clipboard.

Copy (Ctrl + C (Command + C on macOS))

Places a copy of the currently selected content in the clipboard.

Paste (Ctrl + V (Command + V on macOS))

Inserts the current clipboard content into the document at the cursor position.

Image Map Editor

This action is available in the contextual menu when it is invoked on an image. This action applies an image map to the current image (if one does not already exist) and opens the Image Map Editor dialog box. This feature allows you to create hyperlinks in specific areas of an image that will link to various destinations.
**Boxed Text**

Inserts or wraps content in a box with a shaded background.

**Insert submenu**

This submenu includes the following insert actions:

**Figure**

Inserts a `<fig>` element with a `<title>` (inside a `<caption>` element). This action opens a dialog box that allows you to enter the text for the title for the figure.

**Graphic Figure**

Inserts a `<fig>` element with a `<title>` (inside a `<caption>` element), and a `<graphic>` element. A dialog box is displayed that allows you to enter the title for the figure, followed by a dialog box that allows you to select the URL of the graphic to be inserted.

**Display Quote**

Inserts or wraps content in a `<disp-quote>` element.

**Insert Paragraph**

Inserts a new `paragraph` element at current cursor position.

**Insert Image**

Inserts an image reference (on page 719) at the cursor position. Depending on the current location, an image-type element is inserted.

**Insert Entity**

Allows you to insert a predefined entity or character entity. Surrogate character entities (range #x10000 to #x10FFFF) are also accepted. Character entities can be entered in one of the following forms:

- `#<decimal value>` - e.g. #65
- `&<decimal value>` - e.g. &#65
- `#x<hexadecimal value>` - e.g. #x41
- `&#x<hexadecimal value>` - e.g. &#x41

**Style submenu**

This submenu includes the following text styling actions:

**Monospace**

Inserts or wraps content with a `<monospace>` element.

**Bold**
Emphasizes the selected text by surrounding it with a *bold* tag. You can use this action on multiple non-contiguous selections.

*Italic*

Emphasizes the selected text by surrounding it with an *italic* tag. You can use this action on multiple non-contiguous selections.

*Underline*

Emphasizes the selected text by surrounding it with an *underline* tag. You can use this action on multiple non-contiguous selections.

**List > Insert Unordered List**

Inserts an unordered list at the cursor position. A child list item is also automatically inserted by default. You can also use this action to convert selected paragraphs or other types of lists to an unordered list.

**List > Insert Ordered List**

Inserts an ordered list at the cursor position. A child list item is also automatically inserted by default. You can also use this action to convert selected paragraphs or other types of lists to an ordered list.

**Code**

Inserts or wraps content in a `<code>` element.

**Select submenu**

This submenu allows you to select the following:

**Element**

Selects the entire element at the current cursor position.

**Content**

Selects the entire content of the element at the current cursor position, excluding the start and end tag. Performing this action repeatedly will result in the selection of the content of the ancestor of the currently selected element content.

**Parent**

Selects the entire parent element at the current cursor position.

**Text submenu**

This submenu contains the following actions:

**To Lower Case**

Converts the selected content to lower case characters.

**To Upper Case**

Converts the selected content to upper case characters.
Capitalize Sentences

Converts to upper case the first character of every selected sentence.

Capitalize Words

Converts to upper case the first character of every selected word.

Count Words

Counts the number of words and characters (no spaces) in the entire document or in the selection for regular content and read-only content.

Note:
The content marked as deleted with change tracking (on page 3224) is ignored when counting words.

Convert Hexadecimal Sequence to Character (Ctrl + Shift + X (Command + Shift + X on macOS))

Converts a sequence of hexadecimal characters to the corresponding Unicode character (on page 463). The action can be invoked if there is a selection containing a valid hexadecimal sequence or if the cursor is placed at the right side of a valid hexadecimal sequence. A valid hexadecimal sequence can be composed of 2 to 4 hexadecimal characters and may or may not be preceded by the 0x or 0X prefix. Examples of valid sequences and the characters they will be converted to:

- 0x0045 will be converted to E
- 0x0125 to ĥ
- 265 to ɥ
- 2190 to ←

Note:
For more information about finding the hexadecimal value of a character, see Finding the Decimal, Hexadecimal, or Character Entity Equivalent (on page 466).

Refactoring submenu

Contains a series of actions designed to alter the XML structure of the document:

- Toggle Comment
  Encloses the currently selected text in an XML comment, or removes the comment if it is commented.

- Move Up (Alt + UpArrow)
  Moves the current node or selected nodes in front of the previous node.
**Move Down** (Alt + DownArrow)

Moves the current node or selected nodes after the subsequent node.

**Split Element** (Alt + Shift + D (Ctrl + Alt + D on macOS))

Splits the content of the closest element that contains the position of the cursor. Thus, if the cursor is positioned at the beginning or at the end of the element, the newly created sibling will be empty.

**Join Elements**

Joins two adjacent block elements (on page 3216) that have the same name. The action is available only when the cursor position is between the two adjacent block elements. Also, joining two block elements can be done by pressing the Delete or Backspace keys and the cursor is positioned between the boundaries of these two elements.

**Surround with Tags** (Ctrl + E (Command + E on macOS))

Allows you to choose a tag to enclose a selected portion of content. If there is no selection, the start and end tags are inserted at the cursor position.

- If the **Position cursor between tags** option (on page 212) is selected in the Content Completion preferences page, the cursor is placed between the start and end tag.
- If the **Position cursor between tags** option (on page 212) is not selected in the Content Completion preferences page, the cursor is placed at the end of the start tag, in an insert-attribute position.

**Surround with ‘[tag]’** (Ctrl + ForwardSlash (Command + ForwardSlash on macOS))

Surround the selected content with the last tag used.

**Rename Element**

The element from the cursor position, and any elements with the same name, can be renamed according with the options from the Rename dialog box.

**Delete Element Tags**

Deletes the tags of the closest element that contains the position of the cursor. This operation is also executed if the start or end tags of an element are deleted by pressing the Delete or Backspace keys.

**Remove All Markup**

Removes all the XML markup inside the selected block of content and keeps only the text content.

**Remove Text**
Removes the text content of the selected block of content and keeps the markup intact with empty elements.

**Attributes Refactoring Actions**

Contains built-in XML refactoring operations that pertain to attributes with some of the information preconfigured based upon the current context.

- **Add/Change attribute**
  
  Allows you to change the value of an attribute or insert a new one.

- **Convert attribute to element**
  
  Allows you to change an attribute into an element.

- **Delete attribute**
  
  Allows you to remove one or more attributes.

- **Rename attribute**
  
  Allows you to rename an attribute.

- **Replace in attribute value**
  
  Allows you to search for a text fragment inside an attribute value and change the fragment to a new value.

**Comments Refactoring Actions**

Contains built-in XML refactoring operations that pertain to comments with some of the information preconfigured based upon the current context.

- **Delete comments**
  
  Allows you to delete comments found inside one or more elements.

**Elements Refactoring Actions**

Contains built-in XML refactoring operations that pertain to elements with some of the information preconfigured based upon the current context.

- **Delete element**
  
  Allows you to delete elements.

- **Delete element content**
  
  Allows you to delete the content of elements.

- **Insert element**
  
  Allows you to insert new elements.

- **Rename element**
  
  Allows you to rename elements.

- **Unwrap element**
Allows you to remove the surrounding tags of elements, while keeping the content unchanged.

**Wrap element**

Allows you to surround elements with element tags.

**Wrap element content**

Allows you to surround the content of elements with element tags.

**Fragments Refactoring Actions**

Contains built-in XML refactoring operations that pertain to XML fragments with some of the information preconfigured based upon the current context.

**Insert XML fragment**

Allows you to insert an XML fragment.

**Replace element content with XML fragment**

Allows you to replace the content of elements with an XML fragment.

**Replace element with XML fragment**

Allows you to replace elements with an XML fragment.

**Review submenu**

This submenu includes the following actions:

**Track Changes**

Enables or disables the *Track Changes (on page 3224)* support for the current document.

**Accept Change(s) and Move to Next**

Accepts the *Tracked Change (on page 3224)* located at the cursor position or all of the changes in a selection and then moves to the next change. If you select a part of a *deletion or insertion* change, only the selected content is accepted.

**Accept All Changes**

Accepts all *Tracked Changes (on page 3224)* in the current document.

**Reject Change(s) and Move to Next**

Rejects the *Tracked Change (on page 3224)* located at the cursor position or all of the changes in a selection and then moves to the next change. If you select a part of a *deletion or insertion* change, only the selected content is rejected.

**Reject All Changes**

Rejects all *Tracked Changes (on page 3224)* in the current document.

**Comment Change**
Opens a dialog box that allows you to add a comment to an existing *Tracked Change (on page 3224)*. The comment will appear in a callout and a tooltip when hovering over the change. If the action is selected on an existing commented change, the dialog box will allow you to edit the comment.

**Highlight**

Enables the highlighting tool that allows you to mark text in your document.

**Colors**

Allows you to select the color for highlighting text.

**Stop highlighting**

Use this action to deactivate the highlighting tool.

**Remove highlight(s)**

Use this action to remove highlighting from the document.

**Add Comment**

Inserts a comment at the cursor position. The comment appears in a callout box and a tooltip (when hovering over the change).

**Show/Edit Comment**

Opens a dialog box that displays the discussion thread and allows the current user to edit comments that do not have replies. If you are not the author who inserted the original comment, the dialog box just displays the comment without the possibility of editing it.

**Remove Comment**

Removes a selected comment. If you remove a comment that contains replies, all of the replies will also be removed.

**Manage Reviews**

Opens the *Review view (on page 664)*.

**Folding submenu**

This submenu includes the following actions:

**Toggle Fold**

Toggles the state of the current fold.

**Collapse Other Folds** *(Ctrl + NumPad/ Command + NumPad on macOS)*

Folds all the elements except the current element.

**Collapse Child Folds** *(Ctrl + NumPad. Command + NumPad. on macOS)*

Folds the elements indented with one level inside the current element.
Expand Child Folds
Unfolds all child elements of the currently selected element.

Expand All (Ctrl + NumPad* (Command + NumPad* on macOS))
Unfolds all elements in the current document.

About Element > Go to Definition
Moves the cursor to the definition of the current element.

Inspect Styles
Opens the CSS Inspector view (on page 640) that allows you to examine the CSS rules that match the currently selected element.

Options
Opens the Author mode preferences page (on page 176) where you can configure various options with regard to the Author editing mode.

JATS Drag/Drop Actions
Dragging a file from the Project view (on page 402) and dropping it into a JATS document that is edited in Author mode, creates a link to the dragged file (the `<ext-link>` element with the `@xlink:href` attribute) at the drop location. Dragging an image file from the default file system application (Windows Explorer on Windows or Finder on macOS, for example) and dropping it into a JATS document inserts an image element (the `<inline-graphic>` element with the `@xlink:href` attribute) at the drop location, similar to the Insert Image toolbar action.

Tip:
For information about customizing Author mode actions for a particular framework (on page 3219) (document type), see the Customizing the Author Mode Editing Experience for a Framework (on page 2145) section.

Related Information:
Customizing the Author Mode Editing Experience for a Framework (on page 2145)

EPUB Document Type (Framework)

EPUB is an e-book file format that is a ZIP archive and can be downloaded and read on devices such as phones, tablets, computers, or e-readers. Oxygen XML Editor includes an Archive Browser view (on page 2008) that allows you to view the contents and structure of this type of file.

Three distinct frameworks (on page 3219) are supported for the EPUB document type:
• **NCX** - A declarative global navigation definition.

• **OCF** - The Open Container Format (OCF) defines a mechanism by which all components of an Open Publication Structure (OPS) can be combined into a single file system entity.

• **OPF** - The Open Packaging Format (OPF) defines the mechanism by which all components of a published work that conforms to the Open Publication Structure (OPS) standard (including metadata, reading order, and navigational information) are packaged in an OPS Publication.

**Note:**
Oxygen XML Editor supports OPF 2.0, OPF 3.0, and OPF 3.1.

**File Definition**

A file is considered to be an *EPUB* document if it has a file extension of `.epub`.

**Default Document Templates**

There are a variety of default *EPUB* templates available when creating new documents from templates (on page 368) and they can be found the following folders in Framework Templates: NCX, OCF, OPF 2.0, OPF 3.0, and OPF 3.1.

- The default templates for the **NCX** document types are located in the `{OXYGEN_INSTALL_DIR}/frameworks/ncx/templates` folder.
- The default templates for the **OCF** document types are located in the `{OXYGEN_INSTALL_DIR}/frameworks/ocf/templates` folder.
- The default template for the **OPF 2.0** document type is located in the `{OXYGEN_INSTALL_DIR}/frameworks/opf/templates/2.0` folder.
- The default template for the **OPF 3.0** document type is located in the `{OXYGEN_INSTALL_DIR}/frameworks/opf/templates/3.0` folder.
- The default template for the **OPF 3.1** document type is located in the `{OXYGEN_INSTALL_DIR}/frameworks/opf/templates/3.1` folder.

**Default Schema**

The default schema files for the various types of EPUB document types are located in the following directories:

- The default schema files for the **NCX** document types are located in the `{OXYGEN_INSTALL_DIR}/frameworks/ncx/schemas` folder.
- The default schema files for the **OCF** document types are located in the `{OXYGEN_INSTALL_DIR}/frameworks/ocf/schemas` folder.
- The default schema files for the **OPF 2.0** document type is located in the `{OXYGEN_INSTALL_DIR}/frameworks/opf/schemas/2.0` folder.
• The default schema files for the **OPF 3.0** document type is located in the 
  \[OXYGEN_INSTALL_DIR\]/frameworks/opf/schemas/3.0 folder.
• The default schema files for the **OPF 3.1** document type is located in the 
  \[OXYGEN_INSTALL_DIR\]/frameworks/opf/schemas/3.1 folder.

Related Information:
Working with Archive Files *(on page 2011)*

## Other Supported Document Types

Along with the fully supported built-in frameworks *(document types)* *(on page 1291)*, Oxygen XML Editor also provides limited support (including document templates) for editing a variety of other document types. All the specialized views, editors, actions, and options are dynamic according to the type of file that is opened or created. Other document types that are supported in Oxygen XML Editor include:

• **EPUB (NCX, OCF, OPF 2.0, 3.0, & 3.1)** *(on page 1426)* - A standard for e-book files.
• **OOXML (on page 2011)** - An XML-based file format for representing spreadsheets, charts, presentations, and word processing documents.
• **ODF (on page 2011)** - An free and open-source XML-based file format for electronic office documents, such as spreadsheets, charts, presentations, and word processing documents.
• **DocBook Targetset (on page 1336)** - For resolving cross-references when using olinks.
• **Ant Build Scripts (on page 934)** - A tool for automating software build processes, written in Java and primarily intended for use with Java.
• **XSLT Stylesheets (on page 886)** - A document type that provides a visual mode for editing XSLT stylesheets.
• **WSDL (on page 1051)** - Web Services Description Language is an XML language for describing the functionality offered by a web service.
• **Schematron (on page 1192)** - For making assertions about the presence or absence of patterns in XML documents. This document type applies to the ISO Schematron version.
• **Schematron Quick Fixes (SQF) (on page 815)** - An extension of the ISO standard Schematron, allows developers to define Quick Fixes *(on page 3223)* for Schematron errors.
• **StratML (Part 1 & 2) (on page 1183)** - Part 1 and 2 of the Strategy Markup Language specification.
• **XProc (on page 1190)** - A document type for processing XProc script files.
• **XML Schema (on page 947)** - Documents that provide support for editing annotations.
• **SVG (on page 1251)** - Scalable Vector Graphics is a language for describing two-dimensional graphics in XML.
• **MathML (on page 749)** - Mathematical Markup Language (2.0 and 3.0) is an application of XML for describing mathematical notations.
• **XLIFF (1.2, 2.0, 2.1) (on page 1184)** - XML Localization Interchange File Format is a standard for passing data between tools during a localization process.
• **XQuery (on page 1033)** - The common query language for XML.
• **CSS (on page 1076)** - Cascading Style Sheets is a language used for describing the look and formatting of a document.
• **LESS (on page 1080)** - A dynamic style sheet language that can be compiled into CSS.
• **Relax NG Schema (on page 1083)** - A schema language that specifies a pattern for the structure and content of an XML document.
• **NVDL Schema (on page 1102)** - Namespace Validation Dispatching Language allows you to specify sections of XML documents to be validated against various schemas.
• **JSON (on page 1110)** - JavaScript Object Notation is a lightweight data-interchange format.
• **Markdown (on page 1262)** - A lightweight markup language with plain text formatting syntax that can be converted to HTML or DITA.
• **JavaScript (on page 1184)** - Programming language of HTML and the Web.
• **XMLSpec** - A markup language for W3C specifications and other technical reports.
• **DITAVAL** - DITA conditional processing profile to identify the values you want to conditionally process for a particular output, build, or other purpose.
• **Daisy XML** - A technical standard for digital audio books, periodicals, and computerized text. It is designed to be an audio substitute for print material.
• **EAD** - Encoded Archival Description is an XML standard for encoding archival finding aids.
• **KML** - Keyhole Markup Language is an XML notation for expressing geographic visualization in maps and browsers.
• **Maven Project & Settings** - Project or settings file for Maven build automation tool that is primarily used for Java projects.
• **Oasis XML Catalog** - An XML Catalog (on page 3225) document that describes a mapping between external entity references and locally-cached equivalents.
• **Other Non-XML Files (on page 397)** - Oxygen XML Editor also includes a simple text editor and a variety of helpful features for creating and editing non-XML files.
Oxygen XML Editor supports custom frameworks (document types) contributed by the XML community (for example, the S1000D framework (on page 1430)). They provide support for additional functionality and XML vocabularies.

Similar to the built-in frameworks (on page 3219), the additional frameworks may define:

- A default grammar used for validation and content completion in both Author mode and Text mode.
- CSS stylesheets for rendering XML documents in Author mode.
- User actions invoked from toolbars or menus in Author mode.
- Built-in transformation scenarios used for publishing XML documents.
- XML Catalogs (on page 3225) used for mapping resources.
- New document templates to make it easy to create XML documents.
- User-defined extensions for customizing the interaction with the content author in Author mode.

**S1000D Document Type (Framework)**

S1000D is an international specification for the procurement and production of technical publications (mainly used in aerospace and aviation industries). It is an XML-based specification for preparing, managing, and using equipment maintenance and operations information.

S1000D is articulated based on three main notions:

- **Data Module** - It is an XML file that defines a standalone information unit.

- **Data Module Structure** - It defines how a Data Module is divided:
  - The Identification and Status part that identifies the Data Module within the CSDB structure.
  - The Content part that is the detailed information of the Data Module.

- **Common Source DataBase (CSDB)** - It defines how the Data Modules are arranged inside a publication.

Oxygen XML Editor does not have default built-in support for S1000D, but a company called Amplexor has developed a framework that can be installed in Oxygen XML Editor to add support for S1000D documents.

To install the framework in Oxygen XML Editor, follow these steps:
1. Download or clone the framework on GitHub and install it as an additional framework (on page 139).

2. Download the S1000D specifications package that contains samples and schemas at S1000D Downloads.

3. Copy the XML Schema files from ZIP\XML Schema Package\xml_schema_flat into the corresponding version of the xml_schema_flat folder.

Note:
To display the CGM images, you have to install the CGM Image Support Add-on (on page 2579).

If you want more advanced S1000D editing features, you can ask some of our partners.

**OpenAPI (Swagger) Document Type (Framework)**

The OpenAPI specification is a community-driven open specification that defines a standard, programming language-agnostic interface description for HTTP APIs, which allows both humans and computers to discover and understand the capabilities of a service without requiring access to source code, additional documentation, or inspection of network traffic. Use cases for machine-readable API definition documents include interactive documentation, code generation for documentation, automation of test cases, and more. OpenAPI documents describe an APIs services and are represented in either YAML or JSON formats.

In Oxygen XML Editor, you can, of course, edit OpenAPI documents in Text mode, but Oxygen XML Editor also includes a built-in framework that renders OpenAPI (Swagger) documents in the Author visual editing mode. The framework is an extension of the JSON framework. When opening a detected OpenAPI document in Author mode, you will have access to some form controls, collapsible sections, and other features to help you visualize and edit these documents.

Tip:
There is an OpenAPI sample document named petsore.json located in [OXYGEN-INSTALL-DIR]/samples/json/openapi that you can use to see how these documents are rendered in Oxygen XML Editor.

For more details about the OpenAPI Specification, along with example documents, go to https://github.com/OAI/OpenAPI-Specification.
11.

Publishing

XML documents can be transformed into a variety of user-friendly output formats that can be viewed by end-users. This process is known as a transformation.

Oxygen XML Editor includes numerous built-in transformation possibilities to publish XML content in various output formats (such as WebHelp, PDF, CHM, EPUB, JavaHelp, Eclipse Help, XHTML, etc.)

For transformations that are not included in your installed version of Oxygen XML Editor, simply install the tool chain required to perform the specific transformation and process the files in accordance with the processor instructions. A multitude of target formats are possible. The basic condition for a transformation to any format is that your source document is well-formed.

**Warning:**

Keep in mind that there could be instances where there are differences between what you see in **Author** mode and what you see in the published output. This is typically due to certain limitations in the publishing engine, especially when the source documents contain complex referencing.

**Transformation Scenarios**

A transformation scenario is a set of complex operations and settings that gives you the possibility to obtain outputs of multiple types (XML, HTML, PDF, EPUB, etc.) from the same source of XML files and stylesheets.

**Note:**

You need to use the appropriate stylesheet according to the source definition and the desired output. For example, if you want to transform into an HTML format using a DocBook stylesheet, your source XML document should conform with the DocBook DTD.

Executing a transformation scenario implies multiple actions, such as:

- Validating the input file.
- Obtaining intermediate output files (for example, formatting objects for the XML to PDF transformation).
- Using transformation engines to produce the output.

Before transforming an XML document in Oxygen XML Editor, you need to define a transformation scenario to apply to that document. A scenario is a set of values for various parameters that define a transformation. It is not related to a particular document, but rather to a document type. Oxygen XML Editor includes preconfigured...
built-in transformation scenarios (on page 1433), but you can also create new transformation scenarios (on page 1464).

When creating new transformation scenarios, the types that are available include:

- **Scenarios that Apply to XML Files** - This type of scenario contains the location of an XSLT stylesheet that is applied on the edited XML document, as well as other transformation parameters. For more information, see XML Transformation with XSLT (on page 1464) and XML Transformation with XQuery (on page 1480).

- **Scenarios that Apply to XSLT Files** - This type of scenario contains the location of an XML document that the edited XSLT stylesheet is applied to, as well as other transform parameters. For more information, see XSLT Transformation (on page 1510).

- **Scenarios that Apply to XQuery Files** - This type of scenario contains the location of an XML source, that the edited XQuery file is applied to, as well as other transform parameters. When the XML source is a native XML database, the XML source field of the scenario is empty because the XML data is read with XQuery-specific functions, such as `document()`. When the XML source is a local XML file, the URL of the file is specified in the XML input field of the scenario. For more information, see XQuery Transformation (on page 1536).

- **Scenarios that Apply to SQL Files** - This type of scenario specifies a database connection for the database server that runs the SQL file that is associated with the scenario. The data processed by the SQL script is located in the database.

- **Scenarios that Apply to XProc Files** - This type of scenario contains the location of an XProc script, as well as other transform parameters. For more information, see SQL Transformation (on page 1544).

- **DITA-OT Scenarios** - This type of scenario provides the parameters for an Ant transformation that executes a DITA-OT build script. Oxygen XML Editor includes a built-in version of Ant and a built-in version of DITA-OT, although you can also set other versions in the scenario. For more information, see DITA-OT Transformation (on page 1490).

- **ANT Scenarios** - This type of scenario contains the location of an Ant build script, as well as other transform parameters. For more information, see Ant Transformation (on page 1506).

**Note:**

Status messages generated during the transformation process are displayed in the **Information view (on page 512).**

**Built-in Transformation Scenarios**

Oxygen XML Editor includes preconfigured built-in transformation scenarios that are used for common transformations. They can be found in the various sections in the **Configure Transformation Scenario(s) dialog box (on page 1548)** or **Transformation Scenarios view (on page 1555).** All the built-in **document types (frameworks) (on page 3219)** that are included in Oxygen XML Editor have various transformation scenarios in their specific sections, including the most popular frameworks, such as DITA, DocBook, TEI, XHTML, JATS, OOXML, and more.
To obtain the desired output, use one of the following actions from the toolbar or Transform submenu in the contextual menu of the Project view (on page 402):

• **Apply Transformation Scenario(s) (Ctrl + Shift + T (Command + Shift + T on macOS))** - If you have associated transformation scenarios for the current document, this action will simply apply the association (on page 1548) and begin the transformation process. If an association is not detected, this action will open the Configure Transformation Scenario(s) dialog box (on page 1548) where you can choose the scenarios you want to apply.

• **Configure Transformation Scenario(s) (Ctrl + Shift + C (Command + Shift + C on macOS))** - This action will open the Configure Transformation Scenario(s) dialog box (on page 1548) where you can choose the scenarios you want to apply.

**Note:**

- You can apply a transformation even if the current document is not associated with a transformation scenario.
- If the document contains an `xml-stylesheet` processing instruction that references an XSLT stylesheet (commonly used to display the document in web browsers), Oxygen XML Editor prompts you to associate the document with a built-in transformation scenario.
- The default transformation scenario is suggested based on the processing instruction from the edited document.

**Related Information:**

Creating New Transformation Scenarios (on page 1464)
Editing a Transformation Scenario (on page 1545)
Configure Transformation Scenario(s) Dialog Box (on page 1548)
Applying Associated Transformation Scenarios (on page 1548)
Transformation Scenarios View (on page 1555)

**DITA Map Transformation Scenarios**

Built-in transformation scenarios allow you to transform DITA maps (on page 3218) to a variety of outputs, such as WebHelp, PDF, ODF, XHTML, EPUB, CHM, Kindle, and MS Word. Oxygen XML Editor also includes a special Run DITA-OT Integrator (on page 1455) that can be used to integrate a DITA-OT plugin and a DITA Map Metrics Report transformation that generates a statistics report for your DITA map. All of them are listed in the DITA Map section in the Configure Transformation Scenario(s) dialog box (on page 1548).

A variety of transformations scenarios are available for DITA maps (on page 3218):
• Built-in transformation scenarios allow you to transform a DITA map to a variety of outputs, such as WebHelp, PDF, ODF, XHTML, EPUB, CHM, Kindle, and MS Word.

• Run DITA-OT Integrator (on page 1455) - Use this transformation scenario if you want to integrate a DITA-OT plugin (on page 3164). This scenario runs an Ant task that integrates all the plugins from the DITA-OT/plugins directory.

• DITA Map Metrics Report - Use this type of transformation scenario if you want to generate a DITA map statistics report. It contains information such as:
  ◦ The number of processed maps and topics.
  ◦ Content reuse percentage.
  ◦ Number of elements, attributes, words, and characters used in the entire DITA map structure.
  ◦ DITA conditional processing attributes used in the DITA maps.
  ◦ Processing instructions.
  ◦ External links.
  ◦ All @outputclass attribute values gathered from the DITA project.

Important:
If you have cross references that point to content outside the scope of the DITA map, that referenced content will not be counted. For example, if you have links to topics that are not included in the DITA map hierarchy, the content in those topics is ignored when generating the statistics.

Related Information:
Editing a Transformation Scenario (on page 1545)
Configure Transformation Scenario(s) Dialog Box (on page 1548)
Applying Associated Transformation Scenarios (on page 1548)
DITA Topic Transformation Scenarios (on page 3103)

DITA Map WebHelp Responsive Transformation

DITA content can be transformed into several types of WebHelp Responsive systems (with or without a feedback section). The WebHelp Responsive layout and features (on page 1560) are designed to adapt to any device and screen size to provide an optimal viewing and interaction experience. Oxygen XML Editor also provides numerous possibilities for customizing the WebHelp Responsive output (on page 1642).

WebHelp Responsive Transformation Scenario

To publish a DITA map (on page 3218) as WebHelp Responsive output, follow these steps:

1. Select the Configure Transformation Scenario(s) action from the DITA Maps Manager (on page 2897) toolbar.
2. Select the DITA Map WebHelp Responsive scenario from the DITA Map section.
3. If you want to configure the transformation, click the **Edit** button.

**Step Result:** This opens an **Edit scenario** configuration dialog box that allows you to configure various options in the following tabs:

- **Templates Tab (on page 3106)** - This tab contains a set of built-in publishing templates (on page 1603) that you can use for the layout of your WebHelp system output. You can also create your own publishing templates or edit existing ones (on page 1642).
- **Parameters Tab (on page 3112)** - This tab includes numerous parameters that can be set to customize your WebHelp system output. See the Parameters section (on page ___) below for details about the most commonly used parameters for WebHelp Responsive transformations.
- **Feedback Tab (on page 3113)** - This tab is for those who want to add the Oxygen Feedback comments component at the bottom of each WebHelp page so that you can interact with your readers.
- **Filters Tab (on page 3114)** - This tab allows you to filter certain content elements from the generated output.
- **Advanced Tab (on page 3116)** - This tab allows you to specify some advanced options for the transformation scenario.
- **Output Tab (on page 3118)** - This tab allows you to configure options that are related to the location where the output is generated.

4. Click **Apply associated** to process the transformation.

**Result:** When the **DITA Map WebHelp Responsive** transformation is complete, the output is automatically opened in your default browser.

**General Parameters for Customizing WebHelp Responsive Output**

To customize a transformation scenario, you can edit various parameters, including the following most commonly used ones:

- **default.language**

  This parameter is used if the language is not detected in the DITA map. The default value is en-us.

- **clean.output**

  Deletes all files from the output folder before the transformation is performed (only no and yes values are valid and the default value is no).

- **editlink.remote.ditamap.url**

  Use this parameter in conjunction with editlink.web.author.url to add an *Edit* link next to the topic title in the WebHelp output. When a user clicks the link, the topic is opened in Oxygen XML Web Author where they can make changes that can be saved to a file server. The value should be set as the custom URL of the main DITA map. For example, a GitHub custom URL might look like this: https://getFileContent/oxyengxml/userguide/master/UserGuide.ditamap.

- **editlink.web.author.url**
This parameter needs to be used in conjunction with `editlink.remote.ditamap.url` to add an *Edit* link next to the topic title in the WebHelp output. When a user clicks the link, the topic is opened in Oxygen XML Web Author where they can make changes that can be saved to a file server. The value should be set as the URL of the Web Author installation. For example: https://www.oxygenxml.com/oxygen-xml-web-author/.

**editlink.present.only.path.to.topic**

When this parameter is set to "true", the DITA topic path is displayed to the right of each topic title in the WebHelp Responsive output. Also, when this parameter is used, the `editlink.ditamap.edit.url`, `editlink.remote.ditamap.url`, and `editlink.web.author.url` parameters are ignored.

**fix.external.refs.com.oxygenxml** (Only supported when the DITA-OT transformation process is started from Oxygen XML Editor)

The DITA Open Toolkit usually has problems processing references that point to locations outside of the directory of the processed *DITA map*. This parameter is used to specify whether or not the application should try to fix such references in a temporary files folder before the DITA Open Toolkit is invoked on the fixed references. The fix has no impact on your edited DITA content. Allowed values: *true* or *false* (default).

**force.unique**

When set to *true* (default value), the transformation will be forced to create unique output files for each instance of a resource when a map contains multiple references to a single topic.

**use.stemming**

Controls whether or not you want to include stemming search algorithms into the published output (default setting is *false*).

**webhelp.custom.resources**

The file path to a directory that contains resources files. All files from this directory will be copied to the root of the WebHelp output.

**webhelp.favicon**

The file path that points to an image to be used as a *favicon* in the WebHelp output.

**webhelp.reload.stylesheet**

Set this parameter to *true* if you have out of memory problems when generating WebHelp. It will increase processing time but decrease the memory footprint. The default value is *false*.

**webhelp.search.custom.excludes.file**

The path of the file that contains name patterns for HTML files that should not be indexed by the WebHelp search engine. Each exclude pattern must be on a new line. The patterns are considered to be relative to the output directory, and they accept wildcards such as * * (matches zero or more characters) or *?* (matches one character). For more information about the patterns, see https://ant.apache.org/manual/dirtasks.html#patterns.
webhelp.search.japanese.dictionary

The file path of the dictionary that will be used by the Kuromoji morphological engine for indexing Japanese content in the WebHelp pages. The encoding for the dictionary must be UTF8.

webhelp.search.enable.pagination

Specifies whether or not search results will be displayed on multiple pages. Allowed values are yes or no.

webhelp.search.index.elements.to.exclude

Specifies a list of HTML elements that will not be indexed by the search engine. The value of the @class attribute can be used to exclude specific HTML elements from indexing. For example, the div.not-indexed value will not index all <div> elements that have a @class attribute with the value of not-indexed. Use a comma separator to specify more than one element.

webhelp.search.page.numberOfItems

Specifies the number of search results items displayed on each page. This parameter is only used when the webhelp.search.enable.pagination parameter is enabled.

webhelp.search.ranking

If this parameter is set to false then the 5-star rating mechanism is no longer included in the search results that are displayed on the Search tab (default setting is true).

webhelp.search.stop.words.include

Specifies a list of words that will be ignored by the search engine. Use a comma separator to specify more than one word.

webhelp.show.changes.and.comments

When set to yes, user comments, replies to comments, and tracked changes are published in the WebHelp output. The default value is no.

webhelp.sitemap.base.url

Base URL for all the <loc> elements in the generated sitemap.xml file. If this parameter is specified, the loc element will contain the value of this parameter plus the relative path to the page. If this parameter is not specified, the loc element will only contain the relative path of the page (the relative file path from the @href attribute of a <topicref> element from the DITA map, appended to this base URL value).

webhelp.sitemap.change.frequency

The value of the <changefreq> element in the generated sitemap.xml file. The <changefreq> element is optional in sitemap.xml. If you leave this parameter set to its default empty value, then the <changefreq> element is not added in sitemap.xml. Allowed values: <empty string> (default), always, hourly, daily, weekly, monthly, yearly, never.

webhelp.sitemap.priority
The value of the `<priority>` element in the generated `sitemap.xml` file. It can be set to any fractional number between 0.0 (least important priority) and 1.0 (most important priority). For example, 0.3, 0.5, or 0.8. The `<priority>` element is optional in `sitemap.xml`. If you leave this parameter set to its default empty value, then the `<priority>` element is not added in `sitemap.xml`.

**Parameters Specific to Oxygen WebHelp Responsive**

**webhelp.fragment.feedback**

You can integrate Oxygen Feedback with your WebHelp Responsive output to provide a comments area at the bottom of each page where readers can offer feedback. When you create an Oxygen Feedback site configuration, an HTML fragment is generated during the final step of the creation process and that fragment should be set as the value for this parameter.

**webhelp.default.collection.type.sequence**

Specifies if the `sequence` value will be used by default when the `@collection-type` attribute is not specified. This option is helpful if you want to have Next and Previous navigational buttons generated for all HTML pages. Allowed values are no (default) and yes.

**webhelp.enable.search.autocomplete**

Specifies if the Autocomplete feature is enabled in the WebHelp search text field. The default value is yes.

**webhelp.enable.html.fragments.cleanup**

Enables or disables the automatic conversion of HTML fragments to well-formed XML. If set to true (default), the transformation automatically converts non-well-formed HTML content to a well-formed XML equivalent. If set to false, the transformation will fail if at least one HTML fragment is not well-formed.

**webhelp.fragment.after.body**

This parameter can be used to display a given XHTML fragment after the body in all types of pages. The value of the parameter can be either a well-formed XHTML fragment or a path to a file that contains a well-formed XHTML fragment.

**webhelp.fragment.after.body.main.page**

This parameter can be used to display a given XHTML fragment after the body in the main page. The value of the parameter can be either a well-formed XHTML fragment or a path to a file that contains a well-formed XHTML fragment.

**webhelp.fragment.after.body.topic.page**

This parameter can be used to display a given XHTML fragment after the body in the topic page. The value of the parameter can be either a well-formed XHTML fragment or a path to a file that contains a well-formed XHTML fragment.

**webhelp.fragment.after.body.search.page**
This parameter can be used to display a given XHTML fragment after the body in the search results page. The value of the parameter can be either a **well-formed** XHTML fragment or a path to a file that contains a **well-formed** XHTML fragment.

**webhelp.fragment.after.body.terms.page**

This parameter can be used to display a given XHTML fragment after the body in the index terms page. The value of the parameter can be either a **well-formed** XHTML fragment or a path to a file that contains a **well-formed** XHTML fragment.

**webhelp.fragment.after.logo_and_title**

This parameter can be used to display a given XHTML fragment after the logo and title in all types of pages. The value of the parameter can be either a **well-formed** XHTML fragment or a path to a file that contains a **well-formed** XHTML fragment.

**webhelp.fragment.after.search.input**

This parameter can be used to display a given XHTML fragment after the search field in all types of pages. The value of the parameter can be either a **well-formed** XHTML fragment or a path to a file that contains a **well-formed** XHTML fragment.

**webhelp.fragment.after.main.page.search (deprecated)**

This parameter is deprecated. Use **webhelp.fragment.after.search.input.main.page** instead.

**webhelp.fragment.after.search.input.main.page**

This parameter can be used to display a given XHTML fragment after the search field in all the main page. The value of the parameter can be either a **well-formed** XHTML fragment or a path to a file that contains a **well-formed** XHTML fragment.

**webhelp.fragment.after.search.input.topic.page**

This parameter can be used to display a given XHTML fragment after the search field in all the topic page. The value of the parameter can be either a **well-formed** XHTML fragment or a path to a file that contains a **well-formed** XHTML fragment.

**webhelp.fragment.after.search.input.search.page**

This parameter can be used to display a given XHTML fragment after the search field in all the search results page. The value of the parameter can be either a **well-formed** XHTML fragment or a path to a file that contains a **well-formed** XHTML fragment.

**webhelp.fragment.after.search.input.terms.page**

This parameter can be used to display a given XHTML fragment after the search field in all the index terms page. The value of the parameter can be either a **well-formed** XHTML fragment or a path to a file that contains a **well-formed** XHTML fragment.

**webhelp.fragment.after.toc_or_tiles**

This parameter can be used to display a given XHTML fragment after the table of contents or tiles in the main page. The value of the parameter can be either a **well-formed** XHTML fragment or a path to a file that contains a **well-formed** XHTML fragment.
**webhelp.fragment.after.top_menu**

This parameter can be used to display a given XHTML fragment after the top menu in all types of pages. The value of the parameter can be either a **well-formed** XHTML fragment or a path to a file that contains a **well-formed** XHTML fragment.

**webhelp.fragment.before.body**

This parameter can be used to display a given XHTML fragment before the page body in all types of pages. The value of the parameter can be either a **well-formed** XHTML fragment or a path to a file that contains a **well-formed** XHTML fragment.

**webhelp.fragment.before.body.main.page**

This parameter can be used to display a given XHTML fragment before the page body in the main page. The value of the parameter can be either a **well-formed** XHTML fragment or a path to a file that contains a **well-formed** XHTML fragment.

**webhelp.fragment.before.body.topic.page**

This parameter can be used to display a given XHTML fragment before the page body in the topic page. The value of the parameter can be either a **well-formed** XHTML fragment or a path to a file that contains a **well-formed** XHTML fragment.

**webhelp.fragment.before.body.search.page**

This parameter can be used to display a given XHTML fragment before the page body in the search results page. The value of the parameter can be either a **well-formed** XHTML fragment or a path to a file that contains a **well-formed** XHTML fragment.

**webhelp.fragment.before.body.terms.page**

This parameter can be used to display a given XHTML fragment before the page body in the index terms page. The value of the parameter can be either a **well-formed** XHTML fragment or a path to a file that contains a **well-formed** XHTML fragment.

**webhelp.fragment.before.logo_and_title**

This parameter can be used to display a given XHTML fragment before the logo and title. The value of the parameter can be either a **well-formed** XHTML fragment or a path to a file that contains a **well-formed** XHTML fragment.

**webhelp.fragment.before.search.input**

This parameter can be used to display a given XHTML fragment before the search field in all types of pages. The value of the parameter can be either a **well-formed** XHTML fragment or a path to a file that contains a **well-formed** XHTML fragment.

**webhelp.fragment.before.main.page.search (deprecated)**

This parameter is deprecated. Use `webhelp.fragment.before.search.input.main.page` instead.

**webhelp.fragment.before.search.input.main.page**
This parameter can be used to display a given XHTML fragment before the search field in the main page. The value of the parameter can be either a well-formed XHTML fragment or a path to a file that contains a well-formed XHTML fragment.

**webhelp.fragment.before.search.input.topic.page**

This parameter can be used to display a given XHTML fragment before the search field in the topic page. The value of the parameter can be either a well-formed XHTML fragment or a path to a file that contains a well-formed XHTML fragment.

**webhelp.fragment.before.search.input.search.page**

This parameter can be used to display a given XHTML fragment before the search field in the search results page. The value of the parameter can be either a well-formed XHTML fragment or a path to a file that contains a well-formed XHTML fragment.

**webhelp.fragment.before.search.input.terms.page**

This parameter can be used to display a given XHTML fragment before the search field in the index terms page. The value of the parameter can be either a well-formed XHTML fragment or a path to a file that contains a well-formed XHTML fragment.

**webhelp.fragment.before.toc_or_tiles**

This parameter can be used to display a given XHTML fragment before the table of contents or tiles in the main page. The value of the parameter can be either a well-formed XHTML fragment or a path to a file that contains a well-formed XHTML fragment.

**webhelp.fragment.before.top_menu**

This parameter can be used to display a given XHTML fragment before the top menu in all types of pages. The value of the parameter can be either a well-formed XHTML fragment or a path to a file that contains a well-formed XHTML fragment.

**webhelp.fragment.footer**

This parameter can be used to display a given XHTML fragment as the page footer in all types of pages. The value of the parameter can be either a well-formed XHTML fragment or a path to a file that contains a well-formed XHTML fragment.

**Important:**

This parameter should only be used if you are using a valid, purchased license of Oxygen XML Editor (do not use it with a trial license).

**webhelp.fragment.head**

This parameter can be used to display a given XHTML fragment in the header section in all types of pages. The value of the parameter can be either a well-formed XHTML fragment or a path to a file that contains a well-formed XHTML fragment.

**webhelp.fragment.head.main.page**
This parameter can be used to display a given XHTML fragment in the header section in the main page. The value of the parameter can be either a well-formed XHTML fragment or a path to a file that contains a well-formed XHTML fragment.

**webhelp.fragment.head.topic.page**

This parameter can be used to display a given XHTML fragment in the header section in the topic page. The value of the parameter can be either a well-formed XHTML fragment or a path to a file that contains a well-formed XHTML fragment.

**webhelp.fragment.head.search.page**

This parameter can be used to display a given XHTML fragment in the header section in the search results page. The value of the parameter can be either a well-formed XHTML fragment or a path to a file that contains a well-formed XHTML fragment.

**webhelp.fragment.head.terms.page**

This parameter can be used to display a given XHTML fragment in the header section in the index terms page. The value of the parameter can be either a well-formed XHTML fragment or a path to a file that contains a well-formed XHTML fragment.

**webhelp.fragment.welcome**

This parameter can be used to display a given XHTML fragment as a welcome message (or title). The value of the parameter can be either a well-formed XHTML fragment or a path to a file that contains a well-formed XHTML fragment.

**webhelp.fragment.after.header**

This parameter can be used to display a given XHTML fragment after the header section in all types of pages. The value of the parameter can be either a well-formed XHTML fragment or a path to a file that contains a well-formed XHTML fragment.

**webhelp.fragment.after.header.main.page**

This parameter can be used to display a given XHTML fragment after the header section in the main page. The value of the parameter can be either a well-formed XHTML fragment or a path to a file that contains a well-formed XHTML fragment.

**webhelp.fragment.after.header.topic.page**

This parameter can be used to display a given XHTML fragment after the header section in the topic page. The value of the parameter can be either a well-formed XHTML fragment or a path to a file that contains a well-formed XHTML fragment.

**webhelp.fragment.after.header.search.page**

This parameter can be used to display a given XHTML fragment after the header section in the search results page. The value of the parameter can be either a well-formed XHTML fragment or a path to a file that contains a well-formed XHTML fragment.

**webhelp.fragment.after.header.terms.page**
This parameter can be used to display a given XHTML fragment after the header section in the index terms page. The value of the parameter can be either a well-formed XHTML fragment or a path to a file that contains a well-formed XHTML fragment.

webhelp.fragment.before.search.input

This parameter can be used to display a given XHTML fragment before the search field in all types of pages. The value of the parameter can be either a well-formed XHTML fragment or a path to a file that contains a well-formed XHTML fragment.

webhelp.fragment.after.search.input

This parameter can be used to display a given XHTML fragment after the search field in all types of pages. The value of the parameter can be either a well-formed XHTML fragment or a path to a file that contains a well-formed XHTML fragment.

webhelp.fragment.before.main.content.area

This parameter can be used to display a given XHTML fragment before the main content section in all types of pages. The value of the parameter can be either a well-formed XHTML fragment or a path to a file that contains a well-formed XHTML fragment.

webhelp.fragment.before.main.content.area.main.page

This parameter can be used to display a given XHTML fragment before the main content section in the main page. The value of the parameter can be either a well-formed XHTML fragment or a path to a file that contains a well-formed XHTML fragment.

webhelp.fragment.before.main.content.area.topic.page

This parameter can be used to display a given XHTML fragment before the main content section in the topic page. The value of the parameter can be either a well-formed XHTML fragment or a path to a file that contains a well-formed XHTML fragment.

webhelp.fragment.before.main.content.area.search.page

This parameter can be used to display a given XHTML fragment before the main content section in the search results page. The value of the parameter can be either a well-formed XHTML fragment or a path to a file that contains a well-formed XHTML fragment.

webhelp.fragment.before.main.content.area.terms.page

This parameter can be used to display a given XHTML fragment before the main content section in the index terms page. The value of the parameter can be either a well-formed XHTML fragment or a path to a file that contains a well-formed XHTML fragment.

webhelp.fragment.after.main.content.area

This parameter can be used to display a given XHTML fragment after the main content section in all types of pages. The value of the parameter can be either a well-formed XHTML fragment or a path to a file that contains a well-formed XHTML fragment.

webhelp.fragment.after.main.content.area.main.page
This parameter can be used to display a given XHTML fragment after the main content section in the main page. The value of the parameter can be either a well-formed XHTML fragment or a path to a file that contains a well-formed XHTML fragment.

webhelp.fragment.after.main.content.area.topic.page

This parameter can be used to display a given XHTML fragment after the main content section in the topic page. The value of the parameter can be either a well-formed XHTML fragment or a path to a file that contains a well-formed XHTML fragment.

webhelp.fragment.before.topic.toolbar

This parameter can be used to display a given XHTML fragment before the toolbar buttons above the topic content in the topic page. The value of the parameter can be either a well-formed XHTML fragment or a path to a file that contains a well-formed XHTML fragment.

webhelp.fragment.after.topic.toolbar

This parameter can be used to display a given XHTML fragment after the toolbar buttons above the topic content in the topic page. The value of the parameter can be either a well-formed XHTML fragment or a path to a file that contains a well-formed XHTML fragment.

webhelp.fragment.before.topic.breadcrumb

This parameter can be used to display a given XHTML fragment before the breadcrumb component in the topic page. The value of the parameter can be either a well-formed XHTML fragment or a path to a file that contains a well-formed XHTML fragment.

webhelp.fragment.after.topic.breadcrumb

This parameter can be used to display a given XHTML fragment after the breadcrumb component in the topic page. The value of the parameter can be either a well-formed XHTML fragment or a path to a file that contains a well-formed XHTML fragment.

webhelp.fragment.before.publication.toc

This parameter can be used to display a given XHTML fragment before the publication's table of contents component in the topic page. The value of the parameter can be either a well-formed XHTML fragment or a path to a file that contains a well-formed XHTML fragment.

webhelp.fragment.after.publication.toc

This parameter can be used to display a given XHTML fragment after the publication's table of contents component in the topic page. The value of the parameter can be either a well-formed XHTML fragment or a path to a file that contains a well-formed XHTML fragment.

webhelp.fragment.before.topic.content

This parameter can be used to display a given XHTML fragment before the topic's content in the topic page. The value of the parameter can be either a well-formed XHTML fragment or a path to a file that contains a well-formed XHTML fragment.

webhelp.fragment.after.topic.content
This parameter can be used to display a given XHTML fragment after the topic's content in the topic page. The value of the parameter can be either a well-formed XHTML fragment or a path to a file that contains a well-formed XHTML fragment.

**webhelp.fragment.before.feedback**

This parameter can be used to display a given XHTML fragment before the Oxygen Feedback commenting component in the topic page. The value of the parameter can be either a well-formed XHTML fragment or a path to a file that contains a well-formed XHTML fragment.

**webhelp.fragment.after.feedback**

This parameter can be used to display a given XHTML fragment after the Oxygen Feedback commenting component in the topic page. The value of the parameter can be either a well-formed XHTML fragment or a path to a file that contains a well-formed XHTML fragment.

**webhelp.fragment.before.topic.toc**

This parameter can be used to display a given XHTML fragment before the topic's table of contents component in the topic page. The value of the parameter can be either a well-formed XHTML fragment or a path to a file that contains a well-formed XHTML fragment.

**webhelp.fragment.after.topic.toc**

This parameter can be used to display a given XHTML fragment after the topic's table of contents component in the topic page. The value of the parameter can be either a well-formed XHTML fragment or a path to a file that contains a well-formed XHTML fragment.

**webhelp.fragment.custom.search.engine.results**

This parameter can be used to replace the search results area with custom XHTML content. The value of the parameter is the path to an XHTML file that contains your custom content.

**webhelp.fragment.custom.search.engine.script**

This parameter can be used to replace WebHelp's built-in search engine with your own custom search engine. The value of the parameter is the path to an XHTML file that contains the scripts required for your custom search engine to run.

**webhelp.merge.nested.topics.related.links**

Specifies if the related links from nested topics will be merged with the links in the parent topic. Thus the links will be moved from the topic content to the related links component and all of the links from the same group (for example, Related Tasks, Related References, Related Information) are merged into a single group. The default value is yes.

**webhelp.search.stop.words.exclude**

Specifies a list of words that will be excluded from the default list of stop words that are filtered out before the search processing. Use comma separators to specify more than one word (for example: if, for, is).

**webhelp.show.breadcrumb**

Specifies if the breadcrumb component will be presented in the output. The default value is yes.
webhelp.show.child.links
Specifications if child links will be generated in the output for all topics that have subtopics. The default value is no.

webhelp.show.indexterms.link
Specifications if an icon that links to the index will be presented in the output. The default value is yes.

webhelp.show.main.page.tiles
Specifications if the tiles component will be presented in the main page of the output. For a tree style layout, this parameter should be set to no.

webhelp.show.main.page.toc
Specifications if the table of contents will be presented in the main page of the output. The default value is yes.

webhelp.show.navigation.links
Specifications if navigation links will be presented in the output. The default value is yes.

webhelp.show.print.link
Specifications if a print link or icon will be presented within each topic in the output. The default value is yes.

webhelp.show.related.links
Specifications if the related links component will be presented in the WebHelp Responsive output. The default value is yes. The webhelp.merge.nested.topics.related.links parameter can be used in conjunction with this one to merge the related links from nested topics into the links in the parent topic.

webhelp.show.publication.toc
Specifications if a table of contents will be presented on the left side of each topic in the output. The default value is yes.

webhelp.show.top.menu
Specifications if a menu will be presented at the topic of the main page in the output. The default value is yes.

webhelp.publication.toc.hide.chunked.topics
Specifications if the table of contents will contain links for chunked topics. The default value is yes.

webhelp.publication.toc.links
Specifications which links will be included in the table of contents. The possible values are:
- **chapter** (default) - The TOC will include links for the current topic, its children, its siblings, and its direct ancestor (including the direct ancestor’s siblings), and the parent chapter.
- **topic** - The TOC will only include links for the current topic and its direct children.
- **all** - The TOC will include all links.

**webhelp.publication.toc.tooltip.position**

By default, if a topic contains a `<shortdesc>` element, its content is displayed in a tooltip when the user hovers over its link in the table of contents. This parameter controls whether or not this tooltip is displayed and its position relative to the link. The possible values are:

- **left**
- **right** (default)
- **top**
- **bottom**
- **hidden** - The tooltip will not be displayed.

**webhelp.top.menu.activated.on.click**

When this parameter is activated (set to `yes`), clicking an item in the top menu will expand the submenu (if available). You can then click on a submenu item to open the item (topic). You can click outside the menu or press **ESC** to hide the menu. When set to `no` (default), hovering over a menu item displays the menu content.

**webhelp.top.menu.depth**

Specifies the maximum depth level of the topics that will be included in the top menu. The default value is 3. A value of 0 means that the menu has unlimited depth.

**webhelp.topic.collapsible.elements.initial.state**

Specifies the initial state of collapsible elements (tables with titles, nested topics with titles, sections with titles, index term groups). The possible values are `collapsed` or `expanded` (default value).

**Parameters for Adding a Link to PDF Documentation in WebHelp Responsive Output**

The following transformation parameters can be used to generate a PDF link component in the WebHelp Responsive output (for example, it could link to the PDF equivalent of the documentation):

**webhelp.pdf.link.url**

Specifies the target URL for the PDF link component.

**webhelp.pdf.link.text**

Specifies the text for the PDF link component.

**webhelp.pdf.link.icon.path**

Specifies the path or URL of the image icon to be used for the PDF link component. If not specified, a default icon is used.
webhelp.show.pdf.link

Specifies whether or not the PDF link component is shown in the WebHelp Responsive output.
Allowed values are: yes (default) and no.

webhelp.pdf.link.anchor.enabled

Specifies whether or not the current topic ID should be appended as the name destination at the end of the PDF link. Allowed values are: yes (default) and no.

Related information
Customizing WebHelp Responsive Output (on page 1642)
Layout and Features (on page 1560)

DITA Map PDF - based on HTML5 & CSS Transformation

Oxygen XML Editor includes a built-in DITA Map PDF - based on HTML5 & CSS transformation scenario based on a DITA-OT CSS-based PDF Publishing plugin that converts DITA maps to PDF using a CSS-based processing engine and an HTML5 intermediate format. Oxygen XML Editor comes bundled with a built-in CSS-based PDF processing engine called Oxygen PDF Chemistry. Oxygen XML Editor also supports some third-party processors.

For those who are familiar with CSS, this makes it very easy to style and customize the PDF output of your DITA projects without having to work with xsl:fo customizations. This transformation also includes some built-in publishing templates that you can use for the layout of your PDF output and you can create your own templates or edit existing ones.

The following CSS-based PDF processors can be used:

- Oxygen PDF Chemistry - A built-in processor that is bundled with Oxygen XML Editor. For more information, see the Oxygen PDF Chemistry User Guide. This is the supported processor.
- Prince Print with CSS (not included in the Oxygen XML Editor installation kit) - A third-party component that needs to be purchased from http://www.princexml.com.
- Antenna House Formatter (not included in the Oxygen XML Editor installation kit) - A third-party component that needs to be purchased from http://www.antennahouse.com/antenna1/formatter/.

How to Create the Transformation Scenario

To create a DITA Map PDF - based on HTML5 & CSS transformation scenario, follow these steps:

1. Click the Configure Transformation Scenario(s) button from the DITA Maps Manager (on page 2897) toolbar.
2. Select the DITA Map PDF - based on HTML5 & CSS transformation scenario.
3. If you want to configure the transformation, click the Edit button.
**Step Result:** This opens an **Edit scenario** configuration dialog box that allows you to configure various options in the following tabs:

- **Templates Tab (on page 3106)** - This tab contains a set of built-in publishing templates that you can use for the layout of your WebHelp system output. You can also create your own publishing templates by saving one from the gallery and changing it.

**Figure 469. DITA Map to PDF Templates**

- **Parameters Tab (on page 3112)** - This tab includes numerous parameters that can be set to customize the transformation.
- **Filters Tab (on page 3114)** - This tab allows you to filter certain content elements from the generated output.
- **Advanced Tab (on page 3116)** - This tab allows you to specify some advanced options for the transformation scenario.
- **Output Tab (on page 3118)** - This tab allows you to configure options that are related to the location where the output is generated.

4. In the **Parameters** tab, configure any of the following parameters (if applicable):

- **args.css** - Specifies a path to a custom CSS to be used in addition to those specified in the publishing template. The files must have URL syntax and be separated using semicolons. Also, the `dita.css.list` parameter must be left empty to use these files in addition to the selection in the Styles drop-down menu.
- **css.processor.type** - This is where you choose the processor type. You can select between Oxygen PDF Chemistry, Prince XML, or Antenna House.
Customizing the Output

For information about customizing the output, see CSS-based DITA to PDF Customization (on page 1768).

Related Information:

Editing a Transformation Scenario (on page 1545)
Configure Transformation Scenario(s) Dialog Box (on page 1548)
Oxygen PDF Chemistry User Guide
CSS-based DITA to PDF Customization (on page 1768)

DITA Map PDF - based on XSL-FO Transformation

Oxygen XML Editor comes bundled with the DITA Open Toolkit that provides a mechanism for converting DITA maps (on page 3218) to PDF output.

Creating a DITA Map PDF - based on XSL-FO Transformation Scenario

To create a DITA Map PDF - based on XSL-FO transformation scenario, follow these steps:

1. Click the Configure Transformation Scenario(s) button from the DITA Maps Manager (on page 2897) toolbar.
2. Select DITA Map PDF - based on XSL-FO and click the Edit button (or use the Duplicate button if your framework (on page 3219) is read-only).
3. Use the various tabs to configure the transformation scenario. In the Parameters tab, you can use a variety of parameters to customize the output. For example, the following parameters are just a few of the most commonly used ones:
- show.changes.and.comments - If set to yes, user comments, replies to comments, and tracked changes are published in the PDF output.
- customization.dir - Specifies the path to a customization directory.
- editlink.present.only.path.to.topic - When this parameter is set to "true", the DITA topic path is displayed to the right of each topic title in the PDF output.

4. Click OK and then the Apply Associated button to run the transformation scenario.

Related Information:
XSL FO-based DITA to PDF Customization (on page 1986)

DITA Map MS Office Word Transformation

Oxygen XML Editor comes bundled with a transformation scenario that allows you to convert DITA maps (on page 3218) to Microsoft Office Word documents. It utilizes the DITA to Word plugin created by Jarno Elovirta. This plugin contains a Word document named Normal.docx (located in: 
\{OXYGEN_INSTALL_DIR\}/frameworks/dita/DITA-OT3.x/plugins/com.elovirta.ooxml/resources) that is used by the transformation scenario as a template to generate the final Word document.

Tip:
You can make general modifications to the Normal.docx template file to alter the published output.

Configuring the Transformation Scenario

To configure a DITA Map to MS Office Word transformation scenario, follow these steps:

1. Open the DITA map in the DITA Maps Manager (on page 2897).
2. Click the Configure Transformation Scenario(s) button from the DITA Maps Manager toolbar.
3. Select DITA Map MS Office Word.
4. For advanced customizations, in the Parameters tab you can use any of the following parameters that are unique to this transformation scenario to specify paths to files that affect the output in various ways:
   - dotx.file - Specifies the path to a Word template file (.docx) that will be used in the transformation to generate the final Word document. Set this parameter if you want to use a different template file other than the Normal.docx file that is used by default.
   - document.flat.xsl - Specifies the path to a pre-process clean-up stylesheet.
   - core.xsl - Specifies the path to a core metadata stylesheet.
   - custom.xsl - Specifies the path to a custom metadata stylesheet.
   - document.xsl - Specifies the path to a main document stylesheet.
   - comments.xsl - Specifies the path to a comments stylesheet.
   - numbering.xsl - Specifies the path to a list and title numbering stylesheet.
   - footnotes.xsl - Specifies the path to a footnote stylesheet.
5. Click OK and run the transformation scenario.

Result: The result of the transformation will automatically be opened in your system's default word processing application (such as Microsoft Word).

Related Information:
- Editing a Transformation Scenario (on page 1545)
- Configure Transformation Scenario(s) Dialog Box (on page 1548)
- Migrating MS Office Documents to DITA (on page 3188)

DITA Map Markdown Transformation

Using the Configure Transformation Scenarios toolbar button, you can create a new transformation scenario of the type DITA-OT transformation and then choose one of the Markdown-specific output types:

- GitHub-flavored Markdown
- Markdown
- GitBook

More information about the Markdown output types is available in the DITA-OT documentation: https://www.dita-ot.org/dev/topics/dita2markdown.html.

The generated Markdown content can be used with a static web site generator (such as MKDocs) to build a web site: https://blog.oxygenxml.com/topics/publishing_dita_content_using_a_markdown_static_web_site_generator.html.

DITA Map CHM (Compiled HTML Help) Transformation

To perform a Compiled HTML Help (CHM) transformation, Oxygen XML Editor needs Microsoft HTML Help Workshop to be installed on your computer. Oxygen XML Editor automatically detects if HTML Help Workshop is installed and uses it.

Note:

HTML Help Workshop might fail if the files used for transformation contain accents in their names, due to different encodings used when writing the .hhp and .hcc files. If the transformation fails to produce the CHM output but the .hhp (HTML Help Project) file is already generated, you can manually try to build the CHM output using HTML Help Workshop.
Changing the Output Encoding

Oxygen XML Editor uses the `htmlhelp.locale` parameter to correctly display specific characters of different languages in the output of the Compiled HTML Help (CHM) transformation. By default, the DITA Map CHM transformation scenario that comes bundled with Oxygen XML Editor has the `htmlhelp.locale` parameter set to `en-US`.

To customize this parameter, follow this procedure:

1. Use the Configure Transformation Scenario(s) (`Ctrl + Shift + C (Command + Shift + C on macOS)`) action from the toolbar or the Document > Transformation menu.
2. Select the DITA Map CHM transformation scenario and click the Edit button.
3. In the Parameter tab, search for the `htmlhelp.locale` parameter and change its value to the desired language tag.

Note:
The format of the `htmlhelp.locale` parameter is `LL-CC`, where `LL` represents the language code (`en`, for example) and `CC` represents the country code (`US`, for example). The language codes are contained in the ISO 639-1 standard and the country codes are contained in the ISO 3166-1 standard. For further details about language tags, go to [http://www.rfc-editor.org/rfc/rfc5646.txt](http://www.rfc-editor.org/rfc/rfc5646.txt).

Customizing the CHM Output

There are several possibilities available for customizing the CHM output:

- You can use a custom CSS stylesheet to customize how the HTML content is rendered in the output:
  1. Create the custom CSS.
  2. Select the DITA Map CHM transformation scenario and click the Edit button.
  3. In the Parameter tab, set the `args.css` parameter to point to the location of your custom CSS and make sure the `args.copy.css` parameter is set to `yes` to instruct the transformation to copy the custom CSS to the output folder.
  4. Run the transformation.

- If you are familiar with XSLT, there are two XSLT stylesheets that are used in the transformation to compile various settings and components in the CHM output. They are found in the following directory: `OXYGEN_INSTALL_DIR/frameworks/dita/DITA-OT3.x/plugins/org.dita.htmlhelp/xsl/map2htmlhelp`. The files are as follows:
  - `map2hhcImpl.xsl` - This file is used to compile the table of contents.
  - `map2hhpImpl.xsl` - This file contains information for compiling the CHM and various settings that are read by the HTML Help Workshop when creating the output.
DITA Map Kindle Transformation

Oxygen XML Editor requires KindleGen to generate Kindle output from DITA maps (on page 3218). To install KindleGen for use by Oxygen XML Editor, follow these steps:

1. Go to www.amazon.com/kindleformat/kindlegen and download the zip file that matches your operating system.
2. Unzip the file on your local disk.
3. Start Oxygen XML Editor and open a DITA map in the DITA Maps Manager view (on page 2897).
4. Click the ➤ Configure Transformation Scenario(s) button.
5. Select the DITA Map Kindle transformation and click the Edit button to edit it.
6. Go to Parameters tab and set the kindlegen.executable parameter as the path to the KindleGen directory.
7. Accept the changes.

Run DITA-OT Integrator Transformation

Oxygen XML Editor comes bundled with a transformation scenario designed to integrate DITA-OT plugins (on page 3222). These DITA-OT plugins are used for various customizations. It is called Run DITA-OT Integrator and is found in the DITA Map section of the ➤ Configure Transformation Scenario(s) dialog box (on page 1548).

⚠️ Attention:
The integration will be performed on the DITA-OT version specified in the DITA Open Toolkit section of the DITA preferences page (on page 269).

Running the Transformation Scenario

To integrate a DITA-OT plugin, follow these steps:

1. If Oxygen XML Editor was installed in the default location, you may need to restart and run it as an administrator.
2. Select the ➤ Apply Transformation Scenario(s) or ➤ Configure Transformation Scenario(s) (on page 1548) action from the DITA Maps Manager toolbar (you could also use the same action on the main toolbar or open the Transformation Scenarios view (on page 1555)).
3. Select the Run DITA-OT Integrator transformation scenario. If the integrator is not visible, select the Show all scenarios action that is available in the ➤ Settings drop-down menu.
4. Apply the scenario (on page 1548).
5. Check the Results panel at the bottom of the application to make sure the build was successful.
6. Restart Oxygen XML Editor with your normal permissions.
Solving DITA Transformation Errors

If a DITA transformation results in errors or warnings, the information is displayed in the message panel at the bottom of the editor. The information includes the severity, description of the problem, the name of the resource, and the path of the resource.

To help prevent and solve DITA transformation problems, follow these steps:

1. **Validate the DITA map** (on page 2941) by using the **Validate and Check for Completeness** action that is available on the **DITA Maps Manager** (on page 2897) toolbar and in the **DITA Maps** menu.
2. If this action results in validation errors, solve them prior to executing the transformation. Also, you should pay attention to the warning messages because they may identify problems in the transformation.
3. **Run the DITA transformation scenario** (on page 1490).
4. If the transformation results in errors or warnings, they are displayed in the **Results panel** (on page 548) at the bottom of the editor. The following information is presented to help you troubleshoot the problems:
   - **Severity** - The first column displays the following icons that indicate the severity of the problem:
     - **Informational** - The transformation encountered a condition of which you should be aware.
     - **Warning** - The transformation encountered a problem that should be corrected.
     - **Error** - The transformation encountered a more severe problem, and the output is affected or cannot be generated.
   - **Info** - Click the **See More** icon to open a web page that contains more details about DITA-OT error messages.
   - **Description** - A description of the problem.
   - **Resource** - The name of the transformation resource.
   - **System ID** - The path of the transformation resource.
5. Use this information or other resources from the online DITA-OT community to solve the transformation problems before re-executing the transformation scenario.
6. If you need to contact the Oxygen technical support team, they will need you to send the entire transformation scenario execution log. To obtain it:
   a. Go to the **Options > Preferences > DITA preferences page** and set the **Show console output** option to **Always**.
   b. Execute the transformation scenario again. The console output messages are displayed in the **DITA-OT** view.
c. Copy the entire log, save it in a text file, then send it to the Oxygen technical support team.
d. After your issue has been solved, go back to the Options > Preferences > DITA preferences page and set the Show console output option to When build fails.

Related Information:
Troubleshooting DITA Transformation Problems (on page 3126)

DITA Topic Transformation Scenarios

Oxygen XML Editor includes built-in transformation scenarios for transforming individual DITA Topics to HTML5, XHTML, or PDF output. They can be found in the DITA section in the Configure Transformation Scenario(s) dialog box (on page 1548).

The available transformations scenarios for individual DITA topics include:

- **DITA HTML5** - This DITA-OT transformation scenario generates HTML5 output from a single DITA topic.
- **DITA XHTML** - This DITA-OT transformation scenario generates XHTML output from a single DITA topic. This was the first transformation scenario created for the DITA Open Toolkit and it originally served as the basis for all HTML-based transformations.
- **DITA PDF - based on HTML5 & CSS** - This transformation scenario converts individual DITA topics to PDF using a CSS-based processing engine and an HTML5 intermediate format. Oxygen XML Editor comes bundled with a built-in CSS-based PDF processing engine called Oxygen PDF Chemistry. Oxygen XML Editor also supports some third-party processors.

For those who are familiar with CSS, this makes it very easy to style and customize the PDF output of your DITA projects without having to work with xsl:fo customizations. Another advantage of this transformation scenario is that you can use the same customization CSS (on page 1814) or publishing template (on page 1783) that you use for converting entire DITA maps.

The transformation scenario automatically detects the currently selected context DITA map (root map) (on page 2901) so that keys and references are properly resolved (the detected context map is set as the value of the args.root.map parameter (this can be changed in the Parameters tab). It also automatically detects the currently applied profiling condition set (on page 3142) to be used as the default filtering option in the transformation scenario (this can be changed in the Filters tab).

The transformation scenario also supports a parameter named args.enable.root.map.key.processing that can be used to specify whether or not the values for @keyref and @conkeyref attributes within the transformed topics are resolved. The possible values are:

- **no** - This means that the values for all @keyref and @conkeyref attributes are ignored in the transformation. This results in lower processing times.
- **yes** - This means that the values for any @keyref and @conkeyref attributes found in the transformed topic are processed and resolved using the value of the args.root.map parameter.
- **auto** - This means that the process will search for any `@keyref` and `@conkeyref` attributes within the transformed topic and if any are found, the values will be processed and resolved using the value of the `args.root.map` parameter. If none are found, the `@keyref` and `@conkeyref` attributes are ignored.

- **DITA PDF - based on XSL-FO** - This DITA-OT transformation scenario converts individual DITA topics to PDF using an `xsl:fo` processor.

**DocBook Transformation Scenarios**

Built-in transformation scenarios allow you to transform DocBook documents to a variety of outputs, such as WebHelp, PDF, HTML, HTML Chunk, XHTML, XHTML Chunk, and EPUB. Oxygen XML Editor also includes a DocBook 5.1 transformation scenario for `Assembly` documents. All of them are listed in the DocBook 4 and DocBook 5 sections in the Configure Transformation Scenario(s) dialog box (on page 1548).

**WebHelp Classic Transformation Scenario**

DocBook documents can be transformed into several types of WebHelp systems (with or without a feedback section). The WebHelp Classic layout and features (on page 1734) are designed for desktop systems and include a familiar classical style. Oxygen XML Editor also provides numerous possibilities for customizing the WebHelp Classic output (on page 1741).

To publish a DocBook document as a WebHelp Classic system, follow these steps:

1. Click the Configure Transformation Scenario(s) action from the toolbar (or the Document > Transformation menu).
2. Select the DocBook WebHelp Classic scenario from the DocBook 4 or DocBook 5 section.
3. Click Apply associated.
Result: When the DocBook WebHelp Classic transformation is complete, the output is automatically opened in your default browser.

Customizing DocBook WebHelp Transformation Scenarios

To customize a DocBook WebHelp transformation scenario, you can edit various parameters, including the following most commonly used ones:

**default.language**

This parameter is used if the language is not detected in the DITA map. The default value is en-us.

**clean.output**

Deletes all files from the output folder before the transformation is performed (only no and yes values are valid and the default value is no).

**l10n.gentext.default.language**

This parameter is used to identify the correct stemmer that differs from language to language. For example, for English the value of this parameter is en or for French it is fr, and so on.

**use.stemming**

Controls whether or not you want to include stemming search algorithms into the published output (default setting is false).

**webhelp.copyright**

Adds a small copyright text that appears at the end of the Table of Contents pane.

**webhelp.custom.resources**

The file path to a directory that contains resources files. All files from this directory will be copied to the root of the WebHelp output.

**webhelp.favicon**

The file path that points to an image to be used as a favicon in the WebHelp output.

**webhelp.footer.file**

Path to an XML file that includes the footer content for your WebHelp output pages. You can use this parameter to integrate social media features (such as widgets for Facebook™, Twitter™, Google Analytics, or Google+™). The file must be well-formed, each widget must be in separate <div> or <span> element, and the code for each <script> element is included in an XML comment (also, the start and end tags for the XML comment must be on a separate line). The following code exert is an example for adding a Facebook™ widget:

```xml
<div id="facebook">
  <div id="fb-root"/>
  <script>  
    <!-- (function(d, s, id) { var js, fjs = d.getElementsByTagName(s)[0];
    if (d.getElementById(id)) return;
    js = d.createElement(s); js.id = id;
    js.src = "http://connect.facebook.net/en_us/all.js#xfbml=1";
    fjs.parentNode.insertBefore(js, fjs); 
    (d, s, "script", "facebook-jssdk")}})(document, 'script', 'facebook-jssdk'); 
  </script> 
</div>
```
webhelp.footer.include

Specifies whether or not to include footer in each WebHelp page. Possible values: yes, no. If set to no, no footer is added to the WebHelp pages. If set to yes and the webhelp.footer.file parameter has a value, then the content of that file is used as footer. If the webhelp.footer.file has no value then a default Oxygen footer is inserted in each WebHelp page.

webhelp.logo.image.target.url

Specifies a target URL that is set on the logo image. When you click the logo image, you will be redirected to this address.

webhelp.logo.image

Specifies a path to an image displayed as a logo in the left side of the output header.

webhelp.product.id (available only for Feedback-enabled systems)

This parameter specifies a short name for the documentation target, or product (for example, mobile-phone-user-guide, hvac-installation-guide).

Note:

You can deploy documentation for multiple products on the same server.

Restriction:

The following characters are not allowed in the value of this parameter: < > / \ ' ( ).

webhelp.product.version (available only for Feedback-enabled systems)

Specifies the documentation version number (for example, 1.0, 2.5, etc.). New user comments are bound to this version.

Note:

Multiple documentation versions can be deployed on the same server.
Restriction:
The following characters are not allowed in the value of this parameter: `< > / \ ' ( ) { } = ; * % + & .

webhelp.search-ranking

If this parameter is set to `false` then the 5-star rating mechanism is no longer included in the search results that are displayed on the Search tab (default setting is `true`).

webhelp.skin.css

Path to a CSS file that sets the style theme in the WebHelp Classic output. It can be one of the built-in skin CSS from the OXYGEN_INSTALL_DIR\frameworks\docbook\xsl\com.oxygenxml.webhelp.classic\predefined-skins directory, or it can be a custom skin CSS generated with the Oxygen Skin Builder web application.


Related information

Customizing WebHelp Classic Output (on page 1741)
Deploying the Oxygen Feedback Comments Component for DocBook (on page 1740)

DocBook to DITA Transformation

Oxygen XML Editor includes a built-in transformation scenario that is designed to convert DocBook content to DITA. This transformation scenario is based upon a DITA Open Toolkit plugin that is available at sourceforge.net.

To convert a DocBook document to DITA, follow these steps:

1. Use one of the following two methods to begin the transformation process:
   - To apply the transformation scenario to a newly opened file, use the Apply Transformation Scenario(s) (Ctrl + Shift + T (Command + Shift + T on macOS)) action from the toolbar or the Document > Transformation menu.
   - To customize the transformation or change the scenario that is associated with the document, use the Configure Transformation Scenario(s) (Ctrl + Shift + C (Command + Shift + C on macOS)) action from the toolbar or the Document > Transformation menu.

2. Select the DocBook to DITA transformation scenario in the DocBook 4 or DocBook 5 section.

3. Click the Apply associated button to run the transformation.
**Step Result:** The transformation will convert as many of the DocBook elements into equivalent DITA elements as it can recognize in its mapping process. For elements that cannot be mapped, the transformation will insert XML comments so that you can see which elements could not be converted.

4. Adjust the resulting DITA composite to suit your needs. You may have to remove comments, fix validation errors, adjust certain attributes, or split the content into individual topics.

**Related Information:**
- Editing a Transformation Scenario (on page 1545)
- Configure Transformation Scenario(s) Dialog Box (on page 1548)

### DocBook to PDF Transformation

Oxygen XML Editor includes a built-in transformation scenario that is designed to convert DocBook content to PDF.

To convert a DocBook document to PDF, follow these steps:

1. Use one of the following two methods to begin the transformation process:
   - To apply the transformation scenario to a newly opened file, use the **Apply Transformation Scenario(s) (Ctrl + Shift + T (Command + Shift + T on macOS))** action from the toolbar or the **Document > Transformation** menu.
   - To customize the transformation or change the scenario that is associated with the document, use the **Configure Transformation Scenario(s) (Ctrl + Shift + C (Command + Shift + C on macOS))** action from the toolbar or the **Document > Transformation** menu.

2. Select the **DocBook PDF** transformation scenario in the **DocBook 4** or **DocBook 5** section.

3. Click the **Apply associated** button to run the transformation.

For information about customizing the PDF output for DocBook content, see **DocBook to PDF Output Customization (on page 1997).**

**Related Information:**
- Editing a Transformation Scenario (on page 1545)
- Configure Transformation Scenario(s) Dialog Box (on page 1548)
- DocBook to PDF Output Customization (on page 1997)

### DocBook to EPUB Transformation

Oxygen XML Editor includes a built-in transformation scenario that is designed to convert DocBook content to EPUB. The EPUB specification recommends the use of **OpenType** fonts (recognized by their `.otf` file extension) whenever possible. To use a specific font, follow these steps:
1. Declare it in your CSS file, as in the following example:

```css
@font-face {
  font-family: "MyFont";
  font-weight: bold;
  font-style: normal;
  src: url(fonts/MyFont.otf);
}
```

2. In the CSS, specify where this font is used. To set it as default for `<h1>` elements, use the `font-family` rule, as in the following example:

```css
h1 {
  font-size: 20pt;
  margin-bottom: 20px;
  font-weight: bold;
  font-family: "MyFont";
  text-align: center;
}
```

3. Open the **Configure Transformation Scenario(s)** dialog box *(on page 1548)*, select the **DocBook EPUB** transformation scenario in the **DocBook 4** or **DocBook 5** section, and click **Edit**.

4. In the **Parameters** tab, set the `epub.embedded.fonts` parameter to `fonts/MyFont.otf`. If you need to provide more files, use commas to separate their file paths.

**Note:**
The `html.stylesheet` parameter allows you to include a custom CSS in the output EPUB.

5. Run the transformation scenario.

**DocBook PDF (Show Change Tracking and Comments)**

Oxygen XML Editor includes a built-in transformation scenario that is designed to show tracked changes and comment in DocBook to PDF output.

To include comments and *tracked changes* (stored within your DocBook 5 documents) in the PDF output, follow these steps:

1. Click the **Configure Transformation Scenario(s)** button.
2. Select **DocBook PDF (Show Change Tracking and Comments)** in the **DocBook 5** section.
3. If you need to configure the transformation, click the **Edit** *(on page 1545)* or **Duplicate** *(on page 1547)* button, make your changes to the scenario, and click **OK**.
4. Click the **Apply Associated** button to run the transformation scenario.

**Result:** Comment threads and tracked changes will now appear in the PDF output. Details about each comment or change will be available in the footer section for each page.
Creating New Transformation Scenarios

Defining a transformation scenario is the first step in the process of transforming a document. This section includes information on the types of new scenarios that are available in Oxygen XML Editor and how to create each type of transformation.

XML Transformation with XSLT

This type of transformation specifies the transformation parameters and location of an XSLT stylesheet that is applied to the edited XML document. This scenario is useful when you develop an XML document and the XSLT document is in its final form.

To create an XML transformation with XSLT scenario, use one of the following methods:

- Use the Configure Transformation Scenario(s) (Ctrl + Shift + C (Command + Shift + C on macOS)) action from the toolbar or the Document > Transformation menu. Then click the New button and select XML transformation with XSLT.
- Go to Window > Show View and select Transformation Scenarios to display this view (on page 1555). Click the New Scenario drop-down menu button and select XML transformation with XSLT.

Both methods open the New Scenario dialog box.

The upper part of the dialog box allows you to specify the Name of the transformation scenario and the following Storage options:

- Project Options (on page 3222) - The scenario is stored in the project file and can be shared with other users. For example, if your project is saved on a source versioning/sharing system (CVS, SVN, Source Safe, etc.) or a shared folder, your team can use the scenarios that you store in the project file.
- Global Options (on page 3219) - The scenario is saved in the global options that are stored in the user home directory and is not accessible to other users.

The lower part of the dialog box contains several tabs that allow you to configure the options that control the transformation.

XSLT Tab

When you create a new transformation scenario (on page 1464) or edit an existing one (on page 1545), a configuration dialog box is displayed that allows you to customize the transformation with various options in several tabs.

The XSLT tab contains the following options:

**XML URL**

Specifies the source XML file. You can specify the path by using the text field, its history drop-down, the Insert Editor Variables (on page 323) button, or the browsing actions in
the Browse drop-down list. You can also use the Open in editor button to open the specified file in the editor panel. This URL is resolved through the catalog resolver. If the catalog does not have a mapping for the URL, then the file is used directly from its remote location.

**Note:**
If the transformer engine is Saxon 9.x and a custom URI resolver is configured in the advanced Saxon preferences page (on page 249), the XML input of the transformation is passed to that URI resolver. If the transformer engine is one of the built-in XSLT 2.0 / 3.0 engines and the name of an initial template (on page 1465) is specified in the scenario, the XML URL field can be empty. The XML URL field can also be empty if you use external XSLT processors (on page 1479). Otherwise, a value is mandatory in this field.

**XSL URL**

Specifies the source XSL file that the transformation will use. You can specify the path by using the text field, its history drop-down, the Insert Editor Variables (on page 323) button, or the browsing actions in the Browse drop-down list. You can also use the Open in editor button to open the specified file in the editor panel. This URL is resolved through the catalog resolver. If the catalog does not have a mapping for the URL, the file is used directly from its remote location.

**Use "xml-stylesheet" declaration**

If selected, the scenario applies the stylesheet specified explicitly in the XML document with the xml-stylesheet processing instruction. By default, this option is deselected and the transformation applies the XSLT stylesheet that is specified in the XSL URL field.

**Transformer**

This drop-down menu presents all the transformation engines available to Oxygen XML Editor for performing a transformation. These include the built-in engines and the external engines defined in the Custom Engines preferences page (on page 259). The engine you choose is used as the default transformation engine. Also, if an XSLT or XQuery document does not have an associated validation scenario, this transformation engine is used in the validation process (if it provides validation support).

**Advanced options**

Allows you to configure the advanced options of the Saxon HE/PE/EE engine (on page 1468) for the current transformation scenario. To configure the same options globally, go to the Saxon-HE/PE/EE preferences page (on page 247). For the current transformation scenario, these Advanced options override the options configured in that preferences page.

**Parameters**

Opens a Configure parameters dialog box (on page 1466) that allows you to configure the XSLT parameters used in the current transformation. In this dialog box, you can also configure the parameters for additional XSLT stylesheets (on page 1466). If the XSLT transformation engine is
custom-defined, you cannot use this dialog box to configure the parameters sent to the custom engine. Instead, you can copy all parameters from the dialog box using contextual menu actions and edit the custom XSLT engine to include the necessary parameters in the command line that starts the transformation process.

Extensions

Opens a dialog box for configuring the XSLT extension JARS or classes (on page 1467) that define extension Java functions or extension XSLT elements used in the transformation.

Additional XSLT stylesheets

Opens a dialog box for adding XSLT stylesheets (on page 1468) that are applied on the main stylesheet specified in the XSL URL field. This is useful when a chain of XSLT stylesheets must be applied to the input XML document.

XSLT Parameters

The global parameters of the XSLT stylesheet used in a transformation scenario can be configured by using the Parameters button in the XSLT tab of a new or edited transformation scenario dialog box.

The resulting dialog box includes a table that displays all the parameters of the current XSLT stylesheet, all imported and included stylesheets, and all additional stylesheets (on page 1468), along with their descriptions and current values. You can also add, edit, and remove parameters, and you can use the Filter text box to search for a specific term in the entire parameters collection. Note that edited parameters are displayed with their name in bold.

If the XPath column is selected, the parameter value is evaluated as an XPath expression before starting the XSLT transformation.

Example:

For example, you can use expressions such as:

```
doc('test.xml')//entry
//person[@atr='val']
```

Note:

1. The doc function solves the argument relative to the XSL stylesheet location. You can use full paths or editor variables (on page 323) (such as $cfdu [current file directory]) to specify other locations: doc('$(cfdu)/test.xml')//*
2. You cannot use XSLT Functions. Only XPath functions are allowed.

Below the table, the following actions are available for managing the parameters:

New
Opens the **Add Parameter** dialog box that allows you to add a new parameter to the list. An **editor variable (on page 323)** can be inserted in the text box using the 📒 ![Insert Editor Variables](image). If the **Evaluate as XPath** option is selected, the parameter will be evaluated as an XPath expression.

**Edit**

Opens the **Edit Parameter** dialog box that allows you to edit the selected parameter. An **editor variable (on page 323)** can be inserted in the text box using the 📒 ![Insert Editor Variables](image). If the **Evaluate as XPath** option is selected, the parameter will be evaluated as an XPath expression.

**Unset**

Resets the selected parameter to its default value. Available only for edited parameters with set values.

**Delete**

Removes the selected parameter from the list. It is available only for new parameters that have been added to the list.

The bottom panel presents the following:

- The default value of the parameter selected in the table.
- A description of the parameter, if available.
- The system ID of the stylesheet that declares it.

**Related Information:**

*Editor Variables (on page 323)*

**XSLT Extensions**

The **Extensions** button opens a dialog box that allows you to specify the [JARS (on page 3220)](image) and classes that contain extension functions called from the XSLT file of the current transformation scenario. You can use the **Add**, **Edit**, and **Remove** buttons to configure the extensions.

**Tip:**

You can specify the path to the resources using wildcards (for example, `${oxygenHome}/lib/*.jar`).

An extension function called from the XSLT file of the current transformation scenario will be searched, in the specified extensions, in the order displayed in this dialog box. To change the order of the items, select the item to be moved and click the 🔄 Move up or ⌬ Move down buttons.
**Additional XSLT Stylesheets**

Use the **Additional XSLT Stylesheets** button in the **XSLT** tab to display a list of additional XSLT stylesheets to be used in the transformation and you can add files to the list or edit existing entries. The following actions are available:

- **Add**
  
  Adds a stylesheet in the **Additional XSLT stylesheets** list using a file browser dialog box. You can type an editor variable *(on page 323)* in the file name field of the browser dialog box. The name of the stylesheet will be added in the list after the current selection.

- **Remove**
  
  Deletes the selected stylesheet from the **Additional XSLT stylesheets** list.

- **Open**
  
  Opens the selected stylesheet in a separate view.

- **Up**
  
  Moves the selected stylesheet up in the list.

- **Down**
  
  Moves the selected stylesheet down in the list.

**Advanced Saxon HE/PE/EE XSLT Transformation Options**

The XSLT transformation scenario allows you to configure advanced options that are specific for the Saxon HE (Home Edition), PE (Professional Edition), and EE (Enterprise Edition) engines. They are the same options as those in the **Saxon HE/PE/EE preferences page (on page 247)** but they are configured as a specific set of transformation options for each transformation scenario, while the values set in the preferences page apply as global options. The advanced options configured in a transformation scenario override the **global options (on page 3219)** defined in the preferences page.

**Saxon-HE/PE/EE Options**

The advanced options for Saxon 10.6 Home Edition (HE), Professional Edition (PE), and Enterprise Edition (EE) are as follows:

- **Mode** ("-im")
  
  A Saxon-specific option that sets the initial mode for the transformation. If this value is also specified in a **configuration file (on page 1469)**, the value in this option takes precedence.

- **Template** ("-it")
  
  A Saxon-specific option that sets the name of the initial XSLT template to be executed. If this value is also specified in a **configuration file (on page 1469)**, the value in this option takes precedence.
Tip:
If your stylesheet includes `<xsl:template name="xsl:initial-template">`, Oxygen XML Editor will automatically detect and use it as the initial template, so this option is not needed in this case.

**Use a configuration file ("-config")**

Select this option if you want to use a Saxon 10.6 configuration file that will be executed for the XSLT transformation and validation processes. You can specify the path to the configuration file by entering it in the URL field, or by using the Insert Editor Variables button, or using the browsing actions in the Browse drop-down list.

**Debugger trace into XPath expressions (applies to debugging sessions)**

Instructs the XSLT Debugger (on page 2121) to step into XPath expressions.

**Enable Optimizations ("-opt")**

This option is selected by default, which means that optimization is enabled. If not selected, the optimization is suppressed, which is helpful when reducing the compiling time is important, optimization conflicts with debugging, or optimization causes extension functions with side-effects to behave unpredictably.

**Line numbering ("-l")**

Line numbers where errors occur are included in the output messages.

**Expand attributes defaults ("-expand")**

Specifies whether or not the attributes defined in the associated DTD or XML Schema are expanded in the output of the transformation you are executing.

**DTD validation of the source ("-dtd")**

Specifies whether or not the source document will be validated against their associated DTD. You can choose from the following:

- **On** - Requests DTD validation of the source file and of any files read using the `document()` function.
- **Off** - (default setting) Suppresses DTD validation.
- **Recover** - Performs DTD validation but treats the errors as non-fatal.

**Note:**
Any external DTD is likely to be read even if not used for validation, since DTDs can contain definitions of entities.

**Strip whitespaces ("-strip")**

Specifies how the strip whitespaces operation is handled. You can choose one of the following values:
• All ("all") - Strips all whitespace text nodes from source documents before any further processing, regardless of any @xml:space attributes in the source document.
• Ignore ("ignorable") - Strips all ignorable whitespace text nodes from source documents before any further processing, regardless of any @xml:space attributes in the source document. Whitespace text nodes are ignorable if they appear in elements defined in the DTD or schema as having element-only content.
• None ("none") - Strips no whitespace before further processing.

Enable profiling ("-TP")

If selected, profiling of the execution time in a stylesheet is enabled. The corresponding text field is used to specify the path to the output file where the profiling information will be saved. As long as the option is selected, and the output file specified, it will gather timed tracing information and create a profile report to the specified file.

Saxon-PE/EE Options

The following advanced options are specific for Saxon 10.6 Professional Edition (PE) and Enterprise Edition (EE) only:

Register Saxon-JS extension functions and instructions

Registers the Saxon-CE extension functions and instructions when compiling a stylesheet using the Saxon 10.6 processors.

Note:
Saxon-CE, being JavaScript-based, was designed to run inside a web browser. This means that you will use Oxygen XML Editor only for developing the Saxon-CE stylesheet, leaving the execution part to a web browser. See more details about executing such a stylesheet on Saxonica's website.

Allow calls on extension functions ("-ext")

If selected, the stylesheet is allowed to call external Java functions. This does not affect calls on integrated extension functions, including Saxon and EXSLT extension functions. This option is useful when loading an untrusted stylesheet (such as from a remote site using http://[URL]). It ensures that the stylesheet cannot call arbitrary Java methods and thus gain privileged access to resources on your machine.

Enable assertions ("-ea")

In XSLT 3.0, you can use the <xsl:assert> element to make assertions in the form of XPath expressions, causing a dynamic error if the assertion turns out to be false. If this option is selected, XSLT 3.0 <xsl:assert> instructions are enabled. If it is not selected (default), the assertions are ignored.
Saxon-EE Options

The advanced options that are specific for Saxon 10.6 Enterprise Edition (EE) are as follows:

**XML Schema version**

Use this option to change the default XML Schema version for this transformation. To change the default XML Schema version globally, open the Preferences dialog box (Options > Preferences) (on page 124) and go to XML > XML Parser > XML Schema and use the Default XML Schema version option (on page 238).

**Validation of the source file (“-val”)**

Requests schema-based validation of the source file and of any files read using `document()` or similar functions. It can have the following values:

- **Schema validation ("strict")** - This mode requires an XML Schema and allows for parsing the source documents with strict schema-validation enabled.
- **Lax schema validation ("lax")** - If an XML Schema is provided, this mode allows for parsing the source documents with schema-validation enabled but the validation will not fail if, for example, element declarations are not found.
- **Disable schema validation** - This specifies that the source documents should be parsed with schema-validation disabled.

**Validation errors in the result tree treated as warnings ("-outval")**

Normally, if validation of result documents is requested, a validation error is fatal. Selecting this option causes such validation failures to be treated as warnings.

**Write comments for non-fatal validation errors of the result document**

The validation messages for non-fatal errors are written (wherever possible) as a comment in the result document itself.

**Enable streaming mode**

Selecting this option will allow an XSLT to run in streaming mode. It is not selected by default.

**Other Options**

**Initializer class**

Equivalent to the `-init` Saxon command-line argument. The value is the name of a user-supplied class that implements the `net.sf.saxon.lib.Initializer` interface. This initializer is called during the initialization process, and may be used to set any options required on the configuration programmatically. It is particularly useful for tasks such as registering extension functions, collations, or external object models, especially in Saxon-HE where the option cannot be set via a configuration file. Saxon only calls the initializer when running from the command line, but the same code may be invoked to perform initialization when running user application code.
Using Saxon Integrated Extension Functions

Saxon, the transformation and validation engine used by Oxygen XML Editor, can be customized by adding custom functions (called Integrated Extension Functions) that can be called from XPath.

To define such a function, follow these steps:

1. Create a file with a Java class that extends `net.sf.saxon.lib.ExtensionFunctionDefinition`. Here is an example:

   ```java
   private static class ShiftLeft extends ExtensionFunctionDefinition {
   
   @Override
   public StructuredQName getFunctionQName() {
       return new StructuredQName("eg", "http://example.com/saxon-extension", "shift-left");
   }
   
   @Override
   public SequenceType[] getArgumentTypes() {
       return new SequenceType[]{SequenceType.SINGLE_INTEGER, SequenceType.SINGLE_INTEGER};
   }
   
   @Override
   public SequenceType getResultType(SequenceType[] suppliedArgumentTypes) {
       return SequenceType.SINGLE_INTEGER;
   }
   
   @Override
   public ExtensionFunctionCall makeCallExpression() {
       return new ExtensionFunctionCall() {
           public SequenceIterator call(SequenceIterator[] arguments, XPathContext context)
               throws XPathException {
               long v0 = ((IntegerValue)arguments[0].next()).longValue();
               long v1 = ((IntegerValue)arguments[1].next()).longValue();
               long result = v0<<v1;
               return Value.asIterator(Int64Value.makeIntegerValue(result));
           }
       };
   }
   }
   ```

2. Compile the class and add it to a JAR file.
3. Add a file called `net.sf.saxon.lib.ExtensionFunctionDefinition` that contains the fully qualified name of the Java class in the `META-INF/services/` folder of the JAR file.

**Note:**

To add more function definitions in the same JAR file, you need to add their fully qualified names on different lines.

To enable Oxygen XML Editor to pick up your custom function definition, the JAR file should be added to the classpath of the transformer. Here are some possibilities:

- If you develop a framework, you just need to link the JAR file in the Classpath tab (on page 144).
- In a validation scenario (on page 788), you can use the Extensions button to open a dialog box where you can add libraries.
- In a transformation scenario, you can use the Extensions button in the XSLT tab (on page 1466) to open a dialog box where you can add libraries.
- You can also create a plugin that contributes such a JAR file in the classpath (on page 2411).

**FO Processor Tab (XSLT Transformations)**

When you create a new transformation scenario (on page 1464) or edit an existing one (on page 1545), a configuration dialog box is displayed that allows you to customize the transformation with various options in several tabs.

The FO Processor tab contains the following options:

**Perform FO Processing**

Specifies whether or not an FO processor is applied (either the built-in Apache FOP engine or an external engine defined in Preferences) during the transformation.

**Input**

Choose between the following options to specify which input file to use:

- **XSLT result as input** - The FO processor is applied to the result of the XSLT transformation that is defined in the XSLT tab.
- **XML URL as input** - The FO processor is applied to the input XML file.

**Method**

The output format of the FO processing. The available options depend on the selected processor type.

**Processor**

 Specifies the FO processor to be used for the transformation. It can be the built-in Apache FOP processor or an external processor (on page 261).
Output Tab (XSLT Transformations)

When you create a new transformation scenario (on page 1464) or edit an existing one (on page 1545), a configuration dialog box is displayed that allows you to customize the transformation with various options in several tabs.

The Output tab contains the following options:

**Prompt for file**

At the end of the transformation, a file browser dialog box is displayed for specifying the path and name of the file that stores the transformation result.

**Save As**

The path of the file where the result of the transformation is stored. You can specify the path by using the text field, the 📋 Insert Editor Variables (on page 323) button, or the 📦 Browse button.

**Open in Browser/System Application**

If selected, Oxygen XML Editor automatically opens the result of the transformation in a system application associated with the file type of the result (for example, in Windows PDF files are often opened in Acrobat Reader).

---

**Note:**

To set the web browser that is used for displaying HTML/XHTML pages, go to Options > Preferences > Global, and set it in the Default Internet browser field.

- **Output file** - When Open in Browser/System Application is selected, you can use this button to automatically open the default output file at the end of the transformation.
- **Other location** - When Open in Browser/System Application is selected, you can use this option to open the file specified in this field at the end of the transformation. You can specify the path by using the text field, the 📋 Insert Editor Variables (on page 323) button, or the 📦 Browse button.

**Open in editor**

When this is option is selected, at the end of the transformation, the default output file is opened in a new editor panel with the appropriate built-in editor type (for example, if the result is an XML file it is opened in the built-in XML editor, or if it is an XSL-FO file it is opened with the built-in FO editor).

**Show in results view as**

You can choose to view the results in one of the following:
• **XML** - If this is selected, Oxygen XML Editor displays the transformation result in an XML viewer panel at the bottom of the application window with syntax highlighting (on page 224).

• **SVG** - If this is selected, Oxygen XML Editor displays the transformation result in an integrated SVG viewer in the Results panel (on page 1254) at the bottom of the application window and renders the result as an SVG image.

• **XHTML** - This option is only available if **Open in Browser/System Application** is not selected. If selected, Oxygen XML Editor displays the transformation result in a built-in XHTML browser panel at the bottom of the application window.

| Important:
| When transforming very large documents, you should be aware that selecting this option may result in very long processing times. This drawback is due to the built-in Java XHTML browser implementation. To avoid delays for large documents, if you want to see the XHTML result of the transformation, you should use an external browser by selecting the **Open in Browser/System Application** option instead. |

• **Image URLs are relative to** - If **Show in results view as XHTML** is selected, this option specifies the path used to resolve image paths contained in the transformation result. You can specify the path by using the text field, the ![Insert Editor Variables](on page 323) button, or the ![Browse](button). |  

| Attention:
| If your input XSLT contains `<xsl:result-document>` elements, then the secondary results will be saved to the specified URIs while the principal result is specified in this **Output** tab. For more information, see: [https://www.w3.org/TR/xslt-30/#element-result-document](https://www.w3.org/TR/xslt-30/#element-result-document). |

### Configuring an XSLT Processor for Generating Output

This section explains how to configure an XSLT processor and extensions for such a processor in Oxygen XML Editor.

### Supported XSLT Processors

Oxygen XML Editor includes the following XSLT processors:

• **Xalan 2.7.2** - [Xalan-Java](https://xml.apache.org/xalan-j/) is an XSLT processor for transforming XML documents into HTML, text, or other XML document types. It implements XSL Transformations (XSLT) Version 1.0 and XML Path Language (XPath) Version 1.0.

• **Saxon 6.5.5** - [Saxon 6.5.5](https://xml.apache.org/xalan-j/) is an XSLT processor that implements the Version 1.0 XSLT and XPath with a number of powerful extensions. This version of Saxon also includes many of the new features that
were first defined in the XSLT 1.1 working draft, but for conformance and portability reasons these are not available if the stylesheet header specifies `version="1.0"`.

- **Saxon 10.6 Home Edition (HE), Professional Edition (PE)** - Saxon-HE/PE implements the basic conformance level for XSLT 2.0 / 3.0 and XQuery 1.0. The term basic XSLT 2.0 / 3.0 processor is defined in the draft XSLT 2.0 / 3.0 specifications. It is a conformance level that requires support for all features of the language other than those that involve schema processing. The HE product remains open source, but removes some of the more advanced features that are present in Saxon-PE.

- **Saxon 10.6 Enterprise Edition (EE)** - Saxon EE is the schema-aware edition of Saxon and it is one of the built-in processors included in Oxygen XML Editor. Saxon EE includes an XML Schema processor, and schema-aware XSLT, XQuery, and XPath processors.

The validation in schema aware transformations is done according to the XML Schema 1.0 or 1.1. This can be configured in Preferences (on page 238).

---

**Note:**

Oxygen XML Editor implements a Saxon framework that allows you to create Saxon configuration files. Two templates are available: Saxon collection catalog and Saxon configuration. Both of these templates support content completion, element annotation, and attribute annotation.

---

**Note:**

Saxon can use the *ICU-J localization library* (`saxon9-icu.jar`) to add support for sorting and date/number formatting in a wide variety of languages. This library is not included in the Oxygen XML Editor installation kit. However, Saxon will use the default collation and localization support available in the currently used JRE. To enable this capability, follow these steps:

2. Unpack the downloaded archive.
3. Create a new XSLT transformation scenario (or edit an existing one). In the XSLT tab, click the Extensions button to open the list of additional libraries used by the transformation process.
4. Click Add and browse to the folder where you unpacked the downloaded archive and choose the `saxon9-icu.jar` file.

Note that the `saxon9-icu.jar` should NOT be added to the application library folder because it will conflict with another version of the ICU-J library that comes bundled with Oxygen XML Editor.

---

- **Saxon-CE (Client Edition)** is Saxonica's implementation of XSLT 2.0 for use on web browsers. Oxygen XML Editor provides support for editing stylesheets that contain Saxon-CE extension functions and instructions. This support improves the validation, content completion, and syntax highlighting.
Note:
Saxon-CE, being JavaScript-based, was designed to run inside a web browser. This means that you will use Oxygen XML Editor only for developing the Saxon-CE stylesheet, leaving the execution part to a web browser. See more details about executing such a stylesheet on Saxonica's website.

Note:
A specific template, named Saxon-CE stylesheet, is available in the New document wizard (on page 368).

• **Xsltproc (libxslt)** - Libxslt is the XSLT C library developed for the Gnome project. Libxslt is based on libxml2, the XML C library developed for the Gnome project. It also implements most of the EXSLT set of processor-portable extensions, functions, and some of Saxon's evaluate and expression extensions.

Oxygen XML Editor uses Libxslt through its command-line tool (Xsltproc). Depending on your operating system, you must download the Libxslt libraries on your machine from http://xmlsoft.org/XSLT/downloads.html and place them in a local folder. Then you need to update the PATH environmental variable to contain the parent folder where the xsltproc executable is located.

Tip:
As an example, a Windows installation of the Xsltproc engine would follow these steps:
2. Unzip all of them into the same folder of your choice.
3. Edit the PATH environment variable and add the bin folder for all four archives:

   ```
   Edit environment variable
   D:\apache-maven-3.1.1\bin
   D:\Python27
   %PATH%
   C:\Users\r\Desktop\abc\libxslt-1.1.26.win32\bin
   C:\Users\r\Desktop\abc\libxml2-2.7.8.win32\bin
   C:\Users\r\Desktop\abc\iconv-1.9.2.win32\bin
   C:\Users\r\Desktop\abc\zlib-1.2.5\bin
   ```

4. Restart Oxygen XML Editor.
Result: You can now use the `xsltproc` processor as an XSLT engine in the XSLT transformation scenario.

Note: The Xsltproc processor can be configured from the XSLTPROC options page (on page 250).

CAUTION:
There is a known problem where file paths that contain spaces are not handled correctly in the LIBXML processor. For example, the built-in XML Catalog (on page 3225) files of the built-in document types (DocBook, TEI, DITA, etc.) are not handled properly by LIBXML if Oxygen XML Editor is installed in the default location on Windows (C:\Program Files). This is because the built-in XML catalog files are stored in the {OXYGEN_INSTALL_DIR}/frameworks subdirectory of the installation directory, and in this case it contains a space character.

- **MSXML 4.0 (Legacy)** - MSXML 4.0 is available only on Windows platforms. It can be used for transformation (on page 1464) and validation of XSLT stylesheets (on page 889).

Oxygen XML Editor uses the Microsoft XML parser through its command-line tool `msxsl.exe`.

Since `msxsl.exe` is only a wrapper, Microsoft Core XML Services (MSXML) must be installed on the computer. Otherwise, you will get a corresponding warning. You can get the latest Microsoft XML parser from Microsoft's web-site.

- **MSXML .NET (Legacy)** - MSXML .NET is available only on Windows platforms. It can be used for transformation (on page 1464) and validation of XSLT stylesheets (on page 889).

Oxygen XML Editor performs XSLT transformations and validations using the .NET Framework XSLT implementation (`System.Xml.Xsl.XslTransform` class) through the `nxslt` command-line utility. The `nxslt` version included in Oxygen XML Editor is 1.6.

You should have the .NET Framework version 1.0 already installed on your system. Otherwise, you will get the following warning: **MSXML.NET requires .NET Framework version 1.0 to be installed. Exit code: 128.**

You can get the .NET Framework version 1.0 from the Microsoft website.

- **.NET 1.0 (Legacy)** - A transformer based on the `System.Xml` 1.0 library available in the .NET 1.0 and .NET 1.1 frameworks from Microsoft. It is available only on Windows.

You should have the .NET Framework version 1.0 or 1.1 already installed on your system. Otherwise, you will get the following warning: **MSXML.NET requires .NET Framework version 1.0 to be installed. Exit code: 128.**

You can get the .NET Framework version 1.0 from the Microsoft website.
• .NET 2.0 (Legacy) - A transformer based on the System.Xml 2.0 library available in the .NET 2.0 Framework from Microsoft. It is available only on Windows.

You should have the .NET Framework version 2.0 already installed on your system. Otherwise, you will get the following warning: MSXML.NET requires .NET Framework version 2.0 to be installed. Exit code: 128.

You can get the .NET Framework version 2.0 from the Microsoft website.

For information about configuring the XSLT preferences, see the XSLT options (on page 245) section.

Configuring Custom XSLT Processors

Oxygen XML Editor allows you to configure custom processors to be used for running XSLT and XQuery transformations.

To add a new custom processor, follow these steps:

1. Open the Preferences dialog box (Options > Preferences) (on page 124) and go to XML > XSLT-XQuery > Custom Engines.
2. Click the New button at the bottom of the dialog box.
3. Configure the parameters for the custom engine (on page 259).
4. Click OK.

Note:
You can not use these custom engines in the Debugger perspective (on page 2102).

The output messages of a custom processor are displayed in an output view at the bottom of the application window. If an output message follows the format of an Oxygen XML Editor linked message (on page 784), clicking it highlights the location of the message in an editor panel containing the file referenced in the message.

Related Information:
Custom Engines Preferences (on page 259)

Configuring the XSLT Processor Extensions Paths

The Xalan and Saxon processors support the use of extension elements and extension functions. Unlike a literal result element, which the stylesheet simply transfers to the result tree, an extension element performs an action. The extension is usually used because the XSLT stylesheet fails in providing adequate functions for accomplishing a more complex task.

For more information about how to use extensions, see the following links:
• Xalan - http://xml.apache.org/xalan-j/extensions.html
• Saxon 6.5.5 - http://saxon.sourceforge.net/saxon6.5.5/extensions.html
• Saxon 10.6 - http://www.saxonica.com/documentation9.5/index.html#extensibility

To set an XSLT processor extension (a directory or a jar file), use the Extensions button (on page 1466) in the Edit scenario dialog box.

XML Transformation with XQuery

This type of transformation specifies the transform parameters and location of an XQuery file that is applied to the edited XML document.

Use the XML transformation with XQuery scenario to apply a transformation to have an XQuery file query an XML file for the output results.

To create an XML transformation with XQuery scenario, use one of the following methods:

• Use the Configure Transformation Scenario(s) (Ctrl + Shift + C (Command + Shift + C on macOS)) action from the toolbar or the Document > Transformation menu. Then click the New button and select XML transformation with XQuery.
• Go to Window > Show View and select Transformation Scenarios to display this view (on page 1555). Click the New Scenario drop-down menu button and select XML transformation with XQuery.

Both methods open the New Scenario dialog box.

The upper part of the dialog box allows you to specify the Name of the transformation scenario and the following Storage options:

• Project Options (on page 3222) - The scenario is stored in the project file and can be shared with other users. For example, if your project is saved on a source versioning/sharing system (CVS, SVN, Source Safe, etc.) or a shared folder, your team can use the scenarios that you store in the project file.
• Global Options (on page 3219) - The scenario is saved in the global options that are stored in the user home directory and is not accessible to other users.

The lower part of the dialog box contains several tabs that allow you to configure the options that control the transformation.

XQuery Tab

When you create a new transformation scenario (on page 1464) or edit an existing one (on page 1545), a configuration dialog box is displayed that allows you to customize the transformation with various options in several tabs.

The XQuery tab contains the following options:

XML URL
Specifies the source XML file. You can specify the path by using the text field, its history drop-down, the Insert Editor Variables (on page 323) button, or the browsing actions in the Browse drop-down list. You can also use the Open in editor button to open the specified file in the editor panel. This URL is resolved through the catalog resolver. If the catalog does not have a mapping for the URL, then the file is used directly from its remote location.

**Note:**
If the transformer engine is Saxon 9.x and a custom URI resolver is configured in the advanced Saxon preferences page (on page 249), the XML input of the transformation is passed to that URI resolver.

**XQuery URL**

Specifies the source XQuery file to be used for the transformation. You can specify the path by using the text field, its history drop-down, the Insert Editor Variables (on page 323) button, or the browsing actions in the Browse drop-down list. You can also use the Open in editor button to open the specified file in the editor panel. This URL is resolved through the catalog resolver. If the catalog does not have a mapping for the URL, the file is used directly from its remote location.

**Transformer**

This drop-down menu presents all the transformation engines available to Oxygen XML Editor for performing a transformation. These include the built-in engines and the external engines defined in the Custom Engines preferences page (on page 259). The engine you choose is used as the default transformation engine. Also, if an XSLT or XQuery document does not have an associated validation scenario, this transformation engine is used in the validation process (if it provides validation support).

**Advanced options**

Allows you to configure the advanced options of the Saxon HE/PE/EE engine (on page 1483) for the current transformation scenario. To configure the same options globally, go to the Saxon-HE/PE/EE preferences page (on page 247). For the current transformation scenario, these Advanced options override the options configured in that preferences page.

**Parameters**

Opens the Configure parameters dialog box (on page 1482) for configuring the XQuery parameters. You can use the buttons in this dialog box to add, edit, or remove parameters. If the XQuery transformation engine is custom-defined, you cannot use this dialog box to set parameters. Instead, you can copy all parameters from the dialog box using contextual menu actions and edit the custom XQuery engine to include the necessary parameters in the command line that starts the transformation process.

**Extensions**

Opens a dialog box for configuring the XQuery extension JARS or classes (on page 1483) that define extension Java functions or extension XSLT elements used in the transformation.
XQuery Parameters

The global parameters of the XQuery file used in a transformation scenario can be configured by using the Parameters button in the XQuery tab.

The resulting dialog box includes a table that displays all the parameters of the current XQuery file, along with their descriptions and current values. You can also add, edit, and remove parameters, and use the Filter text box to search for a specific term in the entire parameters collection. Note that edited parameters are displayed with their name in bold.

If the XPath column is selected, the parameter value is evaluated as an XPath expression before starting the XQuery transformation.

Example:

For example, you can use expressions such as:

```
  doc('test.xml')//entry
  //person[@atr='val']
```

Note:

1. The `doc` function solves the argument relative to the XQuery file location. You can use full paths or editor variables (on page 323) (such as `$cfdu` [current file directory]) to specify other locations: `doc($cfdu/test.xml')//`

2. Only XPath functions are allowed.

Below the table, the following actions are available for managing the parameters:

**New**

Opens the Add Parameter dialog box that allows you to add a new parameter to the list. An editor variable (on page 323) can be inserted in the text box using the Insert Editor Variables button. If the Evaluate as XPath option is selected, the parameter will be evaluated as an XPath expression.

**Edit**

Opens the Edit Parameter dialog box that allows you to edit the selected parameter. An editor variable (on page 323) can be inserted in the text box using the Insert Editor Variables button. If the Evaluate as XPath option is selected, the parameter will be evaluated as an XPath expression.

**Unset**

Resets the selected parameter to its default value. Available only for edited parameters with set values.

**Delete**
Removes the selected parameter from the list. It is available only for new parameters that have been added to the list.

The bottom panel presents the following:

- The default value of the parameter selected in the table.
- A description of the parameter, if available.
- The system ID of the stylesheet that declares it.

Related Information:
Editor Variables (on page 323)

XQuery Extensions

The Extensions button is used to specify the JAR (on page 3220) and classes that contain extension functions called from the XQuery file of the current transformation scenario. You can use the Add, Edit, and Remove buttons to configure the extensions.

An extension function called from the XQuery file of the current transformation scenario will be searched, in the specified extensions, in the order displayed in this dialog box. To change the order of the items, select the item to be moved and click the Move up or Move down buttons.

Advanced Saxon HE/PE/EE XQuery Transformation Options

The XQuery transformation scenario allows you to configure advanced options that are specific for the Saxon HE (Home Edition), PE (Professional Edition), and EE (Enterprise Edition) engines. They are the same options as those in the Saxon HE/PE/EE preferences page (on page 254) but they are configured as a specific set of transformation options for each transformation scenario, while the values set in the preferences page apply as global options. The advanced options configured in a transformation scenario override the global options (on page 3219) defined in the preferences page.

Saxon-HE/PE/EE Options

The advanced options for Saxon 10.6 Home Edition (HE), Professional Edition (PE), and Enterprise Edition (EE) are as follows:

Use a configuration file ("-config")

Sets a Saxon 10.6 configuration file that is used for XQuery transformation and validation scenarios.

Enable Optimizations ("-opt")

This option is selected by default, which means that optimization is enabled. If not selected, the optimization is suppressed, which is helpful when reducing the compiling time is important, optimization conflicts with debugging, or optimization causes extension functions with side-effects to behave unpredictably.
Use linked tree model ("-tree:linked")

This option activates the linked tree model.

Strip whitespaces ("-strip")

Specifies how the strip whitespaces operation is handled. You can choose one of the following values:

- **All** ("all") - Strips all whitespace text nodes from source documents before any further processing, regardless of any @xml:space attributes in the source document.
- **Ignore** ("ignorable") - Strips all ignorable whitespace text nodes from source documents before any further processing, regardless of any @xml:space attributes in the source document. Whitespace text nodes are ignorable if they appear in elements defined in the DTD or schema as having element-only content.
- **None** ("none") - Strips no whitespace before further processing.

Enable profiling ("-TP")

If selected, profiling of the execution time in a query is enabled. The corresponding text field is used to specify the path to the output file where the profiling information will be saved. As long as the option is selected, and the output file specified, it will gather timed tracing information and create a profile report to the specified file.

**Note:**
The profiling support works only if the Present as a sequence transformation option (on page 1486) is not set.

Saxon-PE/EE Options

The following advanced options are specific for Saxon 10.6 Professional Edition (PE) and Enterprise Edition (EE) only:

Allow calls on extension functions ("-ext")

If selected, calls on external functions are allowed. Selecting this option is not recommended in an environment where untrusted stylesheets may be executed. It also disables user-defined extension elements and the writing of multiple output files, both of which carry similar security risks.

Saxon-EE Options

The advanced options that are specific for Saxon 10.6 Enterprise Edition (EE) are as follows:

Validation of the source file ("-val")

Requests schema-based validation of the source file and of any files read using document() or similar functions. It can have the following values:
• **Schema validation** ("strict") - This mode requires an XML Schema and allows for parsing the source documents with strict schema-validation enabled.

• **Lax schema validation** ("lax") - If an XML Schema is provided, this mode allows for parsing the source documents with schema-validation enabled but the validation will not fail if, for example, element declarations are not found.

• **Disable schema validation** - This specifies that the source documents should be parsed with schema-validation disabled.

**Validation errors in the result tree treated as warnings ("-outval")**

Normally, if validation of result documents is requested, a validation error is fatal. Selecting this option causes such validation failures to be treated as warnings.

**Write comments for non-fatal validation errors of the result document**

The validation messages for non-fatal errors are written (wherever possible) as a comment in the result document itself.

**Enable XQuery update ("-update:(on|off)")**

This option controls whether or not XQuery update syntax is accepted. The default value is off.

**Backup files updated by XQuery ("-backup:(on|off)")**

If selected, backup versions for any XML files updated with an XQuery Update are generated. This option is available when the **Enable XQuery update** option is selected.

**Other Options**

**Initializer class**

Equivalent to the `-init` Saxon command-line argument. The value is the name of a user-supplied class that implements the `net.sf.saxon.lib.Initializer` interface. This initializer is called during the initialization process, and may be used to set any options required on the configuration programmatically. It is particularly useful for tasks such as registering extension functions, collations, or external object models, especially in Saxon-HE where the option cannot be set via a configuration file. Saxon only calls the initializer when running from the command line, but the same code may be invoked to perform initialization when running user application code.

**FO Processor Tab (XQuery Transformations)**

When you create a new transformation scenario (on page 1464) or edit an existing one (on page 1545), a configuration dialog box is displayed that allows you to customize the transformation with various options in several tabs.

The **FO Processor** tab contains the following options:

**Perform FO Processing**
Specifies whether or not an FO processor is applied (either the built-in Apache FOP engine or an external engine defined in Preferences) during the transformation.

**Input**

Choose between the following options to specify which input file to use:

- **XQuery result as input** - The FO processor is applied to the result of the XQuery transformation that is defined in the XQuery tab.
- **XML URL as input** - The FO processor is applied to the input XML file.

**Method**

The output format of the FO processing. The available options depend on the selected processor type.

**Processor**

Specifies the FO processor to be used for the transformation. It can be the built-in Apache FOP processor or an external processor (on page 261).

**Output Tab (XQuery Transformations)**

When you create a new transformation scenario (on page 1464) or edit an existing one (on page 1545), a configuration dialog box is displayed that allows you to customize the transformation with various options in several tabs.

The **Output** tab contains the following options:

**Present as a sequence**

Selecting this option will reduce the time necessary to fetch the full results, as it will only fetch the first chunk of the results.

**Prompt for file**

At the end of the transformation, a file browser dialog box is displayed for specifying the path and name of the file that stores the transformation result.

**Save As**

The path of the file where the result of the transformation is stored. You can specify the path by using the text field, the Insert Editor Variables (on page 323) button, or the Browse button.

**Open in Browser/System Application**

If selected, Oxygen XML Editor automatically opens the result of the transformation in a system application associated with the file type of the result (for example, in Windows PDF files are often opened in Acrobat Reader).
Note:
To set the web browser that is used for displaying HTML/XHTML pages, go to Options > Preferences > Global, and set it in the Default Internet browser field.

- **Output file** - When Open in Browser/System Application is selected, you can use this button to automatically open the default output file at the end of the transformation.
- **Other location** - When Open in Browser/System Application is selected, you can use this option to open the file specified in this field at the end of the transformation. You can specify the path by using the text field, the Insert Editor Variables (on page 323) button, or the Browse button.

Open in editor

When this is option is selected, at the end of the transformation, the default output file is opened in a new editor panel with the appropriate built-in editor type (for example, if the result is an XML file it is opened in the built-in XML editor, or if it is an XSL-FO file it is opened with the built-in FO editor).

Show in results view as

You can choose to view the results in one of the following:

- **XML** - If this is selected, Oxygen XML Editor displays the transformation result in an XML viewer panel at the bottom of the application window with syntax highlighting (on page 224).
- **SVG** - If this is selected, Oxygen XML Editor displays the transformation result in an integrated SVG viewer in the Results panel (on page 1254) at the bottom of the application window and renders the result as an SVG image.
- **XHTML** - This option is only available if Open in Browser/System Application is not selected. If selected, Oxygen XML Editor displays the transformation result in a built-in XHTML browser panel at the bottom of the application window.

Important:
When transforming very large documents, you should be aware that selecting this option may result in very long processing times. This drawback is due to the built-in Java XHTML browser implementation. To avoid delays for large documents, if you want to see the XHTML result of the transformation, you should use an external browser by selecting the Open in Browser/System Application option instead.
Image URLs are relative to - If Show in results view as XHTML is selected, this option specifies the path used to resolve image paths contained in the transformation result. You can specify the path by using the text field, the Insert Editor Variables (on page 323) button, or the Browse button.

XML to PDF Transformation with CSS

This type of transformation uses the Oxygen PDF Chemistry processing engine to obtain PDF output by applying CSS styling to the edited XML document. This scenario is useful for those who are familiar with CSS and want to obtain PDF output as its final form.

To create an XML to PDF transformation with CSS scenario, use one of the following methods:

- Use the Configure Transformation Scenario(s) (Ctrl + Shift + C (Command + Shift + C on macOS)) action from the toolbar or the Document > Transformation menu. Then click the New button and select XML to PDF transformation with CSS.
- Go to Window > Show View and select Transformation Scenarios to display this view (on page 1555). Click the New Scenario drop-down menu button and select XML to PDF transformation with CSS.

Both methods open the New Scenario dialog box.

The upper part of the dialog box allows you to specify the Name of the transformation scenario and the following Storage options:

- Project Options (on page 3222) - The scenario is stored in the project file and can be shared with other users. For example, if your project is saved on a source versioning/sharing system (CVS, SVN, Source Safe, etc.) or a shared folder, your team can use the scenarios that you store in the project file.
- Global Options (on page 3219) - The scenario is saved in the global options that are stored in the user home directory and is not accessible to other users.

The lower part of the dialog box contains several tabs that allow you to configure the options that control the transformation.

For more information about the Oxygen PDF Chemistry processing engine and numerous tips for customizing the output, see the Oxygen Chemistry User Guide.

CSS Tab (XML to PDF Transformation with CSS)

When you create a new transformation scenario (on page 1464) or edit an existing one (on page 1545), a configuration dialog box is displayed that allows you to customize the transformation with various options in several tabs.

The CSS tab contains the following options:

XML URL
Specifies the source XML file. You can specify the path by using the text field, its history drop-down, the Insert Editor Variables button, or the browsing actions in the Browse drop-down list. You can also use the Open in editor button to open the specified file in the editor panel. This URL is resolved through the catalog resolver. If the catalog does not have a mapping for the URL, then the file is used directly from its remote location.

**CSS URL**

Optionally, you can use this option to specify the location of a custom CSS file to be applied to the transformation. If this option is left blank, only the CSS referenced directly from the document will be applied. You can specify the path by using the text field, its history drop-down, the Insert Editor Variables button, or the browsing actions in the Browse drop-down list. You can also use the Open in editor button to open the specified file in the editor panel.

**Apply CSS stylesheets set in the current framework**

If selected, CSS stylesheets that are specified in the framework (in the Document Type configuration CSS subtab) are applied to the transformation in addition to any CSS referenced directly in the document or specified in the CSS URL field.

**Note:**

If CSS files are specified in multiple ways, the transformation applies the CSS in the following order (from lowest priority to highest):

- CSS files that are specified in the framework (in the Document Type configuration CSS subtab).
- CSS files referenced directly in the document.
- CSS files specified in the CSS URL field.

**Processor options link**

Opens the CSS-based Processors preferences page where you can configure some options for generating PDF output.

**Output Tab (XML to PDF Transformation with CSS)**

When you create a new transformation scenario or edit an existing one, a configuration dialog box is displayed that allows you to customize the transformation with various options in several tabs.

The Output tab contains the following options:

**Output File section**

Save As
The path of the file where the result of the transformation is stored. You can specify the path by using the text field, the Insert Editor Variables (on page 323) button, or the Browse button.

Open in Browser/System Application

If selected, Oxygen XML Editor automatically opens the result of the transformation in a system application associated with the PDF file type (for example, in Windows PDF files are often opened in Acrobat Reader).

Debugging section

Dump the intermediate annotated XML

Select this option to include (dump) the intermediate, annotated XML file in the same location as the output file. This can be used for debugging purposes.

Dump the FO file

Select this option to include (dump) the FO file (before it is converted to PDF) in the same location as the output file. This can be used for debugging purposes.

Console options link

Opens the CSS-based Processors preferences page (on page 264) where you can configure some options for generating PDF output.

DITA-OT Transformation

This type of transformation specifies the parameters for an Ant transformation that executes a DITA-OT build script. Oxygen XML Editor includes a built-in version of Ant and a built-in version of DITA-OT, but other versions can be set in the scenario.

To create a DITA-OT Transformation scenario, use one of the following methods:

- Use the Configure Transformation Scenario(s) (Ctrl + Shift + C (Command + Shift + C on macOS)) action from the DITA Maps Manager toolbar, main toolbar, or the Document > Transformation menu. Then click the New button and select DITA-OT Transformation.
- Go to Window > Show View and select Transformation Scenarios to display this view (on page 1555). Click the New Scenario drop-down menu button and select DITA-OT Transformation.

Both methods open the DITA Transformation Type dialog box that presents the list of possible outputs.
Figure 470. DITA Transformation Type Dialog Box

Select the desired type of output and click OK. This opens the New Scenario dialog box.

The upper part of the dialog box allows you to specify the Name of the transformation scenario and the following Storage options:

- **Project Options (on page 3222)** - The scenario is stored in the project file and can be shared with other users. For example, if your project is saved on a source versioning/sharing system (CVS, SVN, Source Safe, etc.) or a shared folder, your team can use the scenarios that you store in the project file.
- **Global Options (on page 3219)** - The scenario is saved in the global options that are stored in the user home directory and is not accessible to other users.

The lower part of the dialog box contains several tabs that allow you to configure the options that control the transformation.

**Skins Tab (DITA-OT Transformations)**

When you create a new transformation scenario (on page 1464) or edit an existing one (on page 1545), a configuration dialog box is displayed that allows you to customize the transformation with various options in several tabs.

The Skins tab is available for DITA-OT transformations with the WebHelp Classic output type and it provides a set of built-in skins that you can use as a base for your WebHelp system output.
A skin is a collection of CSS properties that can alter the look of the output by changing colors, font types, borders, margins, and paddings. This allows you to rapidly adapt the look and feel of your output.

The Skins tab includes the following sections:

**Built-in Skins**

This section presents the built-in skins that are included in Oxygen XML Editor. The built-in skins cover a wide range of chromatic themes, ranging from a very light one to a high-contrast variant. To see how the skin looks when applied on a sample documentation project that is stored on the Oxygen XML Editor website, click the **Online preview** link.

**Custom Skins**

You can use this section to customize the look of the output.

**CSS File**

You can set this field to point to a custom CSS stylesheet or customized skin. A custom CSS file will overwrite a skin selection.
this parameter take precedence over the properties declared in the skin set in the Skins tab.

Create custom skin

Use this link to open the WebHelp Skin Builder (on page 1742) tool.

Templates Tab (DITA-OT Transformations)

When you create a new transformation scenario (on page 1464) or edit an existing one (on page 1545), a configuration dialog box is displayed that allows you to customize the transformation with various options in several tabs.

The Templates tab is available for DITA-OT transformations with WebHelp Responsive or PDF - based on HTML5 & CSS output types and it provides a set of built-in publishing templates (on page 1603). You can use one of them to publish your documentation or as a starting point for a new publishing template.

Filtering and Previewing Templates

You can click on the tags at the top of the pane to filter the templates and narrow your search. Each built-in template also includes an Online preview icon in the bottom-right corner that opens a webpage in your default browser providing a sample of how the main page will look when that particular template is used to generate the output.

Built-in Templates Locations

Oxygen XML Editor scans the following locations to find the built-in templates to display in the dialog box:
• **WebHelp Responsive Templates** - All built-in WebHelp Responsive publishing templates are stored in the following directory: `DITA-OT-DIR/plugins/com.oxygenxml.webhelp-responsive/templates`.

• **PDF - based on HTML5 & CSS** - All built-in PDF publishing templates are stored in the following directories:
  - `DITA-OT-DIR/plugins/com.oxygenxml.pdf.css/templates`
  - `DITA-OT-DIR/plugins/com.oxygenxml.webhelp-responsive/templates`

### Custom Templates Locations

Oxygen XML Editor scans the locations specified in the [DITA > Publishing preferences page](on page 275) to find custom templates to display in the dialog box. You can access that preferences page directly from the Template tab by clicking on the **Configure Publishing Templates Gallery** link.

### Selecting Custom Templates

Once you are finished configuring your template, you can click the **Choose Custom Publishing Template** link to select your template.

You can also **add your custom templates** (on page 1644) to the list of templates displayed in the Templates tab. To do this, store them in a directory, then click the **Configure Publishing Templates Gallery** link to open the [DITA > Publishing preferences page](on page 275) where you can add that directory to the list. All the templates contained in your template directory will be displayed in the preview pane along with all the built-in templates.

### Save Template As Button

You can use the **Save template as** button (at the bottom-left of the transformation dialog box) to export the currently selected template into a new template package that can be used as a starting point to [create your own custom template](on page 1791). Clicking this button will open a template package configuration dialog box (on page 3109) that contains some options and displays the parameters that will be exported to your template package.

### Template Errors

When the Templates tab is opened, all templates (built-in and custom) are loaded and validated. Specifically, certain elements in the template descriptor file are checked for validity. If errors are encountered that prevents the template from loading, the following message will be displayed toward the bottom of the dialog box:

![Warning icon] Some templates could not be loaded. [More details]

If you click the More details link, a window will open with more information about the encountered error. For example, it might offer a hint that the element is missing from the expected descriptor file structure.
Also, if a template could be loaded, but certain elements could not be found in the descriptor file, a warning icon (⚠️) will be displayed on the template's image (in the Templates tab of the transformation dialog box). For example, this happens if a valid preview-image element cannot be found.

**Sharing Publishing Template**

To share a publishing template with others, following these steps:

1. Copy your template in a new folder.
2. Go to Options > Preferences > DITA > Publishing (on page 275) and add that new folder to the list.
3. Switch the option as the bottom of that preferences page to Project Options.
4. Share your project file (.xpr).

For more information about customizing publishing templates, watch our video demonstration:

https://www.youtube.com/embed/zNmXfKWXwO8

**Template Package Configuration Dialog Box**

The Save template as button (at the bottom-left of the transformation dialog box for WebHelp Responsive or PDF - based on HTML5 & CSS transformations) can be used to export the currently selected template into a new template package that can be used as a starting point to create your own custom template (on page 1791). The result will be a ZIP archive that contains a template descriptor file and other resources (such as CSS files) that were attached to the selected template.

Clicking the Save template as button opens a template package configuration dialog box contains the following options and components:

**Name**

Required field used to specify the name for the new template. This will become the text value of the `<name>` element in the template descriptor file. This information is displayed as the name of the template in the transformation scenario dialog box.

**Description**

Optional field used to specify a template description. This will become the text value of the `<description>` element in the template descriptor file. This information is displayed when the user hovers over the template in the transformation scenario dialog box.

**Parameter Table**
This table displays the parameters that will be exported. Only certain relevant parameters are exported. The parameters and their values will be inserted in the `<parameters>` section of the template descriptor file. If any of the parameter values point to a file path that references a template resource (such as CSS files, custom HTML fragments, images), those resources will automatically be copied to the new template package and their references will be changed accordingly.

**Note:**

Additional resources that are referenced in CSS files or other resources will not be copied to the new template package, so you will need to copy them manually and update their references in the template descriptor file.

**Include WebHelp Customization**

The same publishing template package can contain both a WebHelp Responsive and PDF customization and you can use the same template in both types of transformations (DITA Map WebHelp Responsive (on page 1435) or DITA Map to PDF - based on HTML5 & CSS (on page 1449)). This option specifies that the custom template will include a WebHelp Responsive customization.

**Include HTML Page Layout Files**

For WebHelp Responsive customizations, select this option if you want to copy the default HTML Page Layout Files (on page 1622) into your template package. They are helpful if you want to change the structure of the generated HTML pages.

**Include PDF Customization**

The same publishing template package can contain both a WebHelp Responsive and PDF customization and you can use the same template in both types of transformations (DITA Map WebHelp Responsive (on page 1435) or DITA Map to PDF - based on HTML5 & CSS (on page 1449)). This option specifies that the custom template will include a PDF customization.

**Save as**

Use this field to specify the name and path of the ZIP file where the template will be saved.
Figure 473. Template Package Configuration Dialog Box

Related Information:
- Publishing Templates (on page 1603)
- Publishing Template Package Contents for PDF Customizations (on page 1785)
- Publishing Template Package Contents for WebHelp Responsive Customizations (on page 1606)

**FO Processor Tab (DITA-OT Transformations)**

When you create a new transformation scenario (on page 1464) or edit an existing one (on page 1545), a configuration dialog box is displayed that allows you to customize the transformation with various options in several tabs.

The **FO Processor** tab is available for DITA-OT transformations with a **PDF** output type.

This tab allows you to select an FO Processor to be used for the transformation.
You can choose one of the following processors:

**Apache FOP**

The default processor that comes bundled with Oxygen XML Editor.

**XEP**

The RenderX XEP processor. If XEP is already installed, Oxygen XML Editor displays the detected installation path under the drop-down menu. XEP is considered installed if it was detected in one of the following sources:

- XEP was configured as an external FO Processor in the FO Processors option page (on page 261).
- The system property `com.oxygenxml.xep.location` was set to point to the XEP executable file for the platform (for example: `xep.bat` on Windows).
- XEP was installed in the `DITA-OT-DIR/plugins/org.dita.pdf2/lib` directory of the Oxygen XML Editor installation directory.

**Antenna House**

The Antenna House (AH Formatter) processor. If Antenna House is already installed, Oxygen XML Editor displays the detected installation path under the drop-down menu. Antenna House is considered installed if it was detected in one of the following sources:

- Environment variable set by Antenna House installation (the newest installation version will be used).
- Antenna House was added as an external FO Processor in the Oxygen XML Editor preferences pages.
To further customize the PDF output obtained from the Antenna House processor, follow these steps:

1. **Edit** the transformation scenario.
2. Open the **Parameters** tab *(on page 3112).*
3. Add the `env.AXF_OPT` parameter and point to the Antenna House configuration file.

**Related information**
- FO Processors Preferences *(on page 261)*
- XSL-FO (Apache FOP) Processor for Generating PDF Output *(on page 1526)*

**Parameters Tab (DITA-OT Transformations)**

When you **create a new transformation scenario** *(on page 1464)* or **edit an existing one** *(on page 1545)*, a configuration dialog box is displayed that allows you to customize the transformation with various options in several tabs.

The **Parameters** tab allows you to configure the parameters sent to the DITA-OT build file.

The table in this tab displays all the parameters that the DITA-OT documentation specifies as available for each chosen type of transformation (for example, XHTML or PDF), along with their description and current values. You can find more information about each parameter in the DITA-OT Documentation. You can also add, edit, and remove parameters, and you can use the text box to filter or search for a specific term in the entire parameters collection. Note that edited parameters are displayed with their name in bold.

Depending on the type of a parameter, its value can be one of the following:

- A simple text field for simple parameter values.
- A combo box with some predefined values.
- A file chooser and an editor variable *(on page 323)* selector to simplify setting a file path as the value of a parameter.

**Note:**
To input parameter values at runtime, use the **ask editor variable** *(on page 325)* in the **Value** column.

Below the table, the following actions are available for managing parameters:

**New**

Opens the **Add Parameter** dialog box that allows you to add a new parameter to the list. You can specify the **Value** of the parameter by using the **Insert Editor Variables** *(on page 323)* button or the **Browse** button. You can also use the **Open in editor** button to open the specified file in the editor panel.

**Unset**
Resets the selected parameter to its default value. Available only for edited parameters with set values.

**Edit**

Opens the **Edit Parameter** dialog box that allows you to change the value of the selected parameter or its description.

**Delete**

Removes the selected parameter from the list. It is available only for new parameters that have been added to the list.

**Parameters Contributed by an Oxygen Publishing Template**

Transformation parameters that are defined in an *Oxygen Publishing Template (on page 1783)* descriptor file are displayed in italics. After creating a publishing template (on page 1791) and adding it to the templates gallery (on page 1644), when you select the template in the **Templates** tab (on page 3106), the **Parameters** tab will automatically be updated to include the parameters defined in the template descriptor file.

**Related Information:**

- DITA Open Toolkit Documentation

**Feedback Tab (DITA-OT Transformations)**

When you create a new transformation scenario (on page 1464) or edit an existing one (on page 1545), a configuration dialog box is displayed that allows you to customize the transformation with various options in several tabs.

The **Feedback** tab is for those who want to provide a way for users to offer feedback and ask questions in the published output and it is available for the **DITA Map WebHelp Responsive** transformation type. To add a comments component in the output, you need to use **Oxygen Feedback** to create a site configuration for the website where your WebHelp output is published and use this **Feedback** tab to instruct the transformation to install the comments component at the bottom of each WebHelp page.

When you create a site configuration in the **Oxygen Feedback** administration interface, an HTML fragment is generated during the final step of the creation process. You need to click the **Edit** button at the bottom-right of this tab to open a dialog box where you will paste the generated HTML fragment. The HTML fragment can also be set in an **Oxygen Publishing Template (on page 1783)**, either as an HTML fragment extension point (on page 1613) or as a transformation parameter (on page 1612) (the `webhelp.fragment.feedback` parameter). If the fragment is specified in multiple places, the order of precedence (from highest to lowest) is:

- The fragment specified directly in the **Feedback** tab.
- The fragment specified in a publishing template as an HTML fragment extension point.
- The fragment specified in a publishing template as a transformation parameter.
Filters Tab (DITA Transformations)

When you create a new transformation scenario (on page 1464) or edit an existing one (on page 1545), a configuration dialog box is displayed that allows you to customize the transformation with various options in several tabs.

The **Filters** tab allows you to add filters to remove certain content elements from the generated output.

![Figure 475. Edit Filters Tab](image)

You can choose one of the following options to define filters:

**Use DITAVAL file**

If you already have a **DITAVAL** file associated with the **DITA map (on page 3218)**, you can specify the file to be used when filtering content. You can specify the path by using the text field, its history drop-down, the **Insert Editor Variables (on page 323)** button, or the browsing actions in the **Browse** drop-down list. You can find out more about constructing a **DITAVAL** file in the **DITA Documentation**.

**Note:**

If a filter file is specified in the `args.filter` parameter (in the **Parameters tab (on page 3112)**), the filters are combined (neither file takes precedence over the other).

**Use profiling condition set**
Sets the profiling condition set (on page 3142) that will be applied to your transformation.

Exclude from output all elements with any of the following attributes

By using the New, Edit, or Delete buttons at the bottom of the pane, you can configure a list of attributes (name and value) to exclude all elements that contain any of these attributes from the output.

Note:
The colors and styles of the profiled content (on page 3147) settings are used for rendering it in Author mode but are not applied in the output.

Advanced Tab (DITA-OT Transformations)

When you create a new transformation scenario (on page 1464) or edit an existing one (on page 1545), a configuration dialog box is displayed that allows you to customize the transformation with various options in several tabs.

The Advanced tab allows you to specify advanced options for the transformation scenario.
You can specify the following options:

**Prefer using the “dita” command**

When selected, Oxygen XML Editor will attempt to use the `dita.bat` executable script (`dita.sh` for macOS and Linux) that is bundled with DITA-OT to run the transformation. If not selected, the transformation will run as an ANT process. Also, when this option is selected, other options (Custom build file, Build target, Ant Home) become unavailable. This setting is checked by default in newly created DITA-OT transformation scenario.

**Note:**

Even when this option is selected, the `dita.bat` (`dita` for macOS and Linux) executable cannot be used in some cases. For example, if the DITA Map is published from a remote location or if the `fix.external.refs` parameter is enabled in the Parameters tab, the transformation is started as an ANT process instead of using the executable.

**Custom build file**
If you use a custom DITA-OT build file, you can specify the path to the customized build file. If empty, the *build.xml* file from the *dita.dir* parameter that is configured in the Parameters tab (on page 3112) is used. You can specify the path by using the text field, the Insert Editor Variables (on page 323) button, or the Browse button.

**Build target**

Optionally, you can specify a build target for the build file. If no target is specified, the default *init* target is used.

**Additional Ant arguments**

You can specify additional Ant-specific command-line arguments (such as `-diagnostics`).

**Ant Home**

You can choose between the default or custom Ant installation to run the transformation. The default path can be configured in the Ant preferences page (on page 265).

**Java Home**

You can choose between the default or custom Java installation to run the transformation. The default path is the Java installation that is used by Oxygen XML Editor.

**Note:**

It may be possible that the used Java version is incompatible with the DITA Open Toolkit engine. For example, DITA-OT 2.0 and newer requires Java 1.7 or newer. Thus, if you encounter related errors running the transformation, consider installing a Java VM that is supported by the DITA-OT publishing engine and using it in the Java Home text field.

**JVM Arguments**

This parameter allows you to set specific parameters for the Java Virtual Machine used by Ant. When performing a large transformation, you may want to increase the memory allocated to the Java Virtual Machine. This will help avoid Out of Memory error messages (*OutOfMemoryError*). For example, if it is set to `-Xmx2g`, the transformation process is allowed to use a maximum 2 gigabytes of memory. If you do not specify an `-Xmx` value in this field, by default the application will use a maximum of 512 megabytes when used with the 32-bit Java Virtual Machine and one gigabyte with the 64-bit Java Virtual Machine.

**Libraries**

By default, Oxygen XML Editor adds libraries (as high priority) that are not transformation-dependent and also patches for certain DITA Open Toolkit bugs. You can use this button to specify additional libraries (*JAR* (on page 3220) files or additional class paths) to be used by the transformer.
Output Tab (DITA-OT Transformations)

When you create a new transformation scenario (on page 1464) or edit an existing one (on page 1545), a configuration dialog box is displayed that allows you to customize the transformation with various options in several tabs.

The Output tab allows you to configure options that are related to the location where the output is generated.

You can specify the following parameters:

**Base directory**

All the relative paths that appear as values in parameters are considered relative to the base directory. The default value is the directory where the transformed map is located. You can specify the path by using the text field, the Insert Editor Variables (on page 323) button, or the Browse button.

**Temporary files directory**
This directory is used to store pre-processed temporary files until the final output is obtained. You can specify the path by using the text field, the Insert Editor Variables button, or the Browse button.

**Output directory**

The folder where the content of the final output is stored. You can specify the path by using the text field, the Insert Editor Variables button, or the Browse button.

**Note:**

If the DITA map or topic is opened from a remote location or a ZIP file, the parameters must specify absolute paths.

**Open in Browser/System Application**

If selected, Oxygen XML Editor automatically opens the result of the transformation in a system application associated with the file type of the result (for example, in Windows PDF files are often opened in Acrobat Reader).

**Note:**

To set the web browser that is used for displaying HTML/XHTML pages, go to Options > Preferences > Global, and set it in the Default Internet browser field.

- **Output file** - When Open in Browser/System Application is selected, you can use this button to automatically open the default output file at the end of the transformation.
- **Other location** - When Open in Browser/System Application is selected, you can use this option to open the file specified in this field at the end of the transformation. You can specify the path by using the text field, the Insert Editor Variables button, or the Browse button.

**Open in editor**

When this is option is selected, at the end of the transformation, the default output file is opened in a new editor panel with the appropriate built-in editor type (for example, if the result is an XML file it is opened in the built-in XML editor, or if it is an XSL-FO file it is opened with the built-in FO editor).

At the bottom of the pane there is a link to the Console options preferences page that contains options to control the display of the console output received from the publishing engine.

**Ant Transformation**

This type of transformation allows you to configure the options and parameters of an Ant build script.
An Ant transformation scenario is usually associated with an Ant build script. Oxygen XML Editor runs an Ant transformation scenario as an external process that executes the Ant build script with the built-in Ant distribution (Apache Ant (on page 3216) version 1.9.8) that is included with the application, or optionally with a custom Ant distribution configured in the scenario.

**Tip:**
Certain Ant tasks require additional JAR libraries (for example, Ant mail tasks). The additional libraries can be added by editing the Ant transformation scenario, and in the Output tab, click the Libraries button (on page 1507) in the bottom right corner. This opens a dialog box where you can add JAR libraries. For a list of library dependencies, see https://ant.apache.org/manual/install.html#librarydependencies.

To create an Ant transformation scenario, use one of the following methods:

- Use the Configure Transformation Scenario(s) (Ctrl + Shift + C (Command + Shift + C on macOS)) action from the toolbar or the Document > Transformation menu. Then click the New button and select ANT transformation.
- Go to Window > Show View and select Transformation Scenarios to display this view (on page 1555). Click the New Scenario drop-down menu button and select ANT transformation.

Both methods open the transformation configuration dialog box.

The upper part of the dialog box allows you to specify the Name of the transformation scenario and the following Storage options:

- **Project Options** (on page 3222) - The scenario is stored in the project file and can be shared with other users. For example, if your project is saved on a source versioning/sharing system (CVS, SVN, Source Safe, etc.) or a shared folder, your team can use the scenarios that you store in the project file.
- **Global Options** (on page 3219) - The scenario is saved in the global options that are stored in the user home directory and is not accessible to other users.

The lower part of the dialog box contains several tabs that allow you to configure the options that control the transformation.

**Related Information:**
Transforming Ant Build Files (on page 936)

**Options Tab (Ant Transformations)**

When you create a new transformation scenario (on page 1464) or edit an existing one (on page 1545), a configuration dialog box is displayed that allows you to customize the transformation with various options in several tabs.

The Options tab allows you to specify the following options:
Working directory

The path of the current directory of the Ant external process. You can specify the path by using the text field, the ➔ Insert Editor Variables (on page 323) button, or the ➔ Browse button.

Build file

The Ant script file that is the input of the Ant external process. You can specify the path by using the text field, the ➔ Insert Editor Variables (on page 323) button, or the ➔ Browse button.

Build target

Optionally, you can specify a build target for the Ant script file. If no target is specified, the Ant target that is specified as the default in the Ant script file is used.

Additional Ant arguments

You can specify additional Ant-specific command-line arguments (such as -diagnostics).

Ant Home

You can choose between the default or custom Ant installation to run the transformation. The default path can be configured in the Ant preferences page (on page 265).

Java Home

You can choose between the default or custom Java installation to run the transformation. The default path is the Java installation that is used by Oxygen XML Editor.

JVM Arguments

This parameter allows you to set specific parameters for the Java Virtual Machine used by Ant. When performing a large transformation, you may want to increase the memory allocated to the Java Virtual Machine. This will help avoid Out of Memory error messages (OutOfMemoryError). For example, if it is set to -Xmx2g, the transformation process is allowed to use a maximum 2 gigabytes of memory. If you do not specify an -Xmx value in this field, by default the application will use a maximum of 512 megabytes when used with the 32-bit Java Virtual Machine and one gigabyte with the 64-bit Java Virtual Machine.

Libraries

By default, Oxygen XML Editor adds libraries (as high priority) that are not transformation-dependent and also patches for certain DITA Open Toolkit bugs. You can use this button to specify additional libraries (JAR (on page 3220) files or additional class paths) to be used by the transformer.

Tip:

You can specify the path to the additional libraries using wildcards (for example, 

`${oxygenHome}/lib/*.*.jar`.


Parameters Tab (Ant Transformations)

When you create a new transformation scenario (on page 1464) or edit an existing one (on page 1545), a configuration dialog box is displayed that allows you to customize the transformation with various options in several tabs.

The Parameters tab allows you to configure the parameters that are accessible as Ant properties in the Ant build script.

The table displays all the parameters that are available in the Ant build script, along with their description and current values. You can also add, edit, and remove parameters, and use the Filter text box to search for a specific term in the entire parameters collection. Note that edited parameters are displayed with their name in bold.

Depending on the type of a parameter, its value can be one of the following:

- A simple text field for simple parameter values.
- A combo box with some predefined values.
- A file chooser and an editor variable (on page 323) selector to simplify setting a file path as the value of a parameter.

Note:
To input parameter values at runtime, use the ask editor variable (on page 325) in the Value column.

Below the table, the following actions are available for managing parameters:

New
Opens the Add Parameter dialog box that allows you to add a new parameter to the list. You can specify the Value of the parameter by using the Insert Editor Variables (on page 323) button or the Browse button. You can also use the Open in editor button to open the specified file in the editor panel.

Edit
Opens the Edit Parameter dialog box that allows you to change the value of the selected parameter or its description.

Delete
Removes the selected parameter from the list. It is available only for new parameters that have been added to the list.

These parameters are also available for the built-in validation processor and the Content Completion Assistant (on page 3217).

Related Information:
Content Completion in Ant Build Files (on page 937)
Output Tab (Ant Transformations)

When you create a new transformation scenario (on page 1464) or edit an existing one (on page 1545), a configuration dialog box is displayed that allows you to customize the transformation with various options in several tabs.

The **Output** tab contains the following options:

**Open**

Allows you to specify the file to open automatically when the transformation is finished. This is usually the output file of the Ant process. You can specify the path by using the text field, the ![Insert Editor Variables (on page 323)](image) button, or the ![Browse](image) button.

- **In System Application** - The file specified in the **Open** text box is opened in the system application that is set in the operating system as the default application for that type of file (for example, in Windows PDF files are often opened in Acrobat Reader).
- **In Editor** - The file specified in the **Open** text box is opened in a new editor panel with the appropriate built-in editor type (for example, if the result is an XML file it is opened in the built-in XML editor).

**Show console output**

Allows you to specify when to display the console output log in the message panel at the bottom of the editor. The following options are available:

- **When build fails** - Displays the console output log only if the build fails.
- **Always** - Displays the console output log, regardless of whether or not the build fails.

**XSLT Transformation**

This type of transformation specifies the parameters and location of an XML document that the edited XSLT stylesheet is applied on. This scenario is useful when you develop an XSLT document and the XML document is in its final form.

To create an **XSLT transformation** scenario, use one of the following methods:

- Use the ![Configure Transformation Scenario(s) (Ctrl + Shift + C (Command + Shift + C on macOS))](image) action from the toolbar or the **Document > Transformation** menu. Then click the **New** button and select **XSLT transformation**.
- Go to **Window > Show View** and select ![Transformation Scenarios](image) to display this view (on page 1555). Click the ![New Scenario](image) drop-down menu button and select **XSLT transformation**.

Both methods open the **New Scenario** dialog box.

The upper part of the dialog box allows you to specify the **Name** of the transformation scenario and the following **Storage** options:
• **Project Options (on page 3222)** - The scenario is stored in the project file and can be shared with other users. For example, if your project is saved on a source versioning/sharing system (CVS, SVN, Source Safe, etc.) or a shared folder, your team can use the scenarios that you store in the project file.

• **Global Options (on page 3219)** - The scenario is saved in the global options that are stored in the user home directory and is not accessible to other users.

The lower part of the dialog box contains several tabs that allow you to configure the options that control the transformation.

**XSLT Tab**

When you create a new transformation scenario (on page 1464) or edit an existing one (on page 1545), a configuration dialog box is displayed that allows you to customize the transformation with various options in several tabs.

The **XSLT** tab contains the following options:

**XML URL**

Specifies the source XML file. You can specify the path by using the text field, its history drop-down, the ✉️ Insert Editor Variables (on page 323) button, or the browsing actions in the 📚 Browse drop-down list. You can also use the 📝 Open in editor button to open the specified file in the editor panel. This URL is resolved through the catalog resolver. If the catalog does not have a mapping for the URL, then the file is used directly from its remote location.

**Note:**
If the transformer engine is Saxon 9.x and a custom URI resolver is configured in the advanced Saxon preferences page (on page 249), the XML input of the transformation is passed to that URI resolver. If the transformer engine is one of the built-in XSLT 2.0 / 3.0 engines and the name of an initial template (on page 1512) is specified in the scenario, the XML URL field can be empty. The XML URL field can also be empty if you use external XSLT processors (on page 1479). Otherwise, a value is mandatory in this field.

**XSL URL**

Specifies the source XSL file that the transformation will use. You can specify the path by using the text field, its history drop-down, the ✉️ Insert Editor Variables (on page 323) button, or the browsing actions in the 📚 Browse drop-down list. You can also use the 📝 Open in editor button to open the specified file in the editor panel. This URL is resolved through the catalog resolver. If the catalog does not have a mapping for the URL, the file is used directly from its remote location.

**Use ”xml-stylesheet” declaration**

If selected, the scenario applies the stylesheet specified explicitly in the XML document with the xml-stylesheet processing instruction. By default, this option is deselected and the transformation applies the XSLT stylesheet that is specified in the XSL URL field.
Transformer

This drop-down menu presents all the transformation engines available to Oxygen XML Editor for performing a transformation. These include the built-in engines and the external engines defined in the Custom Engines preferences page (on page 259). The engine you choose is used as the default transformation engine. Also, if an XSLT or XQuery document does not have an associated validation scenario, this transformation engine is used in the validation process (if it provides validation support).

Advanced options

Allows you to configure the advanced options of the Saxon HE/PE/EE engine (on page 1468) for the current transformation scenario. To configure the same options globally, go to the Saxon-HE/PE/EE preferences page (on page 247). For the current transformation scenario, these Advanced options override the options configured in that preferences page.

Parameters

Opens a Configure parameters dialog box (on page 1466) that allows you to configure the XSLT parameters used in the current transformation. In this dialog box, you can also configure the parameters for additional XSLT stylesheets (on page 1512). If the XSLT transformation engine is custom-defined, you cannot use this dialog box to configure the parameters sent to the custom engine. Instead, you can copy all parameters from the dialog box using contextual menu actions and edit the custom XSLT engine to include the necessary parameters in the command line that starts the transformation process.

Extensions

Opens a dialog box for configuring the XSLT extension JARS or classes (on page 1467) that define extension Java functions or extension XSLT elements used in the transformation.

Additional XSLT stylesheets

Opens a dialog box for adding XSLT stylesheets (on page 1468) that are applied on the main stylesheet specified in the XSL URL field. This is useful when a chain of XSLT stylesheets must be applied to the input XML document.

XSLT Parameters

The global parameters of the XSLT stylesheet used in a transformation scenario can be configured by using the Parameters button in the XSLT tab of a new or edited transformation scenario dialog box.

The resulting dialog box includes a table that displays all the parameters of the current XSLT stylesheet, all imported and included stylesheets, and all additional stylesheets (on page 1468), along with their descriptions and current values. You can also add, edit, and remove parameters, and you can use the Filter text box to search for a specific term in the entire parameters collection. Note that edited parameters are displayed with their name in bold.

If the XPath column is selected, the parameter value is evaluated as an XPath expression before starting the XSLT transformation.
**Example:**

For example, you can use expressions such as:

```xml
<entry>
  <person atr='val'/>
</entry>
```

**Note:**

1. The `doc` function solves the argument relative to the XSL stylesheet location. You can use full paths or editor variables (on page 323) (such as `${cfdu}` [current file directory]) to specify other locations: `doc('${cfdu}/test.xml')="/`  
2. You cannot use XSLT Functions. Only XPath functions are allowed.

Below the table, the following actions are available for managing the parameters:

**New**

Opens the **Add Parameter** dialog box that allows you to add a new parameter to the list. An editor variable (on page 323) can be inserted in the text box using the **Insert Editor Variables** button. If the **Evaluate as XPath** option is selected, the parameter will be evaluated as an XPath expression.

**Edit**

Opens the **Edit Parameter** dialog box that allows you to edit the selected parameter. An editor variable (on page 323) can be inserted in the text box using the **Insert Editor Variables** button. If the **Evaluate as XPath** option is selected, the parameter will be evaluated as an XPath expression.

**Unset**

Resets the selected parameter to its default value. Available only for edited parameters with set values.

**Delete**

Removes the selected parameter from the list. It is available only for new parameters that have been added to the list.

The bottom panel presents the following:

- The default value of the parameter selected in the table.
- A description of the parameter, if available.
- The system ID of the stylesheet that declares it.
XSLT Extensions

The Extensions button opens a dialog box that allows you to specify the JARS and classes that contain extension functions called from the XSLT file of the current transformation scenario. You can use the Add, Edit, and Remove buttons to configure the extensions.

Tip:
You can specify the path to the resources using wildcards (for example, \${oxygenHome}/lib/*.jar).

An extension function called from the XSLT file of the current transformation scenario will be searched, in the specified extensions, in the order displayed in this dialog box. To change the order of the items, select the item to be moved and click the Move up or Move down buttons.

Additional XSLT Stylesheets

Use the Additional XSLT Stylesheets button in the XSLT tab to display a list of additional XSLT stylesheets to be used in the transformation and you can add files to the list or edit existing entries. The following actions are available:

Add

Adds a stylesheet in the Additional XSLT stylesheets list using a file browser dialog box. You can type an editor variable (on page 323) in the file name field of the browser dialog box. The name of the stylesheet will be added in the list after the current selection.

Remove

Deletes the selected stylesheet from the Additional XSLT stylesheets list.

Open

Opens the selected stylesheet in a separate view.

Up

Moves the selected stylesheet up in the list.

Down

Moves the selected stylesheet down in the list.

Advanced Saxon HE/PE/EE XSLT Transformation Options

The XSLT transformation scenario allows you to configure advanced options that are specific for the Saxon HE (Home Edition), PE (Professional Edition), and EE (Enterprise Edition) engines. They are the same options as those in the Saxon HE/PE/EE preferences page (on page 247) but they are configured as a specific set of transformation options for each transformation scenario, while the values set in the preferences page apply as
global options. The advanced options configured in a transformation scenario override the global options (on page 3219) defined in the preferences page.

**Saxon-HE/PE/EE Options**

The advanced options for Saxon 10.6 Home Edition (HE), Professional Edition (PE), and Enterprise Edition (EE) are as follows:

**Mode ("-im")**

A Saxon-specific option that sets the initial mode for the transformation. If this value is also specified in a configuration file (on page 1515), the value in this option takes precedence.

**Template ("-it")**

A Saxon-specific option that sets the name of the initial XSLT template to be executed. If this value is also specified in a configuration file (on page 1515), the value in this option takes precedence.

**Tip:**

If your stylesheet includes `<xsl:template name="xsl:initial-template">`, Oxygen XML Editor will automatically detect and use it as the initial template, so this option is not needed in this case.

**Use a configuration file ("-config")**

Select this option if you want to use a Saxon 10.6 configuration file that will be executed for the XSLT transformation and validation processes. You can specify the path to the configuration file by entering it in the URL field, or by using the Insert Editor Variables button, or using the browsing actions in the Browse drop-down list.

**Debugger trace into XPath expressions (applies to debugging sessions)**

Instructs the XSLT Debugger (on page 2121) to step into XPath expressions.

**Enable Optimizations ("-opt")**

This option is selected by default, which means that optimization is enabled. If not selected, the optimization is suppressed, which is helpful when reducing the compiling time is important, optimization conflicts with debugging, or optimization causes extension functions with side-effects to behave unpredictably.

**Line numbering ("-l")**

Line numbers where errors occur are included in the output messages.

**Expand attributes defaults ("-expand")**

Specifies whether or not the attributes defined in the associated DTD or XML Schema are expanded in the output of the transformation you are executing.

**DTD validation of the source ("-dtd")**
Specifies whether or not the source document will be validated against their associated DTD. You can choose from the following:

- **On** - Requests DTD validation of the source file and of any files read using the `document()` function.
- **Off** - (default setting) Suppresses DTD validation.
- **Recover** - Performs DTD validation but treats the errors as non-fatal.

**Note:**
Any external DTD is likely to be read even if not used for validation, since DTDs can contain definitions of entities.

**Strip whitespaces ("-strip")**

Specifies how the *strip whitespaces* operation is handled. You can choose one of the following values:

- **All** ("all") - Strips all whitespace text nodes from source documents before any further processing, regardless of any `@xml:space` attributes in the source document.
- **Ignore** ("ignorable") - Strips all ignorable whitespace text nodes from source documents before any further processing, regardless of any `@xml:space` attributes in the source document. Whitespace text nodes are ignorable if they appear in elements defined in the DTD or schema as having element-only content.
- **None** ("none") - Strips no whitespace before further processing.

**Enable profiling ("-TP")**

If selected, profiling of the execution time in a stylesheet is enabled. The corresponding text field is used to specify the path to the output file where the profiling information will be saved. As long as the option is selected, and the output file specified, it will gather timed tracing information and create a profile report to the specified file.

**Saxon-PE/EE Options**

The following advanced options are specific for Saxon 10.6 Professional Edition (PE) and Enterprise Edition (EE) only:

**Register Saxon-JS extension functions and instructions**

Registers the Saxon-CE extension functions and instructions when compiling a stylesheet using the Saxon 10.6 processors.

**Note:**
Saxon-CE, being JavaScript-based, was designed to run inside a web browser. This means that you will use Oxygen XML Editor only for developing the Saxon-CE stylesheet,
leaving the execution part to a web browser. See more details about executing such a stylesheet on Saxonica's website.

Allow calls on extension functions ("-ext")

If selected, the stylesheet is allowed to call external Java functions. This does not affect calls on integrated extension functions, including Saxon and EXSLT extension functions. This option is useful when loading an untrusted stylesheet (such as from a remote site using http://[URL]). It ensures that the stylesheet cannot call arbitrary Java methods and thus gain privileged access to resources on your machine.

Enable assertions ("-ea")

In XSLT 3.0, you can use the `<xsl:assert>` element to make assertions in the form of XPath expressions, causing a dynamic error if the assertion turns out to be false. If this option is selected, XSLT 3.0 `<xsl:assert>` instructions are enabled. If it is not selected (default), the assertions are ignored.

Saxon-EE Options

The advanced options that are specific for Saxon 10.6 Enterprise Edition (EE) are as follows:

XML Schema version

Use this option to change the default XML Schema version for this transformation. To change the default XML Schema version globally, open the Preferences dialog box (Options > Preferences) (on page 124) and go to XML > XML Parser > XML Schema and use the Default XML Schema version option (on page 238).

Validation of the source file ("-val")

Requests schema-based validation of the source file and of any files read using `document()` or similar functions. It can have the following values:

- **Schema validation ("strict")** - This mode requires an XML Schema and allows for parsing the source documents with strict schema-validation enabled.
- **Lax schema validation ("lax")** - If an XML Schema is provided, this mode allows for parsing the source documents with schema-validation enabled but the validation will not fail if, for example, element declarations are not found.
- **Disable schema validation** - This specifies that the source documents should be parsed with schema-validation disabled.

Validation errors in the result tree treated as warnings ("-outval")

Normally, if validation of result documents is requested, a validation error is fatal. Selecting this option causes such validation failures to be treated as warnings.

Write comments for non-fatal validation errors of the result document
The validation messages for non-fatal errors are written (wherever possible) as a comment in the result document itself.

**Enable streaming mode**

Selecting this option will allow an XSLT to run in streaming mode. It is not selected by default.

**Other Options**

**Initializer class**

Equivalent to the `--init` Saxon command-line argument. The value is the name of a user-supplied class that implements the `net.sf.saxon.lib.Initializer` interface. This initializer is called during the initialization process, and may be used to set any options required on the configuration programmatically. It is particularly useful for tasks such as registering extension functions, collations, or external object models, especially in Saxon-HE where the option cannot be set via a configuration file. Saxon only calls the initializer when running from the command line, but the same code may be invoked to perform initialization when running user application code.

**Using Saxon Integrated Extension Functions**

Saxon, the transformation and validation engine used by Oxygen XML Editor, can be customized by adding custom functions (called Integrated Extension Functions) that can be called from XPath.

To define such a function, follow these steps:

1. Create a file with a Java class that extends `net.sf.saxon.lib.ExtensionFunctionDefinition`. Here is an example:

```java
private static class ShiftLeft extends ExtensionFunctionDefinition {
    @Override
    public StructuredQName getFunctionQName() {
        return new StructuredQName("eg", "http://example.com/saxon-extension", "shift-left");
    }

    @Override
    public SequenceType[] getArgumentTypes() {
        return new SequenceType[] {SequenceType.SINGLE_INTEGER, SequenceType.SINGLE_INTEGER};
    }

    @Override
    public SequenceType getResultType(SequenceType[] suppliedArgumentTypes) {
        return SequenceType.SINGLE_INTEGER;
    }
}
```
```java
@override
public ExtensionFunctionCall makeCallExpression() {
    return new ExtensionFunctionCall() {
        public SequenceIterator call(SequenceIterator[] arguments, XPathContext context)
            throws XPathException {
            long v0 = ((IntegerValue)arguments[0].next()).longValue();
            long v1 = ((IntegerValue)arguments[1].next()).longValue();
            long result = v0<<v1;
            return Value.asIterator(Int64Value.makeIntegerValue(result));
        }
    };
}
```

2. Compile the class and add it to a JAR file.
3. Add a file called `net.sf.saxon.lib.ExtensionFunctionDefinition` that contains the fully qualified name of the Java class in the `META-INF/services/` folder of the JAR file.

**Note:**
To add more function definitions in the same JAR file, you need to add their fully qualified names on different lines.

To enable Oxygen XML Editor to pick up your custom function definition, the JAR file should be added to the classpath of the transformer. Here are some possibilities:

- If you develop a framework, you just need to link the JAR file in the **Classpath** tab (on page 144).
- In a validation scenario (on page 788), you can use the **Extensions** button to open a dialog box where you can add libraries.
- In a transformation scenario, you can use the **Extensions** button in the **XSLT** tab (on page 1466) to open a dialog box where you can add libraries.
- You can also create a plugin that contributes such a JAR file in the classpath (on page 2411).

**FO Processor Tab (XSLT Transformations)**

When you create a new transformation scenario (on page 1464) or edit an existing one (on page 1545), a configuration dialog box is displayed that allows you to customize the transformation with various options in several tabs.

The **FO Processor** tab contains the following options:

**Perform FO Processing**

Specifies whether or not an FO processor is applied (either the built-in Apache FOP engine or an external engine defined in **Preferences**) during the transformation.
Input

Choose between the following options to specify which input file to use:

- **XSLT result as input** - The FO processor is applied to the result of the XSLT transformation that is defined in the XSLT tab.
- **XML URL as input** - The FO processor is applied to the input XML file.

Method

The output format of the FO processing. The available options depend on the selected processor type.

Processor

Specifies the FO processor to be used for the transformation. It can be the built-in Apache FOP processor or an external processor (on page 261).

Output Tab (XSLT Transformations)

When you create a new transformation scenario (on page 1464) or edit an existing one (on page 1545), a configuration dialog box is displayed that allows you to customize the transformation with various options in several tabs.

The **Output** tab contains the following options:

**Prompt for file**

At the end of the transformation, a file browser dialog box is displayed for specifying the path and name of the file that stores the transformation result.

**Save As**

The path of the file where the result of the transformation is stored. You can specify the path by using the text field, the 📏 **Insert Editor Variables** (on page 323) button, or the 📦 **Browse** button.

**Open in Browser/System Application**

If selected, Oxygen XML Editor automatically opens the result of the transformation in a system application associated with the file type of the result (for example, in Windows PDF files are often opened in Acrobat Reader).

**Note:**

To set the web browser that is used for displaying HTML/XHTML pages, go to Options > Preferences > Global, and set it in the Default Internet browser field.
• **Output file** - When **Open in Browser/System Application** is selected, you can use this button to automatically open the default output file at the end of the transformation.

• **Other location** - When **Open in Browser/System Application** is selected, you can use this option to open the file specified in this field at the end of the transformation. You can specify the path by using the text field, the `Insert Editor Variables (on page 323)` button, or the `Browse` button.

**Open in editor**

When this option is selected, at the end of the transformation, the default output file is opened in a new editor panel with the appropriate built-in editor type (for example, if the result is an XML file it is opened in the built-in XML editor, or if it is an XSL-FO file it is opened with the built-in FO editor).

**Show in results view as**

You can choose to view the results in one of the following:

• **XML** - If this is selected, Oxygen XML Editor displays the transformation result in an XML viewer panel at the bottom of the application window with syntax highlighting (on page 224).

• **SVG** - If this is selected, Oxygen XML Editor displays the transformation result in an integrated SVG viewer in the **Results panel (on page 1254)** at the bottom of the application window and renders the result as an SVG image.

• **XHTML** - This option is only available if **Open in Browser/System Application** is not selected. If selected, Oxygen XML Editor displays the transformation result in a built-in XHTML browser panel at the bottom of the application window.

**Important:**

When transforming very large documents, you should be aware that selecting this option may result in very long processing times. This drawback is due to the built-in Java XHTML browser implementation. To avoid delays for large documents, if you want to see the XHTML result of the transformation, you should use an external browser by selecting the **Open in Browser/System Application** option instead.

• **Image URLs are relative to** - If **Show in results view as XHTML** is selected, this option specifies the path used to resolve image paths contained in the transformation result. You can specify the path by using the text field, the `Insert Editor Variables (on page 323)` button, or the `Browse` button.
Attention:
If your input XSLT contains `<xsl:result-document>` elements, then the secondary results will be saved to the specified URIs while the principal result is specified in this Output tab. For more information, see: https://www.w3.org/TR/xslt-30/#element-result-document.

Configuring an XSLT Processor for Generating Output

This section explains how to configure an XSLT processor and extensions for such a processor in Oxygen XML Editor.

Supported XSLT Processors

Oxygen XML Editor includes the following XSLT processors:

- **Xalan 2.7.2 - Xalan-Java** is an XSLT processor for transforming XML documents into HTML, text, or other XML document types. It implements XSL Transformations (XSLT) Version 1.0 and XML Path Language (XPath) Version 1.0.

- **Saxon 6.5.5 - Saxon 6.5.5** is an XSLT processor that implements the Version 1.0 XSLT and XPath with a number of powerful extensions. This version of Saxon also includes many of the new features that were first defined in the XSLT 1.1 working draft, but for conformance and portability reasons these are not available if the stylesheet header specifies `version="1.0"`.

- **Saxon 10.6 Home Edition (HE), Professional Edition (PE) - Saxon-HE/PE** implements the basic conformance level for XSLT 2.0 / 3.0 and XQuery 1.0. The term basic XSLT 2.0 / 3.0 processor is defined in the draft XSLT 2.0 / 3.0 specifications. It is a conformance level that requires support for all features of the language other than those that involve schema processing. The HE product remains open source, but removes some of the more advanced features that are present in Saxon-PE.

- **Saxon 10.6 Enterprise Edition (EE) - Saxon EE** is the schema-aware edition of Saxon and it is one of the built-in processors included in Oxygen XML Editor. Saxon EE includes an XML Schema processor, and schema-aware XSLT, XQuery, and XPath processors.

The validation in schema aware transformations is done according to the XML Schema 1.0 or 1.1. This can be configured in Preferences (on page 238).

Note:
Oxygen XML Editor implements a Saxon framework that allows you to create Saxon configuration files. Two templates are available: Saxon collection catalog and Saxon configuration. Both of these templates support content completion, element annotation, and attribute annotation.

Note:
Saxon can use the ICU-J localization library (saxon9-icu.jar) to add support for sorting and date/number formatting in a wide variety of languages. This library is not included in
the Oxygen XML Editor installation kit. However, Saxon will use the default collation and localization support available in the currently used JRE. To enable this capability, follow these steps:

2. Unpack the downloaded archive.
3. Create a new XSLT transformation scenario (or edit an existing one). In the XSLT tab, click the Extensions button to open the list of additional libraries used by the transformation process.
4. Click Add and browse to the folder where you unpacked the downloaded archive and choose the saxon9-icu.jar file.

Note that the saxon9-icu.jar should NOT be added to the application library folder because it will conflict with another version of the ICU-J library that comes bundled with Oxygen XML Editor.

• **Saxon-CE (Client Edition)** is Saxonica's implementation of XSLT 2.0 for use on web browsers. Oxygen XML Editor provides support for editing stylesheets that contain Saxon-CE extension functions and instructions. This support improves the validation, content completion, and syntax highlighting.

  **Note:**
  
  Saxon-CE, being JavaScript-based, was designed to run inside a web browser. This means that you will use Oxygen XML Editor only for developing the Saxon-CE stylesheet, leaving the execution part to a web browser. See more details about executing such a stylesheet on Saxonica's website.

  **Note:**
  
  A specific template, named **Saxon-CE stylesheet**, is available in the **New document wizard (on page 368)**.

• **Xsltproc (libxslt)** - Libxslt is the XSLT C library developed for the Gnome project. Libxslt is based on libxml2, the XML C library developed for the Gnome project. It also implements most of the EXSLT set of processor-portable extensions, functions, and some of Saxon's evaluate and expression extensions.

Oxygen XML Editor uses Libxslt through its command-line tool (Xsltproc). Depending on your operating system, you must download the Libxslt libraries on your machine from http://xmlsoft.org/XSLT/downloads.html and place them in a local folder. Then you need to update the PATH environmental variable to contain the parent folder where the xsltproc executable is located.

  **Tip:**
  
  As an example, a Windows installation of the Xsltproc engine would follow these steps:

2. Unzip all of them into the same folder of your choice.

3. Edit the `PATH` environment variable and add the `bin` folder for all four archives:

```
D:\apache-maven-3.1.1\bin
D:\Python27
%PATH%
C:\Users\r\Desktop\abc\libxslt-1.1.26.win32\bin
C:\Users\r\Desktop\abc\libxml2-2.7.8.win32\bin
C:\Users\r\Desktop\abc\iconv-1.9.2.win32\bin
C:\Users\r\Desktop\abc\zlib-1.2.5\bin
```

4. Restart Oxygen XML Editor.

**Result:** You can now use the `xsltproc` processor as an XSLT engine in the XSLT transformation scenario.

**Note:**
The Xsltproc processor can be configured from the XSLTPROC options page (on page 250).

**CAUTION:**
There is a known problem where file paths that contain spaces are not handled correctly in the LIBXML processor. For example, the built-in XML Catalog (on page 3225) files of the built-in document types (DocBook, TEI, DITA, etc.) are not handled properly by LIBXML if Oxygen XML Editor is installed in the default location on Windows (C:\Program Files). This is because the built-in XML catalog files are stored in the `\OXYGEN_INSTALL_DIR\frameworks` subdirectory of the installation directory, and in this case it contains a space character.

- **MSXML 4.0 (Legacy)** - MSXML 4.0 is available only on Windows platforms. It can be used for transformation (on page 1464) and validation of XSLT stylesheets (on page 889).

Oxygen XML Editor uses the Microsoft XML parser through its command-line tool `msxsl.exe`.

Since `msxsl.exe` is only a wrapper, Microsoft Core XML Services (MSXML) must be installed on the computer. Otherwise, you will get a corresponding warning. You can get the latest Microsoft XML parser from Microsoft web-site.
• **MSXML .NET (Legacy)** - MSXML .NET is available only on Windows platforms. It can be used for transformation (on page 1464) and validation of XSLT stylesheets (on page 889).

Oxygen XML Editor performs XSLT transformations and validations using the .NET Framework XSLT implementation (System.Xml.Xsl.XslTransform class) through the nxslt command-line utility. The nxslt version included in Oxygen XML Editor is 1.6.

You should have the .NET Framework version 1.0 already installed on your system. Otherwise, you will get the following warning: MSXML.NET requires .NET Framework version 1.0 to be installed. Exit code: 128.

You can get the .NET Framework version 1.0 from the Microsoft website.

• **.NET 1.0 (Legacy)** - A transformer based on the System.Xml 1.0 library available in the .NET 1.0 and .NET 1.1 frameworks from Microsoft. It is available only on Windows.

You should have the .NET Framework version 1.0 or 1.1 already installed on your system. Otherwise, you will get the following warning: MSXML.NET requires .NET Framework version 1.0 to be installed. Exit code: 128.

You can get the .NET Framework version 1.0 from the Microsoft website.

• **.NET 2.0 (Legacy)** - A transformer based on the System.Xml 2.0 library available in the .NET 2.0 Framework from Microsoft. It is available only on Windows.

You should have the .NET Framework version 2.0 already installed on your system. Otherwise, you will get the following warning: MSXML.NET requires .NET Framework version 2.0 to be installed. Exit code: 128.

You can get the .NET Framework version 2.0 from the Microsoft website.

For information about configuring the XSLT preferences, see the XSLT options (on page 245) section.

### Configuring Custom XSLT Processors

Oxygen XML Editor allows you to configure custom processors to be used for running XSLT and XQuery transformations.

To add a new custom processor, follow these steps:

1. Open the Preferences dialog box (Options > Preferences) (on page 124) and go to XML > XSLT-XQuery > Custom Engines.
2. Click the New button at the bottom of the dialog box.
3. Configure the parameters for the custom engine (on page 259).
4. Click OK.
Note:
You can not use these custom engines in the Debugger perspective (on page 2102).

The output messages of a custom processor are displayed in an output view at the bottom of the application window. If an output message follows the format of an Oxygen XML Editor linked message (on page 784), clicking it highlights the location of the message in an editor panel containing the file referenced in the message.

Related Information:
Custom Engines Preferences (on page 259)

Configuring the XSLT Processor Extensions Paths

The Xalan and Saxon processors support the use of extension elements and extension functions. Unlike a literal result element, which the stylesheet simply transfers to the result tree, an extension element performs an action. The extension is usually used because the XSLT stylesheet fails in providing adequate functions for accomplishing a more complex task.

For more information about how to use extensions, see the following links:

- Saxon 6.5.5 - http://saxon.sourceforge.net/saxon6.5.5/extensions.html

To set an XSLT processor extension (a directory or a jar file), use the Extensions button (on page 1466) in the Edit scenario dialog box.

XSL-FO (Apache FOP) Processor for Generating PDF Output

The Oxygen XML Editor installation package is distributed with the Apache FOP that is a Formatting Objects processor for transforming your XML documents to PDF. FOP is a print and output independent formatter driven by XSL Formatting Objects. FOP is implemented as a Java application that reads a formatting object tree and renders the resulting pages to a specified output.

To see the version of the built-in XSL-FO processor for your installation, go to Help > About > Libraries and search for Apache FOP.

Other FO processors can be configured in the FO Processors preferences page (on page 261).

Add a Font to the Built-in FO Processor - Simple Version

If the font that must be set to Apache FOP is one of the fonts that are installed in the operating system you should follow the next steps for creating and setting a FOP configuration file that looks for the font that it needs in the system fonts. It is a simplified version of the procedure for setting a custom font in Apache FOP (on page 1528).
1. Register the font in FOP configuration. (This is not necessary for DITA PDF transformations, skip to the next step)

   a. Create a FOP configuration file that specifies that FOP should look for fonts in the installed fonts of the operating system.

```
<fop version="1.0">
  <renderers>
    <renderer mime="application/pdf">
      <fonts>
        <auto-detect/>
      </fonts>
    </renderer>
  </renderers>
</fop>
```

   b. Open the Preferences dialog box (Options > Preferences) (on page 124), go to XML > PDF Output > FO Processors, and enter the path of the FOP configuration file in the Configuration file text field.

2. Set the font on the document content.

   This is done usually with XSLT stylesheet parameters and depends on the document type processed by the stylesheet.

   - For DocBook documents you can start with the built-in scenario called DocBook PDF, edit the XSLT parameters (on page 1464) and set the font name (for example, Arial Unicode MS) to the body.font.family and title.font.family parameters.
   - For TEI documents you can start with the built-in scenario called TEI PDF, edit the XSLT parameters (on page 1464) and set the font name (for example, Arial Unicode MS) to the bodyFont and sansFont parameters.
   - For DITA transformations to PDF using DITA-OT you should modify the following two files:
     - `DITA-OT-DIR/plugins/org.dita.pdf2/cfg/fo/font-mappings.xml` - The `<font-face>` element included in each `<physical-font>` element that has the charset="default" attribute must contain the name of the font (for example, Arial Unicode MS)
     - `DITA-OT-DIR/plugins/org.dita.pdf2.fop/fop/conf/fop.xconf` - An `<auto-detect>` element must be inserted in the `<fonts>` element, which is inside the `<renderer>` element that has the mime="application/pdf" attribute:

```
<renderer mime="application/pdf">
  . . .
  <fonts>
    <auto-detect/>
  </fonts>
  . . .
</renderer>
```
Add a Font to the Built-in FO Processor - Advanced Version

If an XML document is transformed to PDF using the built-in Apache FOP processor but it contains some Unicode characters that cannot be rendered by the default PDF fonts, then a special font that is capable to render these characters must be configured and embedded in the PDF result.

Important:
On Windows, fonts are located into the C:\Windows\Fonts directory. On macOS, they are placed in /Library/Fonts. To install a new font on your system, it is enough to copy it in the Fonts directory. If a special font is installed in the operating system, there is a simple way of telling FOP to look for it. See the simplified procedure for adding a font to FOP (on page 1526).

1. Locate the font.
   First, find out the name of a font that has the glyphs for the special characters you used. One font that covers most characters, including Japanese, Cyrillic, and Greek, is Arial Unicode MS.

2. Register the font in the FOP configuration.

   a. For information about registering the font in the FOP Configuration, see: https://xmlgraphics.apache.org/fop/2.3/fonts.html.
   b. Open the Preferences dialog box (Options > Preferences) (on page 124), go to XML > PDF Output > FO Processors, and enter the path of the FOP configuration file in the Configuration file text field.

3. Set the font on the document content.
   This is usually done with XSLT stylesheet parameters and depends on the document type processed by the stylesheet.

   **DocBook Example:** For DocBook documents, you can start with the built-in scenario called DocBook PDF, edit the XSLT parameters (on page 1464), and set the font name (for example, Arialuni) to the body.font.family and title.font.family parameters.

   **TEI Example:** For TEI documents, you can start with the built-in scenario called TEI PDF, edit the XSLT parameters (on page 1464), and set the font name (for example, Arialuni) to the bodyFont and sansFont parameters.

   **DITA Example:** For DITA to PDF transformations using DITA-OT modify the following two files:
- The `<font-face>` element included in each `<physical-font>` element that has the `char-set="default"` attribute must contain the name of the font.

- A `<font>` element must be inserted in the `<fonts>` element, which is inside the `<renderer>` element that has the `mime="application/pdf"` attribute.

For more information, see: https://xmlgraphics.apache.org/fop/2.1/fonts.html.

Adding Libraries to the Built-in FO Processor (XML with XSLT and FO)

Starting with Oxygen XML Editor version 20.0, both hyphenation and PDF image support are enabled by default in the built-in Apache FO processor. For older version of Oxygen XML Editor, use the following procedures to enable such support.

Adding Hyphenation Support for XML with XSLT Transformation Scenarios

If you want to add newer hyphenation libraries or you are using an older version of Oxygen XML Editor, follow this procedure:

1. Create a folder called `fop` in the `[OXYGEN_INSTALL_DIR]/lib` folder.
2. Download the compiled JAR (on page 3220) from OFFO.
3. Copy the `fop-hyph.jar` file into the `[OXYGEN_INSTALL_DIR]/lib/fop` folder.
4. Restart Oxygen XML Editor.

Adding Support for PDF Images

To add support for PDF images in an older version of Oxygen XML Editor, follow these steps:

1. Create a folder called `fop` in the `[OXYGEN_INSTALL_DIR]/lib` folder.
2. Download the `fop-pdf-images` JAR libraries.
3. Copy the libraries into the `[OXYGEN_INSTALL_DIR]/lib/fop` folder.
4. Restart Oxygen XML Editor.

How to Enable Debugging for FO Processor Transformations

If you encounter errors when running PDF transformations that use an FO processor, it is possible to enable debugging/logging to help you identify the problem. To enable debugging/logging for FO processing, follow this procedure:
1. Locate and edit the following configuration file: `{OXYGEN_INSTALL_DIR}/tools/config/logback.xml`.

**Note:**
You need write access to this folder, so if you do not have administrator permissions, you might first need to copy the file to another location where you have write access.

2. Edit the `<root>` element (inside the `<configuration>` element), change its level to `debug`, and save the file.
3. Restart Oxygen XML Editor and re-run the transformation.

**Tip:**
To make it easier to analyze the data in the logs, it is recommended that you use a small input file when trying to reproduce the problem.

4. Once you are finished with the debugging session, remember to edit the `logback.xml` file and change the `<root>` element back to its original value. Otherwise, performance could be affected.

**XProc Transformation**

This type of transformation specifies the parameters and location of an XProc script.

A sequence of transformations described by an XProc script can be executed with an XProc transformation scenario. To create an XProc transformation scenario, use one of the following methods:

- Use the Configure Transformation Scenario(s) (`Ctrl + Shift + C (Command + Shift + C on macOS)`) action from the toolbar or the Document > Transformation menu. Then click the New button and select XProc transformation.

- Go to Window > Show View and select Transformation Scenarios to display this view (on page 1555). Click the New Scenario drop-down menu button and select XProc transformation.

Both methods open the New Scenario dialog box.

The upper part of the dialog box allows you to specify the Name of the transformation scenario and the following Storage options:

- **Project Options (on page 3222)** - The scenario is stored in the project file and can be shared with other users. For example, if your project is saved on a source versioning/sharing system (CVS, SVN, Source Safe, etc.) or a shared folder, your team can use the scenarios that you store in the project file.

- **Global Options (on page 3219)** - The scenario is saved in the global options that are stored in the user home directory and is not accessible to other users.

The lower part of the dialog box contains several tabs that allow you to configure the options that control the transformation.
XProc Tab

When you create a new transformation scenario (on page 1464) or edit an existing one (on page 1545), a configuration dialog box is displayed that allows you to customize the transformation with various options in several tabs.

The XProc tab contains the following options:

XProc URL

Specify the source XProc file to be used by the transformation. You can specify the path by using the text field, its history drop-down, the Insert Editor Variables (on page 323) button, or the browsing actions in the Browse drop-down list. This URL is resolved through the catalog resolver. If the catalog does not have a mapping for the URL, the file is used directly from its remote location.

Processor

Allows you to select the XProc engine to be used for the transformation. You can select the Add-on for Calabash XProc engine or a custom engine that is configured in the XProc Preferences page (on page 244).

Inputs Tab (XProc Transformations)

When you create a new transformation scenario (on page 1464) or edit an existing one (on page 1545), a configuration dialog box is displayed that allows you to customize the transformation with various options in several tabs.

The Inputs tab contains a list with the ports that the XProc script uses to read input data. Use the Filter text box to search for a specific term in the entire ports collection.

Each input port has an assigned name in the XProc script. The XProc engine reads data from the URL specified in the URL column.

The following actions are available for managing the input ports:

New

Opens an Edit dialog box that allows you to add a new port and its URL. The built-in editor variables (on page 323) and custom editor variables (on page 332) can be used to specify the URL.

Edit
Opens an **Edit** dialog box that allows you to modify the selected port and its URL. The **built-in editor variables (on page 323)** and **custom editor variables (on page 332)** can be used to specify the URL.

**Delete**

Removes the selected port from the list. It is available only for new ports that have been added to the list.

**Parameters Tab (XProc Transformations)**

When you **create a new transformation scenario (on page 1464)** or **edit an existing one (on page 1545)**, a configuration dialog box is displayed that allows you to customize the transformation with various options in several tabs.

The **Parameters** tab presents a list of ports and parameters collected from the XProc script. The tab is divided into three sections:

- **List of Ports**
  
  In this section, you can use the **New** and **Delete** buttons to add or remove ports.

- **List of Parameters**
  
  This section presents a list of parameters for each port and includes columns for the parameter name, namespace URI, and its value. Use the **Filter** text box to search for a specific term in the entire parameters collection. You can use the **New** and **Delete** buttons to add or remove parameters. You can edit the value of each cell in this table by double-clicking the cell. You can also sort the parameters by clicking the column headers.

- **Editor Variable Information**
  
  The **built-in editor variables (on page 323)** and **custom editor variables (on page 332)** can be used for specifying the URI. The message pane at the bottom of the dialog box provides more information about the editor variables that can be used.

**Outputs Tab (XProc Transformations)**

When you **create a new transformation scenario (on page 1464)** or **edit an existing one (on page 1545)**, a configuration dialog box is displayed that allows you to customize the transformation with various options in several tabs.

The **Outputs** tab displays a list of output ports (along with the URL) collected from the XProc script. Use the **Filter** text box to search for a specific term in the entire ports collection. You can also sort the columns by clicking the column headers.

The following actions are available for managing the output ports:

- **New**
  
  Opens an **Edit** dialog box that allows you to add a new output port and its URL. An **editor variable (on page 323)** can be inserted for the URL by using the **Insert Editor Variables** button. There
is also a **Show in transformation results view** option that allows you to select whether or not the results will be displayed in the output **Results view (on page 548)**.

**Edit**

Opens an **Edit** dialog box that allows you to edit an existing output port and its URL. An **editor variable (on page 323)** can be inserted for the URL by using the **Insert Editor Variables** button. There is also a **Show in transformation results view** option that allows you to select whether or not the results will be displayed in the output **Results view (on page 548)**.

**Delete**

Removes the selected output port from the list. It is available only for new ports that have been added to the list.

Additional options that are available at the bottom of this tab include:

**Open in Editor**

If this option is selected, the XProc transformation result is automatically opened in an editor panel.

**Open in Browser/System Application**

If this option is selected, you can specify a file to be opened at the end of the XProc transformation in the browser or system application that is associated with the file type. You can specify the path by using the text field, its history drop-down, the **Insert Editor Variables (on page 323)** button, or the browsing actions in the **Browse** drop-down list.

**Results**

The result of the XProc transformation can be displayed as a sequence in an output view with two sections:

- A list with the output ports on the left side.
- The content that correspond to the selected output port on the right side.

**Figure 478. XProc Transformation Results View**
Options Tab (XProc Transformations)

When you create a new transformation scenario (on page 1464) or edit an existing one (on page 1545), a configuration dialog box is displayed that allows you to customize the transformation with various options in several tabs.

The **Options** tab displays a list of the options collected from the XProc script. The tab is divided into two sections:

**List of Options**

This section presents a list of options and includes columns for the option name, namespace URI, and its value. Use the **Filter** text box to search for a specific term in the entire options collection. You can use the **New** and **Delete** buttons to add or remove options. You can edit the value of each cell in this table by double-clicking the cell. You can also sort the parameters by clicking the column headers. The names of edited options are displayed in bold.

**Editor Variable Information**

The built-in editor variables (on page 323) and custom editor variables (on page 332) can be used for specifying the URI. This section provides more information about the editor variables that can be used.

Calabash XProc Processor for Generating PDF Output

To generate PDF output from your XProc pipeline (when using the Calabash XProc processor), follow these steps:

1. Open the **[OXYGEN_INSTALL_DIR]/lib/xproc/calabash/engine.xml** file.
2. Uncomment the `<system-property name="com.xmlcalabash.fo-processor" value="com.xmlcalabash.util.FoXEP"/>` system property.
3. Uncomment the `<system-property name="com.renderx.xep.CONFIG" file="../../../tools/xep/xep.xml"/>` system property. Edit the `<file` attribute to point to the configuration file that is usually located in the XEP installation folder.
4. Uncomment the references to the XEP libraries. Edit them to point to the matching library names from the XEP installation directory.
5. Restart Oxygen XML Editor.

Integrating an External XProc Engine

Oxygen XML Editor includes a bundled version of the Calabash XProc engine that can be used for XProc transformations and validation, but you can also integrate other external XProc engines. When you edit an XProc transformation scenario, there is a **Processor** drop-down menu where you can select the XProc engine to be used for the transformation.

If you do not need the external XProc engine to be used for automatic validation or pass parameters/ports and it is not Java-based, you can simply add the external engine by using the **XProc preferences page** (on page...
Otherwise, if the external engine is Java-based, or it has validation support, or it can receive parameters or ports passed from the transformation, you need to integrate it using the plugin extension procedure below.

For example, there is a public project on GitHub that is an implementation for integrating Morgana XProc with Oxygen XML Editor: [https://github.com/xml-project/support-for-xmleditor](https://github.com/xml-project/support-for-xmleditor). Also, the Javadoc documentation of the XProc API is available for download from the application website as a zip file: [xprocAPI.zip](https://github.com/xml-project/support-for-xmleditor).

To create an XProc integration project, follow these steps:

1. Move the `oxygen.jar` file from `[OXYGEN_INSTALL_DIR]/lib` to the `lib` folder of your project.
2. Implement the `ro.sync.xml.transformer.xproc.api.XProcTransformerInterface` interface.
3. Create a [Java archive (JAR) (on page 3220)](https://github.com/xml-project/support-for-xmleditor) from the classes you created.
4. Create an `engine.xml` file according to the `engine.dtd` file. The attributes of the `<engine>` element are as follows:
   a. `name` - The name of the XProc engine.
   b. `description` - A short description of the XProc engine.
   c. `class` - The complete name of the class that implements `ro.sync.xml.transformer.xproc.api.XProcTransformerInterface`.
   d. `version` - The version of the integration.
   e. `engineVersion` - The version of the integrated engine.
   f. `vendor` - The name of the vendor / implementer.
   g. `supportsValidation` - true if the engine supports validation (otherwise, `false`).

The `<engine>` element has only one child, `<runtime>`. The `<runtime>` element contains several `<library>` elements with the `@name` attribute containing the relative or absolute location of the libraries necessary to run this integration.

5. Create a new folder (for example, named `MyXprocEngine`) and place the `engine.xml` and all the libraries necessary to run the new integration in that folder.
6. Place that new folder (e.g. `MyXprocEngine`) inside a new plugin folder. This new plugin folder should also contain a `plugin.xml` file that points to the new engine folder (e.g. `MyXprocEngine`).

The `plugin.xml` file would look like this (it is based on the [AdditionalXProcEngine extension (on page 2423)](https://github.com/xml-project/support-for-xmleditor)):

```xml
<plugin
   id="morgana.xproc.addon"
   name="Contribute Morgana XProc"
   description="Contribute Morgana XProc"
   version="1.0"
   vendor="Syncro Soft"
   class="ro.sync.exml.plugin.Plugin"
   classLoaderType="preferReferencedResources">
   <extension type="AdditionalXProcEngine" path="MyXprocEngine/"/>
</extension>
</plugin>
```
XQuery Transformation

This type of transformation specifies the parameters and location of an XML source that the edited XQuery file is applied on.

**Note:**
When the XML source is a native XML database, the source field of the scenario is empty because the XML data is read with XQuery-specific functions, such as `document()`. When the XML source is a local XML file, the URL of the file is specified in the input field of the scenario.

To create an XQuery transformation scenario, use one of the following methods:

- Use the Configure Transformation Scenario(s) (Ctrl + Shift + C (Command + Shift + C on macOS)) action from the toolbar or the Document > Transformation menu. Then click the New button and select XQuery transformation.
- Go to Window > Show View and select Transformation Scenarios to display this view (on page 1555). Click the New Scenario drop-down menu button and select XQuery transformation.

Both methods open the New Scenario dialog box.

The upper part of the dialog box allows you to specify the Name of the transformation scenario and the following Storage options:

- **Project Options (on page 3222)** - The scenario is stored in the project file and can be shared with other users. For example, if your project is saved on a source versioning/sharing system (CVS, SVN, Source Safe, etc.) or a shared folder, your team can use the scenarios that you store in the project file.
- **Global Options (on page 3219)** - The scenario is saved in the global options that are stored in the user home directory and is not accessible to other users.

The lower part of the dialog box contains several tabs that allow you to configure the options that control the transformation.

**XQuery Tab**

When you create a new transformation scenario (on page 1464) or edit an existing one (on page 1545), a configuration dialog box is displayed that allows you to customize the transformation with various options in several tabs.
The **XQuery** tab contains the following options:

### XML URL

Specifies the source XML file. You can specify the path by using the text field, its history drop-down, the ✏️ **Insert Editor Variables** ([on page 323](#)) button, or the browsing actions in the 🌐 Browse drop-down list. You can also use the 📖 **Open in editor** button to open the specified file in the editor panel. This URL is resolved through the catalog resolver. If the catalog does not have a mapping for the URL, then the file is used directly from its remote location.

**Note:**

If the transformer engine is Saxon 9.x and a custom URI resolver is configured in the [advanced Saxon preferences page (on page 249)](#), the XML input of the transformation is passed to that URI resolver.

### XQuery URL

Specifies the source XQuery file to be used for the transformation. You can specify the path by using the text field, its history drop-down, the ✏️ **Insert Editor Variables** ([on page 323](#)) button, or the browsing actions in the 🌐 Browse drop-down list. You can also use the 📖 **Open in editor** button to open the specified file in the editor panel. This URL is resolved through the catalog resolver. If the catalog does not have a mapping for the URL, the file is used directly from its remote location.

### Transformer

This drop-down menu presents all the transformation engines available to Oxygen XML Editor for performing a transformation. These include the built-in engines and [the external engines defined in the Custom Engines preferences page (on page 259)](#). The engine you choose is used as the default transformation engine. Also, if an XSLT or XQuery document does not have an associated validation scenario, this transformation engine is used in the validation process (if it provides validation support).

**Advanced options**

Allows you to configure the [advanced options of the Saxon HE/PE/EE engine (on page 1483)](#) for the current transformation scenario. To configure the same options globally, go to the [Saxon-HE/PE/EE preferences page (on page 247)](#). For the current transformation scenario, these Advanced options override the options configured in that preferences page.

### Parameters

Opens the Configure parameters dialog box ([on page 1482](#)) for configuring the XQuery parameters. You can use the buttons in this dialog box to add, edit, or remove parameters. If the XQuery transformation engine is custom-defined, you cannot use this dialog box to set parameters. Instead, you can copy all parameters from the dialog box using contextual menu actions and edit the custom XQuery engine to include the necessary parameters in the command line that starts the transformation process.

### Extensions
Opens a dialog box for configuring the XQuery extension JARS or classes (on page 1483) that define extension Java functions or extension XSLT elements used in the transformation.

**XQuery Parameters**

The global parameters of the XQuery file used in a transformation scenario can be configured by using the **Parameters** button in the **XQuery** tab.

The resulting dialog box includes a table that displays all the parameters of the current XQuery file, along with their descriptions and current values. You can also add, edit, and remove parameters, and use the **Filter** text box to search for a specific term in the entire parameters collection. Note that edited parameters are displayed with their name in bold.

If the **XPath** column is selected, the parameter value is evaluated as an XPath expression before starting the XQuery transformation.

**Example:**

For example, you can use expressions such as:

```
<doc('test.xml')//entry
//person[@atr='val']>
```

**Note:**

1. The `doc` function solves the argument relative to the XQuery file location. You can use full paths or **editor variables (on page 323)** (such as `$cfdu` [current file directory]) to specify other locations: `doc('${cfdu}/test.xml')//*`
2. Only XPath functions are allowed.

Below the table, the following actions are available for managing the parameters:

**New**

Opens the **Add Parameter** dialog box that allows you to add a new parameter to the list. An **editor variable (on page 323)** can be inserted in the text box using the **Insert Editor Variables** button. If the **Evaluate as XPath** option is selected, the parameter will be evaluated as an XPath expression.

**Edit**

Opens the **Edit Parameter** dialog box that allows you to edit the selected parameter. An **editor variable (on page 323)** can be inserted in the text box using the **Insert Editor Variables** button. If the **Evaluate as XPath** option is selected, the parameter will be evaluated as an XPath expression.

**Unset**
Resets the selected parameter to its default value. Available only for edited parameters with set values.

**Delete**

Removes the selected parameter from the list. It is available only for new parameters that have been added to the list.

The bottom panel presents the following:

- The default value of the parameter selected in the table.
- A description of the parameter, if available.
- The system ID of the stylesheet that declares it.

**Related Information:**

*Editor Variables (on page 323)*

**XQuery Extensions**

The Extension button is used to specify the JAR (on page 3220) and classes that contain extension functions called from the XQuery file of the current transformation scenario. You can use the Add, Edit, and Remove buttons to configure the extensions.

An extension function called from the XQuery file of the current transformation scenario will be searched, in the specified extensions, in the order displayed in this dialog box. To change the order of the items, select the item to be moved and click the ✆ Move up or ✅ Move down buttons.

**Advanced Saxon HE/PE/EE XQuery Transformation Options**

The XQuery transformation scenario allows you to configure advanced options that are specific for the Saxon HE (Home Edition), PE (Professional Edition), and EE (Enterprise Edition) engines. They are the same options as those in the Saxon HE/PE/EE preferences page (on page 254) but they are configured as a specific set of transformation options for each transformation scenario, while the values set in the preferences page apply as global options. The advanced options configured in a transformation scenario override the global options (on page 3219) defined in the preferences page.

**Saxon-HE/PE/EE Options**

The advanced options for Saxon 10.6 Home Edition (HE), Professional Edition (PE), and Enterprise Edition (EE) are as follows:

- **Use a configuration file ("-config")**
  
  Sets a Saxon 10.6 configuration file that is used for XQuery transformation and validation scenarios.

- **Enable Optimizations ("-opt")**
This option is selected by default, which means that optimization is enabled. If not selected, the optimization is suppressed, which is helpful when reducing the compiling time is important, optimization conflicts with debugging, or optimization causes extension functions with side-effects to behave unpredictably.

**Use linked tree model ("-tree:linked")**

This option activates the linked tree model.

**Strip whitespaces ("-strip")**

Specifies how the *strip whitespaces* operation is handled. You can choose one of the following values:

- **All ("all")** - Strips all whitespace text nodes from source documents before any further processing, regardless of any `@xml:space` attributes in the source document.
- **Ignore ("ignore")** - Strips all *ignoreable* whitespace text nodes from source documents before any further processing, regardless of any `@xml:space` attributes in the source document. Whitespace text nodes are ignoreable if they appear in elements defined in the DTD or schema as having element-only content.
- **None ("none")** - Strips no whitespace before further processing.

**Enable profiling ("-TP")**

If selected, profiling of the execution time in a query is enabled. The corresponding text field is used to specify the path to the output file where the profiling information will be saved. As long as the option is selected, and the output file specified, it will gather timed tracing information and create a profile report to the specified file.

**Note:**

The profiling support works only if the Present as a sequence transformation option (on page 1486) is not set.

**Saxon-PE/EE Options**

The following advanced options are specific for Saxon 10.6 Professional Edition (PE) and Enterprise Edition (EE) only:

**Allow calls on extension functions ("-ext")**

If selected, calls on external functions are allowed. Selecting this option is not recommended in an environment where untrusted stylesheets may be executed. It also disables user-defined extension elements and the writing of multiple output files, both of which carry similar security risks.

**Saxon-EE Options**

The advanced options that are specific for Saxon 10.6 Enterprise Edition (EE) are as follows:
Validation of the source file ("-val")

Requests schema-based validation of the source file and of any files read using `document()` or similar functions. It can have the following values:

- **Schema validation** ("strict") - This mode requires an XML Schema and allows for parsing the source documents with strict schema-validation enabled.
- **Lax schema validation** ("lax") - If an XML Schema is provided, this mode allows for parsing the source documents with schema-validation enabled but the validation will not fail if, for example, element declarations are not found.
- **Disable schema validation** - This specifies that the source documents should be parsed with schema-validation disabled.

Validation errors in the result tree treated as warnings ("-outval")

Normally, if validation of result documents is requested, a validation error is fatal. Selecting this option causes such validation failures to be treated as warnings.

**Write comments for non-fatal validation errors of the result document**

The validation messages for non-fatal errors are written (wherever possible) as a comment in the result document itself.

Enable XQuery update ("-update:(on|off)")

This option controls whether or not XQuery update syntax is accepted. The default value is off.

Backup files updated by XQuery ("-backup:(on|off)")

If selected, backup versions for any XML files updated with an XQuery Update are generated. This option is available when the Enable XQuery update option is selected.

Other Options

**Initializer class**

Equivalent to the `-init` Saxon command-line argument. The value is the name of a user-supplied class that implements the `net.sf.saxon.lib.Initializer` interface. This initializer is called during the initialization process, and may be used to set any options required on the configuration programmatically. It is particularly useful for tasks such as registering extension functions, collations, or external object models, especially in Saxon-HE where the option cannot be set via a configuration file. Saxon only calls the initializer when running from the command line, but the same code may be invoked to perform initialization when running user application code.
FO Processor Tab (XQuery Transformations)

When you create a new transformation scenario (on page 1464) or edit an existing one (on page 1545), a configuration dialog box is displayed that allows you to customize the transformation with various options in several tabs.

The FO Processor tab contains the following options:

**Perform FO Processing**
- Specifies whether or not an FO processor is applied (either the built-in Apache FOP engine or an external engine defined in Preferences) during the transformation.

**Input**
- Choose between the following options to specify which input file to use:
  - **XQuery result as input** - The FO processor is applied to the result of the XQuery transformation that is defined in the XQuery tab.
  - **XML URL as input** - The FO processor is applied to the input XML file.

**Method**
- The output format of the FO processing. The available options depend on the selected processor type.

**Processor**
- Specifies the FO processor to be used for the transformation. It can be the built-in Apache FOP processor or an external processor (on page 261).

Output Tab (XQuery Transformations)

When you create a new transformation scenario (on page 1464) or edit an existing one (on page 1545), a configuration dialog box is displayed that allows you to customize the transformation with various options in several tabs.

The Output tab contains the following options:

**Present as a sequence**
- Selecting this option will reduce the time necessary to fetch the full results, as it will only fetch the first chunk of the results.

**Prompt for file**
- At the end of the transformation, a file browser dialog box is displayed for specifying the path and name of the file that stores the transformation result.

**Save As**
- The path of the file where the result of the transformation is stored. You can specify the path by using the text field, the 📝 Insert Editor Variables (on page 323) button, or the 🗄️ Browse button.
Open in Browser/System Application

If selected, Oxygen XML Editor automatically opens the result of the transformation in a system application associated with the file type of the result (for example, in Windows PDF files are often opened in Acrobat Reader).

Note:
To set the web browser that is used for displaying HTML/XHTML pages, go to Options > Preferences > Global, and set it in the Default Internet browser field.

- **Output file** - When Open in Browser/System Application is selected, you can use this button to automatically open the default output file at the end of the transformation.
- **Other location** - When Open in Browser/System Application is selected, you can use this option to open the file specified in this field at the end of the transformation. You can specify the path by using the text field, the Insert Editor Variables (on page 323) button, or the Browse button.

Open in editor

When this is option is selected, at the end of the transformation, the default output file is opened in a new editor panel with the appropriate built-in editor type (for example, if the result is an XML file it is opened in the built-in XML editor, or if it is an XSL-FO file it is opened with the built-in FO editor).

Show in results view as

You can choose to view the results in one of the following:

- **XML** - If this is selected, Oxygen XML Editor displays the transformation result in an XML viewer panel at the bottom of the application window with syntax highlighting (on page 224).
- **SVG** - If this is selected, Oxygen XML Editor displays the transformation result in an integrated SVG viewer in the Results panel (on page 1254) at the bottom of the application window and renders the result as an SVG image.
- **XHTML** - This option is only available if Open in Browser/System Application is not selected. If selected, Oxygen XML Editor displays the transformation result in a built-in XHTML browser panel at the bottom of the application window.

Important:
When transforming very large documents, you should be aware that selecting this option may result in very long processing times. This drawback is due to the built-in Java XHTML browser implementation. To avoid delays for large documents, if you want to see the XHTML result of the transformation, you should use an
• **Image URLs are relative to** - If **Show in results view as XHTML** is selected, this option specifies the path used to resolve image paths contained in the transformation result. You can specify the path by using the text field, the **Insert Editor Variables** button, or the **Browse** button.

### SQL Transformation

This type of transformation specifies a database connection for the database server that runs the SQL file associated with the scenario. The data processed by the SQL script is located in the database.

To create an **SQL transformation** scenario, use one of the following methods:

- Use the **Configure Transformation Scenario(s)** (Ctrl + Shift + C (Command + Shift + C on macOS)) action from the toolbar or the **Document > Transformation** menu. Then click the **New** button and select **SQL transformation**.
- Go to **Window > Show View** and select **Transformation Scenarios** to display this view (on page 1555). Click the **New Scenario** drop-down menu button and select **SQL transformation**.

Both methods open the **New Scenario** dialog box. This dialog box allows you to configure the following options that control the transformation:

**Name**

The unique name of the SQL transformation scenario.

**Storage**

Allows you to select one of the following storage options:

- **Project Options** (on page 3222) - The scenario is stored in the project file and can be shared with other users. For example, if your project is saved on a source versioning/sharing system (CVS, SVN, Source Safe, etc.) or a shared folder, your team can use the scenarios that you store in the project file.
- **Global Options** (on page 3219) - The scenario is saved in the global options that are stored in the user home directory and is not accessible to other users.

**SQL URL**

Allows you to specify the URL of the SQL script. You can specify the path by using the text field, its history drop-down, the **Insert Editor Variables** button, or the browsing actions in the **Browse** drop-down list. You can also use the **Open in editor** button to open the specified file in the editor panel.
Allows you to select a connection from a drop-down list. To configure a connection, use the Advanced options button to open the Data Source preferences page (on page 276).

Parameters

Allows you to add or configure parameters for the transformation.

Editing a Transformation Scenario

Editing a transformation scenario is useful if you need to configure some of its parameters.

Note:

Since transformation scenarios that are associated with built-in frameworks (on page 3219) are read-only, to edit one of these scenarios you will need to duplicate it and edit the duplicated scenario (on page 1547).

To configure an existing transformation scenario, follow these steps:

1. Select the Configure Transformation Scenario(s) (Ctrl + Shift + C (Command + Shift + C on macOS)) action from the toolbar or the Document > Transformation menu.

   **Step Result:** The Configure Transformation Scenario(s) dialog box (on page 1548) is opened.

2. Select the particular transformation scenario and click the Edit button at the bottom of the dialog box or from the contextual menu.

   **Tip:** You could also select the scenario and the Edit button in the Transformation Scenarios view (on page 1555) to achieve the same result.

   **Result:** This will open an Edit scenario configuration dialog box that allows you to configure various options in several tabs, depending on the type of transformation scenario that was selected.
Transformation Types

The **Configure Transformation Scenario(s)** dialog box (on page 1548) contains a **Type** column that shows you the transformation type for each of the listed scenarios. Each type of transformation contains some tabs with various configuration options.

The following is a list of the transformation types and their particular tabs (click the name of each tab below to see details about all the options that are available):

- **DITA-OT** - This type of transformation includes configurable options in the following tabs:
  - Templates Tab (on page 3106)
  - FO Processor Tab (on page 3111) (Available for PDF output)
  - Parameters Tab (on page 3112)
  - Filters Tab (on page 3114)
  - Advanced Tab (on page 3116)
  - Output Tab (on page 3118)

- **ANT** - This type of transformation includes configurable options in the following tabs:
  - Options Tab (on page 1507)
  - Parameters Tab (on page 1509)
  - Output Tab (on page 1510)

- **XSLT** - This type of transformation includes configurable options in the following tabs:
  - XSLT Tab (on page 1464)
  - FO Processor Tab (on page 1473)
  - Output Tab (on page 1474)

- **XProc** - This type of transformation includes configurable options in the following tabs:
  - XProc Tab (on page 1531)
  - Inputs Tab (on page 1531)
  - Parameters Tab (on page 1532)
• Outputs Tab (on page 1532)
• Options Tab (on page 1534)

• XQuery - This type of transformation includes configurable options in the following tabs:
  ◦ XQuery Tab (on page 1480)
  ◦ FO Processor Tab (on page 1485)
  ◦ Output Tab (on page 1486)

Related Information:
Creating New Transformation Scenarios (on page 1464)
Duplicating a Transformation Scenario (on page 1547)
Configure Transformation Scenario(s) Dialog Box (on page 1548)
Applying Associated Transformation Scenarios (on page 1548)

Duplicating a Transformation Scenario

Duplicating a transformation scenario is useful for creating a scenario that is similar to an existing one or to edit a built-in transformation scenario.

To configure an existing transformation scenario, follow these steps:

1. Select the Configure Transformation Scenario(s) (Ctrl + Shift + C (Command + Shift + C on macOS)) action from the toolbar or the Document > Transformation menu.

   **Step Result:** The Configure Transformation Scenario(s) dialog box (on page 1548) is opened.

2. Select the particular transformation scenario and click the Duplicate button at the bottom of the dialog box or from the contextual menu.

   **Tip:**
   You could also select the scenario and the Duplicate button in the Transformation Scenarios view (on page 1555) to achieve the same result.

   **Result:** This will open an Edit scenario configuration dialog box that allows you to configure various options in several tabs, depending on the type of transformation scenario that was selected. For information about all the specific options in the various tabs, see the Transformation Types section (on page 1546).

Related Information:
Creating New Transformation Scenarios (on page 1464)
Editing a Transformation Scenario (on page 1545)
Applying Associated Transformation Scenarios

If you have associated transformation scenarios for the current document (in the Configure Transformation Scenario(s) dialog box (on page 1548) or Transformation Scenarios view (on page 1555)), Oxygen XML Editor included an Apply Transformation Scenario(s) action that allows you to quickly apply the associated transformation scenarios on the current document. Note that if an association is not detected, this action will open the Configure Transformation Scenario(s) dialog box (on page 1548) where you can choose the scenarios you want to apply.

The Apply Transformation Scenario(s) action can be initiated from any of the following methods:

- Use the Ctrl + Shift + T (Command + Shift + T on macOS) keyboard shortcut.
- Click the Apply Transformation Scenario(s) button on the main toolbar.
- Click the Apply Transformation Scenario(s) button on the toolbar in the Transformation Scenarios view (on page 1555).
- Right-click a file in the Project view (on page 402) and select Transform > Apply Transformation Scenario(s).
- Use the Apply Associated button in the Configure Transformation Scenario(s) dialog box (on page 1548).

Related Information:
- Creating New Transformation Scenarios (on page 1464)
- Editing a Transformation Scenario (on page 1545)
- Configure Transformation Scenario(s) Dialog Box (on page 1548)
- Transformation Scenarios View (on page 1555)

Configure Transformation Scenario(s) Dialog Box

You can use the Configure Transformation Scenario(s) dialog box for editing existing transformation scenarios (on page 1545) or creating new ones (on page 1464).

To open this dialog box, use the Configure Transformation Scenario(s) (Ctrl + Shift + C (Command + Shift + C on macOS)) action from the toolbar or the Document > Transformation menu.
The dialog box includes the following options and features:

**Search Filter Field**

You can begin typing text in the search field at the top of the dialog box to filter the scenarios shown in the table below this field.

**Settings**

Use this drop-down to access the following options:

**Show all scenarios**

Select this option to display all the available scenarios, regardless of the document they are associated with.

**Show only the scenarios available for the editor**

Select this option to only display the scenarios that Oxygen XML Editor can apply for the current document type.

**Show associated scenarios**

Select this option to only display the scenarios associated with the document you are editing.

**Import scenarios**
This option opens the **Import scenarios** dialog box that allows you to select the scenarios file that contains the scenarios you want to import. If one of the scenarios you import is identical to an existing scenario, Oxygen XML Editor ignores it. If a conflict appears (an imported scenario has the same name as an existing one), you can choose between two options:

- Keep or replace the existing scenario.
- Keep both scenarios.

**Note:**

When you keep both scenarios, Oxygen XML Editor adds **imported** to the name of the imported scenario.

**Export selected scenarios**

Use this option to export selected scenarios individually. Oxygen XML Editor creates a scenarios file that contains the exported scenarios. This is useful if you want to share scenarios with others or export them to another computer.

**Scenarios Table Section**

The middle section of the dialog box is a table that displays the scenarios that you can apply to the current document. You can view both the scenarios associated with the current document type and the scenarios defined at project level *(on page 3222)*. The table includes following sortable columns:

- **Association** - The checkboxes in this column mark whether or not a transformation scenario is associated with the current document.
- **Scenario** - This column presents the names of the transformation scenarios.
- **Type** - If the **Show Type** contextual menu option is selected, this column displays the type of the transformation scenario. For further details about the types of transformation scenarios that are available in Oxygen XML Editor, see the Transformation Types section *(on page 1546)*.

If you right-click in the header area, the following options are accessible:

**Show Type**

Use this option to display the transformation type of each scenario.

**Show Storage**

Use this option to display the storage location of the scenarios.

**Group by Association**

Select this option to group the scenarios depending on whether or not they are associated with the current document.
Group by Type

Select this option to group the scenarios by their type.

Group by Storage

Select this option to group the scenarios by their storage location.

Ungroup all

Select this option to ungroup all the scenarios.

Reset Layout

Select this option to restore the default settings of the layout.

If you right-click any particular transformation scenario, the following actions are accessible:

Edit

This button opens the Edit Scenario configuration dialog box (on page 1545) that allows you to configure the options of the transformations scenario.

Duplicate

Use this button to create a duplicate transformation scenario (on page 1547).

Remove

Use this button to remove custom transformation scenarios.

Change storage

Allows you to change the storage location of a transformation scenario to Project Options (on page 3222) or Global Options (on page 3219). You are also able to keep the original storage location and make a copy of the selected scenario in the new storage location.

Import scenarios

This option opens the Import scenarios dialog box that allows you to select the scenarios file that contains the scenarios you want to import. If one of the scenarios you import is identical to an existing scenario, Oxygen XML Editor ignores it. If a conflict appears (an imported scenario has the same name as an existing one), you can choose between two options:

- Keep or replace the existing scenario.
- Keep both scenarios.

Note:

When you keep both scenarios, Oxygen XML Editor adds imported to the name of the imported scenario.

Export selected scenarios
Use this option to export selected scenarios individually. Oxygen XML Editor creates a scenarios file that contains the exported scenarios. This is useful if you want to share scenarios with others or export them to another computer.

**Bottom Section**

The bottom section of the dialog box contains the following actions and options:

**New**

This button allows you to create a new transformation scenario (on page 1464).

**Edit**

This button opens the Edit Scenario dialog box that allows you to configure the options of the transformations scenario. For information about all the specific options in the various tabs, see the Transformation Types section (on page 1546).

*Note:*

If you try to edit a transformation scenario associated with a defined document type, Oxygen XML Editor displays a warning message to inform you that this is not possible and gives you the option to create a duplicate transformation scenario (on page 1547) to edit instead.

**Duplicate**

Use this button to create a duplicate transformation scenario (on page 1547).

**Remove**

Use this button to remove transformation scenarios.

*Note:*

Removing scenarios associated with a defined document type is not allowed.

**Association follows selection**

Select this checkbox to automatically associate selected transformation scenarios with the current document. This option can also be used for multiple selections.

*Note:*

When this option is selected, the Association column is hidden.

**Run in parallel (DITA-OT or Ant scenarios)**

This option is available if you select multiple DITA-OT or Ant type scenarios. Selecting this option results in the transformations being done in parallel, instead
of sequentially. It should help to reduce the amount of time it takes for the 
publishing to finish when transforming large projects.

**Attention:**
If multiple selected DITA-OT scenarios have the same output or temporary 
files folder, this option is not available since the process would need to 
read and write content to the same folder in this case.

**Associated scenarios section**
Displays the scenarios that are associated with the current document. Selecting a 
checkbox in the **Association** column in the list of scenarios will add that scenario 
to this section. To remove a scenario from being associated with the current 
document, simply click the remove icon (×) to the right of the scenario name.

**Save and close**
Saves the current configuration and closes the dialog box.

**Apply associated**
Use this button to apply the associated scenarios and run the transformation on 
the current document.

**Cancel**
Cancels any changes made in the dialog box and reverts to the previously saved 
association.

**Tip:**
Your selections in the Configure Transformation Scenarios(s) dialog box are persistent so the 
configured associations for the current document will be remembered after the dialog box is closed.

**Related Information:**
- Editing a Transformation Scenario (on page 1545)
- Duplicating a Transformation Scenario (on page 1547)
- Applying Associated Transformation Scenarios (on page 1548)
- Creating New Transformation Scenarios (on page 1464)
- Sharing Transformation Scenarios (on page 1555)

**Batch Transformations**
A transformation action can be applied on a batch of selected files from the contextual menu of the Project 
view (on page 408) without having to open the files involved in the transformation. You can apply the same 
scenario to a batch of files or multiple scenarios to a single file or batch of files.
1. (Optional, but recommended) Organize the files you want to transform in logical folders.

   a. Create a logical folder in the **Project view (on page 402)** by using the **New > Logical Folder** action from the contextual menu of the root file.

   b. Add the files you want to transform to the logical folder by using the **Add Files** or **Add Edited File** actions from the contextual menu of the logical folder.

   ![Note:](image)
   
   You can skip this step if the files are already in a dedicated folder that does not include any additional files or folders. You can also manually select the individual files in the **Project view (on page 402)** each time you want to transform them, but this can be tedious.

2. Select the files you want to transform (or the newly created logical folder) and from the contextual menu, select **Transform > Configure Transformation Scenario(s)** to choose one or more transformation scenarios to be applied on all the files in the logical folder.

3. Use Oxygen XML Editor **editor variables (on page 323)** to specify the input and output files. This ensures that each file from the selected set of resources is processed and that the output is not overwritten by the subsequent processing.

   a. Edit the transformation scenario to make sure the appropriate **editor variable (on page 323)** is assigned for the input file. For example, for a DocBook PDF transformation, make sure the **XML URL** input box is set to the **${currentFileURL}** editor variable (on page 329). For a DITA PDF transformation, make sure the **args.input** parameter is set to the **${cf}** editor variable (on page 329).

   b. Edit the transformation scenario to make sure the appropriate editor variable is assigned for the output file. For example, for an **XML transformation with XSLT**, switch to the **Output** tab and set the path of the output file using a construct of **editor variables (on page 323)**, such as **${cfd}/${cfn}.html**.

4. Now that the logical folder has been associated with one or more transformation scenarios, whenever you want to apply the same batch transformation, you can select **Transform > Transform with** from the contextual menu and the same previously associated scenario(s) will be applied.

5. If you want a different type of transformation to be applied to each file inside the logical folder, associate individual scenarios for each file and select **Transform > Apply Transformation Scenario(s)** from the contextual menu of the logical folder.

**Related Information:**

**Editor Variables (on page 323)**
Sharing Transformation Scenarios

The transformation scenarios and their settings can be shared with other users by saving them at project level (on page 3222) or by exporting them to a specialized scenarios file (on page 323) that can then be imported. When you create a new transformation scenario or edit an existing one, there is a Storage option to control whether the scenarios are stored in Project Options (on page 3222) or Global Options (on page 3219).

Selecting Project Options (on page 3222) stores the scenario in the project file and can be shared with other users that have access to the project. If your project is saved on a source versioning system (CVS, SVN, Source Safe, etc.) then your team will have access to the scenarios that you define. When you create a scenario at the project level, the URLs from the scenario become relative to the project URL.

Selecting Global Options (on page 3219) stores the scenario in the global options that are stored in the user home directory.

You can also change the storage options on existing transformation scenarios by using the Change storage action (on page 1548) from the contextual menu of the list of scenarios.

Related Information:
Sharing Application Settings (on page 312)

Transformation Scenarios View

You can manage the transformation scenarios by using the Transformation Scenarios view. To open this view, select Window > Show View > Transformation Scenarios.
Oxygen XML Editor supports multiple scenarios association. To associate multiple scenarios with a document, select the checkboxes in front of each scenario. You can also associate multiple scenarios with a document from the **Configure Transformation Scenario(s)** dialog box (on page 1548).

The **Transformation Scenarios** view presents both global and project-level (on page 3222) scenarios. By default, Oxygen XML Editor presents the items in the following order:

1. Scenarios that match the current framework (on page 3219).
2. Scenarios that match the current project.
3. Scenarios that match other frameworks.

**Toolbar/Contextual Menu Actions and Options**

The following actions and options are available on the toolbar or in the contextual menu:

- **Apply selected scenarios**
  
  Select this option to run the current transformation scenario.

- **Debug selected scenario**
Select this option to switch to the **Debugger perspective** *(on page 3221)* and initialize it with the parameters from the scenario (the XML, XSLT, or XQuery input, the transformation engine, the XSLT parameters).

**New**

This drop-down menu contains a list of the **scenarios that you can create** *(on page 1464)*. Oxygen XML Editor determines the most appropriate scenarios for the current type of file and displays them at the beginning of the list, followed by the rest of the scenarios.

**Duplicate**

Adds a new scenario to the list that is a duplicate of the current scenario. It is useful for creating a scenario that is similar to an existing one.

**Edit**

Opens the dialog box that allows you to configure various options in several tabs, depending on the type of transformation scenario that was selected. For information about all the specific options in the various tabs, see the **Transformation Types section** *(on page 1546)*.

**Remove**

Removes the current scenario from the list. This action is also available by using the **Delete** key.

**Change storage**

Use this option to change the storage location of the selected scenario. You are also able to keep the original storage location and make a copy of the selected scenario in the target storage location.

**Import scenarios**

This option opens the **Import scenarios** dialog box that allows you to select the **scenarios** file that contains the scenarios you want to import. If one of the scenarios you import is identical to an existing scenario, Oxygen XML Editor ignores it. If a conflict appears (an imported scenario has the same name as an existing one), you can choose between two options:

- Keep or replace the existing scenario.
- Keep both scenarios.

**Note:**

When you keep both scenarios, Oxygen XML Editor adds **imported** to the name of the imported scenario.

**Export selected scenarios**

Use this option to export transformation and validation scenarios individually. Oxygen XML Editor creates a **scenarios** file that contains the scenarios that you export.

**Settings**
This drop-down menu allows you to configure the following options (many of these options are also available if you right-click the name of a column):

**Show all scenarios**
Select this option to display all the available scenarios, regardless of the document they are associated with.

**Show only the scenarios available for the editor**
Select this option to only display the scenarios that Oxygen XML Editor can apply for the current document type.

**Show associated scenarios**
Select this option to only display the scenarios associated with the document you are editing.

**Change storage**
Use this option to change the storage location of the selected scenario to **Project Options (on page 3222)** or **Global Options (on page 3219)**. You are also able to keep the original storage location and make a copy of the selected scenario in the new storage location.

**Import scenarios**
This option opens the **Import scenarios** dialog box that allows you to select the scenarios file that contains the scenarios you want to import. If one of the scenarios you import is identical to an existing scenario, Oxygen XML Editor ignores it. If a conflict appears (an imported scenario has the same name as an existing one), you can choose between two options:

- Keep or replace the existing scenario.
- Keep both scenarios.

**Note:**
When you keep both scenarios, Oxygen XML Editor adds **imported** to the name of the imported scenario.

**Export selected scenarios**
Use this option to export selected scenarios individually. Oxygen XML Editor creates a scenarios file that contains the exported scenarios. This is useful if you want to share scenarios with others or export them to another computer.

**Show Type**
Use this option to display the transformation type of each scenario.

**Show Storage**
Use this option to display the storage location of the scenarios.

**Group by Association**
Select this option to group the scenarios depending on whether or not they are associated with the current document.

**Group by Type**
Select this option to group the scenarios by their type.

**Group by Storage**
Select this option to group the scenarios by their storage location.

**Ungroup all**
Select this option to ungroup all the scenarios.

**Reset Layout**
Select this option to restore the default settings of the layout.

Your selections in the Transformation Scenarios view are persistent so the configured associations for the current document will be remembered whenever the document is opened.

**Related Information:**
- Editing a Transformation Scenario *(on page 1545)*
- Creating New Transformation Scenarios *(on page 1464)*

**WebHelp Output Customization**

**Oxygen XML WebHelp** provides the ability to generate two different types of output, **WebHelp Responsive** and **WebHelp Classic**. Each type has its own set of options and features. The **WebHelp Responsive** variant is available for DITA documents while the **WebHelp Classic** variants are available for DocBook.

<table>
<thead>
<tr>
<th>Table 38. WebHelp System Feature Matrix</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Features</strong></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Desktop Systems</td>
</tr>
<tr>
<td>Mobile Devices</td>
</tr>
<tr>
<td>Built-in Skins</td>
</tr>
<tr>
<td>Built-in Templates</td>
</tr>
<tr>
<td>Search Capabilities</td>
</tr>
<tr>
<td>Modern Layout</td>
</tr>
</tbody>
</table>
Table 38. WebHelp System Feature Matrix (continued)

<table>
<thead>
<tr>
<th>Features</th>
<th>WebHelp System Variants</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Responsive</td>
</tr>
<tr>
<td>Adaptable to Any Screen Size</td>
<td>✓</td>
</tr>
<tr>
<td><strong>Oxygen Feedback</strong> Commenting Platform</td>
<td>✓</td>
</tr>
<tr>
<td>DITA Documents</td>
<td>✓</td>
</tr>
<tr>
<td>DocBook Documents</td>
<td>✓</td>
</tr>
<tr>
<td>Tri-Pane Frames or Frameless Version</td>
<td>✓</td>
</tr>
</tbody>
</table>

**WebHelp Responsive Output for DITA**

WebHelp Responsive features a very flexible layout, is designed to adapt to any screen size to provide an optimal viewing and interaction experience. It is based upon the Bootstrap responsive front-end framework and is available for DITA document types.

WebHelp Responsive output can be generated by using the DITA Map WebHelp Responsive (on page 1435) transformation scenario.

For an in-depth look at WebHelp Responsive features and the publishing process, watch our Webinar: DITA Publishing and Feedback with Oxygen Tools.

**Layout and Features**

This section contains information about the layout and features of the WebHelp Responsive output.

**Layout of the Responsive Page Types**

You can select from several different styles of layouts (for example, by default, you can select either a *tiles* or *tree* style of layout). Furthermore, each layout includes a collection of skins that you can choose from, or you can customize your own.
Figure 482. WebHelp Responsive Output on a Normal Screen
Main Page

The **Main Page** is the home page generated in the WebHelp Responsive output. The main function of the home page is to display top-level information and provide links that help you easily navigate to any of the top-level topics of the publication. These links can be rendered in either a *Tiles* or *Tree* style of layout. The main page also consists of various other components, such as a logo, title, menu, search field, or index link.

Main Page - Tiles Layout

In the *tiles* presentation mode, a tile component is created for each chapter (first-level topic) in the publication. The tile presents a link to the topic and its short description.
Figure 484. Main Page - Tiles Layout

1. Logo Component (on page 1564)
2. Title Component (on page 1564)
3. Search Input Component (on page 1565)
4. Menu Component (on page 1565)
5. Index Terms Link Component (on page 1565)
6. Topic Tiles Component (on page 1565)
7. Footer Component (on page 1565)

Main Page - Tree Layout

In the tree presentation mode, links to the first and second level topics in the publication are displayed using a tree-like component.
Figure 485. Main Page - Tree Layout

1. Logo Component (on page 1564)
2. Title Component (on page 1564)
3. Search Input Component (on page 1565)
4. Menu Component (on page 1565)
5. Index Terms Link Component (on page 1565)
6. Table of Contents Component (on page 1565)
7. Footer Component (on page 1565)

Main Page Components

The layout components displayed in the main page are:

Publication Title

The title of the publication. It is usually taken from the DITA map title.

Logo

Displays a logo associated with the publication. Additionally, you can set a target URL that will be opened when you click on the logo image.
The logo image can be specified using the `webhelp.logo.image` transformation parameter (on page 1718). For the target URL, use the `webhelp.logo.image.target.url` parameter (on page 1718).

**Menu**

Helps you to navigate to your documentation. This component presents a set of links to all topics from your publication. For information about customizing the menu, see How to Customize the Menu (on page 1665) topic.

**Index Terms Link**

Presents a link to the index terms page. You can control if this component is displayed by using the `webhelp.show.indexterms.link` parameter (on page 1727).

**Search Input**

An input text field where you can enter search queries.

**Topic Tiles**

A tile associated with a main topic. Each topic tile has three sections that correspond to the topic title, short description, and image.

**Topic Tile Title**

Presents the navigation title of the associated topic.

**Topic Tile Short Description**

Presents the short description of the topic. It may be collected either from the topic or from the DITA map topic meta.

**Topic Tile Image**

Presents an image associated with the topic. The image association (on page 1665) is done in the DITA map.

**Tree Table of Contents**

An area that contains first and second-level topic titles from your publication.

**Page Footer**

WebHelp Responsive output footer.

**Topic Page**

The *Topic Page* is the page generated for each DITA topic in the WebHelp Responsive output. The HTML pages produced for each topic consist of the topic content along with various other additional components, such as a title, menu, navigation breadcrumb, print icon, or side table of contents.
Figure 486. Topic Page

1. Logo Component (on page 1567)
2. Title Component (on page 1566)
3. Search Input Component (on page 1567)
4. Menu Component (on page 1567)
5. Index Terms Link Component (on page 1567)
6. Expand/Collapse All Sections Component (on page 1567)
7. Navigation Links Component (on page 1567)
8. Print Link Component (on page 1567)
9. Breadcrumb Component (on page 1567)
10. Publication Table of Contents Component (on page 1568)
11. Topic Content Component (on page 1567)
12. Topic Table of Contents Component (on page 1568)

**Topic Page Components**

The layout components displayed in this page are:

**Publication Title**

The title of the publication. It is usually taken from the DITA map title.
Logo

Displays a logo associated with the publication. Additionally, you can set a target URL that will be opened when you click on the logo image.

The logo image can be specified using the `webhelp.logo.image` transformation parameter (on page 1718). For the target URL, use the `webhelp.logo.image.target.url` parameter (on page 1718).

Menu

Helps you to navigate to your documentation. This component presents a set of links to all topics from your publication. For information about customizing the menu, see How to Customize the Menu (on page 1665) topic.

Index Terms Link

Presents a link to the index terms page. You can control if this component is displayed by using the `webhelp.show.indexterms.link` parameter (on page 1727).

Search Input

An input text field where you can enter search queries.

Navigation Links

The navigation links (Previous / Next arrows) can be used to navigate to the previous or next topic. These navigation links are controlled by the `collection-type` attribute. For example, if you set `collection-type="sequence"` on a parent topic reference, navigation links will be generated in the output for that topic and all of its child topics. You can also use the `webhelp.default.collection.type.sequence` parameter and set its value to `yes` to generate navigation links for all topics, regardless of whether or not the `collection-type` attribute is present.

Tip:

To hide the navigation links, you can edit the transformation scenario and set the value of the `webhelp.show.navigation.links` parameter to `no`.

Expand/Collapse Sections Button

Icon that expands or collapses sections listed in the side table of contents within a topic.

Print Link

A print icon that opens the print dialog box for your particular browser.

Breadcrumb

Presents the path of the current displayed DITA topic.

Topic Content

Presents the content of the associated DITA topic.
Publication Table of Contents

A Table of Content for the publication displayed in the left side of the screen. You can use the button to collapse the table of contents (or the button to expand it).

Topic Table of Contents (On this page links)

A table of contents for the topic displayed on the right side with a heading named On this page and it contains links to each section within the current topic and the section corresponding to the current scroll position is highlighted. This component is generated for any topic that contains at least two `<section>` elements and each `<section>` must have an `@id` attribute. You can use the button to collapse the table of contents (or the button to expand it).

Page Footer

WebHelp Responsive output footer.

Search Page

The Search Page presents search results in the WebHelp Responsive output. The HTML page consists of a search results component along with various other additional components, such as a title, menu, or index link.

When you enter search terms in the Search field, the results are displayed in a results page. When you click on a result, the topic is opened in the main pane and the search results are highlighted. If you want to remove the colored highlights, click the Toggle Highlights button at the top-right side of the page. The Search field also includes an autocomplete feature.
Figure 487. Search Results Page

1. Logo Component (on page 1569)
2. Title Component (on page 1569)
3. Search Input Component (on page 1570)
4. Menu Component (on page 1570)
5. Index Terms Link Component (on page 1570)
6. Search Results Component (on page 1570)
7. Footer Component (on page 1570)

Search Page Components

The layout components displayed in the search page are:

Publication Title

The title of the publication. It is usually taken from the DITA map title.

Logo

Displays a logo associated with the publication. Additionally, you can set a target URL that will be opened when you click on the logo image.
The logo image can be specified using the `webhelp.logo.image` transformation parameter (on page 1718). For the target URL, use the `webhelp.logo.image.target.url` parameter (on page 1718).

**Menu**

Helps you to navigate to your documentation. This component presents a set of links to all topics from your publication. For information about customizing the menu, see How to Customize the Menu (on page 1665) topic.

**Index Terms Link**

Presents a link to the index terms page. You can control if this component is displayed by using the `webhelp.show.indexterm.link` parameter (on page 1727).

**Search Input**

An input text field where you can enter search queries.

**Search Results**

Each result includes the topic title that can be clicked to open that page. Under the title, a breadcrumb is displayed that shows the path of the topic and you can click any of the topics in the breadcrumb to open that particular page.

**Page Footer**

WebHelp Responsive output footer.

**Auto-complete Suggestions in the Search Text Field**

When you are typing in the search text field, proposals are presented to help you to compute the search query. The information proposed when you are typing is collected from:

- The search queries from the history of the previous searches.
- The titles collected from your documentation.
- Documentation index terms and keywords. For example, in a DITA topic, the keywords and index terms are specified in the topic prolog section like this:

```xml
<prolog>
  <metadata>
    <keywords><indexterm>databases</indexterm></keywords>
    <keyword>installing</keyword>
    <keyword>uninstalling</keyword>
    <keyword>prerequisites</keyword>
  </metadata>
</prolog>`
**Missing Terms**

If you enter multiple search terms (other than *stop words*), for any result that the search engine found at least one term but not one or more of the other terms, the **Missing terms** will be listed below each result.

**Related information**

WebHelp Responsive Search Engine *(on page 1572)*

**Index Terms Page**

The **Index Terms Page** page consists of an index terms section along with various other additional components, such as a title, menu, or search field.

An alphabet that contains the first letter of the documentation index terms is generated at the top of the index page. Each letter represents a link to a specific indices section. The indexes are presented in multiple columns to make it easier to read this page.

**Figure 488. Index Terms Page**

1. Logo Component *(on page 1572)*
2. Title Component *(on page 1572)*
3. Menu Component *(on page 1572)*
4. Index Terms Link Component *(on page 1572)*
5. Index Terms Component *(on page 1572)*
Index Terms Page Components

The layout components displayed in this page are:

**Publication Title**

The title of the publication. It is usually taken from the DITA map title.

**Logo**

Displays a logo associated with the publication. Additionally, you can set a target URL that will be opened when you click on the logo image.

The logo image can be specified using the `webhelp.logo.image` transformation parameter. For the target URL, use the `webhelp.logo.image.target.url` parameter.

**Menu**

Helps you to navigate to your documentation. This component presents a set of links to all topics from your publication. For information about customizing the menu, see How to Customize the Menu topic.

**Index Terms Link**

Presents a link to the index terms page. You can control if this component is displayed by using the `webhelp.show.indexterms.link` parameter.

**Index Terms Alphabet**

An alphabet that contains the first letter of index terms. Each letter represents a link to a specific indices section.

**Index Terms**

The first letter of the index along with the list of index terms.

**Page Footer**

WebHelp Responsive output footer.

Search Engine

Search engine has two main components:

**Search indexer**

It is also known as a **spider**. This component is active when you publish your documentation to WebHelp and it is responsible for creating the **search index**. This component traverses all HTML pages (for DITA topics) to gather information.

**Search interface**
This component is an interface between the user and the *search index*. It helps the user to search through the *search index* and displays results in the search page.

**Search Field and Results Page**

When you enter search terms in the **Search** field, the results are displayed in a results page. When you click on a result, the topic is opened in the main pane and the search results are highlighted. If you want to remove the colored highlights, click the **Toggle Highlights** button at the top-right side of the page. The **Search** field also includes an *autocomplete* feature.

Each result includes the topic title that can be clicked to open that page. Under the title, a breadcrumb is displayed that shows the path of the topic and you can click any of the topics in the breadcrumb to open that particular page.

If you enter multiple search terms (other than *stop words*), for any result that the search engine found at least one term but not one or more of the other terms, the **Missing** terms will be listed below each result.

**5-Star Rating Mechanism and Sorting**

The **Search** feature is also enhanced with a rating mechanism that computes scores for every result that matches the search criteria. These scores are then translated into a 5-star rating scheme and the stars are displayed to the right of each result. The search results are sorted depending on the following:

- Search entries that satisfy the phrase search criterion are presented first.
- The number of keywords found in a single page (the higher the number, the better).
- The context (for example, a word found in a title, scores better than a word found in unformatted text).

The search ranking order, sorted by relevance is as follows:

- The search term is included in a meta keyword.
- The search term is in the title of the page.
- The search term is in bold text in a paragraph.
- The search term is in normal text in a paragraph.

**Tag Element Scoring Values**

HTML tag elements are also assigned a scoring value and these values are evaluated for the search results. For information about editing these values, see *How to Change Element Scoring in Search Results (on page 1676)*.

**Search Rules**

Rules that are applied during a search include:
• You can use quotes to perform an exact search for multiple word phrases (for example, "grow flowers" will only return results if both words are found consecutively and exactly as they are typed in the search field). This type of search is known as a phrase search.

• Boolean Search is supported using the following operators: and, or, not. When there are two adjacent search terms without an operator, or is used as the default search operator (for example, grow flowers is the same as grow or flowers).

• The space character separates keywords (an expression such as grow flowers counts as two separate keywords).

• Words composed by merging two or more words with colon (";"), minus ("-"), underline ("_"), or dot ("." ) characters count as a single word.

• Your search terms should contain two or more characters (note that stop words will be ignored). This rule does not apply to CJK (Chinese, Japanese, Korean) languages.

• When searching for multi-word phrases in CJK (Chinese, Japanese, Korean) languages that often have multiple words appear in strings without a space separator, you need to add a space to separate the words. Otherwise, WebHelp will not find results. For example, Chinese uses a specialized character for space separators, but the current WebHelp implementation cannot detect such specialized characters, so to search for 开始之前 (it translates as "before you begin" or "before start"), you have to enter 开始之 前 (notice the space between the second and third symbols) in the search field.

Tip:
The <indexterm> and <keywords> DITA elements are an effective way to increase the ranking of a page (for example, content inside a keywords element weighs more than an H1 HTML element).

Excluded Terms

To improve performance, the Search feature excludes certain stop words. For example, the English version of the stop words includes: a, an, and, are, as, at, be, but, by, for, if, in, into, is, it, no, not, of, on, or, such, that, the, their, then, there, these, they, this, to, was, will, with.

Related Information:
WebHelp Responsive HTML5 Pages: Search Page (on page 1568)

Context-Sensitive Help System

Context-sensitive help systems assist users by providing specific informational topics for certain components of a user interface, such as a button or window. This mechanism works based on mappings between a unique ID defined in the topic and a corresponding HTML page.

Generating Context-Sensitive Help

When WebHelp Responsive output is generated, the transformation process produces an XML mapping file called context-help-map.xml and copies it in the output folder of the transformation. This XML file maps an ID to a corresponding HTML page through an <appContext> element, as in the following example:
The possible attributes are as follows:

**helpID**

A Unique ID provided by a topic from two possible sources (<resourceid> element or @id attribute):

**resourceid**

The <resourceid> element is mapped into the <appContext> element and can be specified in either the <topicref> within a DITA map or in a <prolog> within a DITA topic. The <resourceid> element accepts the following attributes:

- **appname** - A name for the external application that references the topic. If this attribute is not specified, its value is considered to be empty ("").
- **appid** - An ID used by an application to identify the topic.
- **id** - Specifies a value that is used by a specific application to identify the topic, but this attribute is ignored if an @appid attribute is used.

**Note:**

Multiple @appid values can be associated with a single appname value (and multiple @appname values can be associated with a single @appid value), but the values for both attributes work in combination to specify a specific ID for a specific application, and therefore each combination of values for the @appid and @appname attributes should be unique within the context of a single root map (on page 3223). For example, suppose that you need two different functions of an application to both open the same WebHelp page.

**Example: The <resourceid> Element Specified in a DITA Map**

The <resourceid> element can be specified in a <topicmeta> element within a <topicref>.

```xml
<map title="App Help">
  <topicref href="app-help1.dita" type="task">
    <topicmeta>
      <resourceid appname="myapp" appid="functionid1"/>
      <resourceid appname="myapp" appid="functionid2"/>
    </topicmeta>
  </topicref>
</map>
```
Example: The `<resourceid>` Element Specified in a DITA Topic

The `<resourceid>` element can be specified in a `<prolog>` element within a DITA topic.

```xml
<task id="app-help1">
  <title>My App Help</title>
  <prolog>
    <resourceid appname="myapp" appid="functionid1"/>
    <resourceid appname="myapp" appid="functionid2"/>
  </prolog>
  ...
</task>
```

For more information about the `<resourceid>` element, see DITA Specifications: `<resourceid>`.

id

If a `<resourceid>` element is not declared in the DITA map or DITA topic (as described above), the `@id` attribute that is set on the topic root element is mapped into the `<appContext>` element.

Important:

You should ensure that these defined IDs are unique in the context of the entire DITA project. If the IDs are not unique, the transformation scenario will display warning messages in the transformation console output and the help system will not work properly.

path

The path to a corresponding WebHelp page. This path is relative to the location of the `context-help-map.xml` mapping file.

There are two ways of implementing context-sensitive help in your system:

- The XML mapping file can be loaded by a PHP script on the server side. The script receives the `contextId` value and will look it up in the XML file.
- Invoke the `cshelp.html` WebHelp system file and pass the `contextId` parameter with a specific value. The WebHelp system will automatically open the help page associated with the value of the `contextId` parameter.

`cshelp.html?contextId=myDITATopic`
Context-Sensitive Queries

You can use the URL field in your browser to search for topics in a context-sensitive WebHelp system with the assistance of the following parameters:

`contextId`

The WebHelp JavaScript engine will look for this value in the `context-help-map.xml` mapping file and load the corresponding help page.

**Note:**

You can use an anchor (on page 3216) in the `contextId` parameter to jump to a specific section in a document. For example, `contextId=topicID#anchor`.

`appname`

You can use this parameter in conjunction with `contextId` to search for this value in the corresponding `appname` attribute value in the mapping file.

http://localhost/webhelp/cshelp.html?contextId=topicID&appname=myApplication

Another parameter indicates the search query:

`searchQuery`

You can use this parameter to perform a search operation when WebHelp is loaded. For example, if you want to open WebHelp showing all search results for *growing flowers*, the URL should look like this: `http://localhost/webhelp/index.html?searchQuery=growing%20flowers`.

Accessibility

Oxygen XML WebHelp Responsive output is compliant with the Section 508 accessibility standard, making the output accessible for people with visual impairment and other disabilities. Documentation and interface components are considered accessible when they have support in place that allows those with disabilities to use assistive technologies to understand the content.
Generally speaking, the WebHelp Responsive output has two major parts: topic content and WebHelp Responsive-related components (publication TOC, breadcrumb, menu). While the WebHelp Responsive components are designed to comply with the accessibility rules, it is important to adhere to some rules when you write DITA topics so that the content is also accessible.

Related Information:
DITA-OT Day 2017 Presentation: Accessibility in DITA-OT

Writing Guidelines for Accessible Documentation

To create accessible content, good authoring practices involve following guidelines, such as marking table headers, using semantic elements where available, and using alternative text for images.

Accessible Images

Images must have text alternatives that describe the information or function represented by them.

Short Text Equivalents for Images

When using the `<image>` element, specify a short alternative text with the `<alt>` element.

```xml
<image href="puffin.jpg">
  <alt>Puffin figure</alt>
</image>
```

Long Descriptions of Images

For complex images, when a short text equivalent does not suffice to adequately convey the function or role of an image, provide additional information in a file designated by the `<longdesc>` element.

```xml
<image href="puffin.jpg">
  <alt>Puffin figure</alt>
  <longdesc href="http://www.example.org/birds/puffin.html"
    scope="external"
    format="html"/>
</image>
```

Related Information:
Darwin Information Typing Architecture (DITA) Specification `<image>` element
Web Accessibility Tutorials: Alt Decision Tree

Accessible Image Maps

For image maps, text alternatives are needed on both the `<image>` element itself (to describe the informative context) and on each of the `<area>` elements (to convey the link destination or the action that will be initiated...
if the link is followed). The `<xref>` content within the `<area>` element contains the intended alternative text or hover text for that image map area.

```xml
<imagemap id="gear_pump_map">
  <image href="../images/Gear_pump_exploded.png" id="gear_pump_exploded">
    <alt>Gear Pump</alt>
  </image>
  <area shape="circle" coords="172, 265, 14">
    <xref href="parts/bushings.dita#bushings_topic/bushings" format="dita">Bushings</xref>
  </area>
  <area shape="circle" coords="324, 210, 14">
    <xref href="parts/ports.dita#ports_topic/suction_port" format="dita">Suction Port</xref>
  </area>
</imagemap>

Related Information:
Darwin Information Typing Architecture (DITA) Specification `<imagemap>` element

### Accessible Tables

Accessible HTML tables need markup that indicates header cells and data cells and defines their relationship. Header cells must be marked with `<th>`, and data cells with `<td>`, to make tables accessible. For more complex tables, explicit associations may be needed using `@scope`, `@id`, and `@headers` attributes.

When you implement the table, it is best to use the `<table>` element (CALS table or OASIS Table Exchange Model). The `<table>` element includes all that you need to make a fully accessible table.

Related Information:
Darwin Information Typing Architecture (DITA) Specification `<table>` element

### Table with Header Cells in the Top Row Only

For this type of table, you have to embed the table rows in the `<thead>` element.

**Table 39. Example: Oxygen Events**

<table>
<thead>
<tr>
<th>Event</th>
<th>Date</th>
<th>Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>Evolution of TC 2018</td>
<td>May 31 - June 1, 2018</td>
<td>Sofia, Bulgaria</td>
</tr>
</tbody>
</table>
Table 39. Example: Oxygen Events (continued)

<table>
<thead>
<tr>
<th>Event</th>
<th>Date</th>
<th>Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>Markup UK</td>
<td>June 9 - 10, 2018</td>
<td>London, United Kingdom</td>
</tr>
<tr>
<td>Balisage 2018 - The Markup Conference</td>
<td>July 31 - August 3, 2018</td>
<td>Rockville, Maryland, USA</td>
</tr>
</tbody>
</table>

```xml
<table colsep="1" rowsep="1" frame="all">
  <title>
    <b>Oxygen Events</b>
  </title>
  <tgroup cols="3">
    <colspec colname="COLSPEC0" colwidth="1*"/>
    <colspec colname="COLSPEC1" colwidth="1.1*"/>
    <colspec colname="newCol3" colwidth="1*"/>
    <thead>
      <row>
        <entry colname="COLSPEC0" valign="top">Event</entry>
        <entry colname="COLSPEC1" valign="top">Date</entry>
        <entry>Location</entry>
      </row>
    </thead>
    <tbody>
      <row>
        <entry>Evolution of TC 2018</entry>
        <entry>May 31 - June 1, 2018</entry>
        <entry>Sofia, Bulgaria</entry>
      </row>
      <row>
        <entry>Markup UK</entry>
        <entry>June 9 – 10, 2018</entry>
        <entry>London, United Kingdom</entry>
      </row>
      <row>
        <entry>Balisage 2018 - The Markup Conference</entry>
        <entry>July 31 – August 3, 2018</entry>
        <entry>Rockville, Maryland, USA</entry>
      </row>
    </tbody>
  </tgroup>
</table>
```
Table with Header Cells in the First Column Only

For this type of table, you have to set the `rowheader="firstcol"` attribute on the `<table>` element to identify the header column.

**Table 40. Example: Oxygen Events**

<table>
<thead>
<tr>
<th>Event</th>
<th>Evolution of TC 2018</th>
<th>Markup UK</th>
<th>Balisage 2018 - The Markup Conference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Date</td>
<td>May 31 - June 1, 2018</td>
<td>June 9 - 10, 2018</td>
<td>July 31 - August 3, 2018</td>
</tr>
<tr>
<td>Location</td>
<td>Sofia, Bulgaria</td>
<td>London, United Kingdom</td>
<td>Rockville, Maryland, USA</td>
</tr>
</tbody>
</table>

```
<table rowheader="firstcol" colsep="1" rowsep="1" frame="all">
  <title>
    <b>Oxygen Events</b>
  </title>
  <tgroup cols="4">
    <colspec colname="COLSPEC0" colwidth="1*"/>
    <colspec colname="COLSPEC1" colwidth="1.1*"/>
    <colspec colname="newCol3" colwidth="1*"/>
    <colspec colname="newCol4" colwidth="1*"/>
    <tbody>
      <row>
        <entry>Event</entry>
        <entry>Evolution of TC 2018</entry>
        <entry>Markup UK</entry>
        <entry>Balisage 2018 - The Markup Conference</entry>
      </row>
      <row>
        <entry>Date</entry>
        <entry>May 31 - June 1, 2018</entry>
        <entry>June 9 - 10, 2018</entry>
        <entry>July 31 - August 3, 2018</entry>
      </row>
      <row>
        <entry>Location</entry>
        <entry>Sofia, Bulgaria</entry>
        <entry>London, United Kingdom</entry>
        <entry>Rockville, Maryland, USA</entry>
      </row>
    </tbody>
  </tgroup>
</table>
```
Table with Header Cells in the Top Row and First Column

For this type of table, you can use `<thead>` to identify header rows and `<rowheader>` to identify a header column.

Table 41. Example: Bus Timetable

<table>
<thead>
<tr>
<th>Time</th>
<th>Monday</th>
<th>Tuesday</th>
<th>Wednesday</th>
<th>Thursday</th>
<th>Friday</th>
</tr>
</thead>
<tbody>
<tr>
<td>09:00 - 11:00</td>
<td>Closed</td>
<td>Open</td>
<td>Open</td>
<td>Closed</td>
<td>Closed</td>
</tr>
<tr>
<td>11:00 - 13:00</td>
<td>Open</td>
<td>Open</td>
<td>Closed</td>
<td>Closed</td>
<td>Closed</td>
</tr>
<tr>
<td>13:00 - 15:00</td>
<td>Open</td>
<td>Open</td>
<td>Open</td>
<td>Closed</td>
<td>Closed</td>
</tr>
<tr>
<td>15:00 - 17:00</td>
<td>Closed</td>
<td>Closed</td>
<td>Closed</td>
<td>Open</td>
<td>Open</td>
</tr>
</tbody>
</table>

```xml
<table id="table_dqk_n24_vdb" rowheader="firstcol" colsep="1" rowsep="1" frame="all">
  <title>Example: Bus Timetable</title>
  <tgroup cols="6">
    <colspec colnum="1" colname="col1"/>
    <colspec colnum="2" colname="col2"/>
    <colspec colnum="3" colname="col3"/>
    <colspec colnum="4" colname="col4"/>
    <colspec colnum="5" colname="col5"/>
    <colspec colnum="6" colname="col6"/>
  </tgroup>
  <thead>
    <row>
      <entry/></entry>
      <entry>Monday</entry>
      <entry>Tuesday</entry>
      <entry>Wednesday</entry>
      <entry>Thursday</entry>
      <entry>Friday</entry>
    </row>
  </thead>
  <tbody>
    <row>
      <entry>09:00 - 11:00</entry>
      <entry>Closed</entry>
      <entry>Open</entry>
      <entry>Open</entry>
      <entry>Closed</entry>
      <entry>Closed</entry>
    </row>
  </tbody>
</table>
```
WebHelp Responsive VPAT Accessibility Conformance Report

International Edition

VPAT®, Version 2.3 – April 2019

Product Name/Version

Oxygen XML WebHelp Responsive

Product Description

Oxygen XML WebHelp Responsive enables you to publish DITA content on the web and present it in a user-friendly interface that is easy to navigate. You can design your WebHelp Responsive output to be available on desktop systems or various mobile devices. With Oxygen XML WebHelp Responsive, your published content is accessible, interactive, and convenient.
Date

May 2019

Contact Information

support@oxygenxml.com

Notes

Oxygen XML WebHelp Responsive has been designed and enhanced to adhere to the U.S. Government Section 508 accessibility standards and the Web Content Accessibility Guidelines (WCAG). For details, see WebHelp Responsive Accessibility (on page 1577).

Evaluation Methods Used:

The following applications were used for testing Oxygen XML WebHelp Responsive:

- Desktop browsers: Chrome, Firefox, Safari, Edge.
- Assistive technologies: NVDA, VoiceOver, JAWS, Microsoft Narrator.

Applicable Standards/Guidelines

This report covers the degree of conformance for the following accessibility standards/guidelines:

<table>
<thead>
<tr>
<th>Standard/Guideline</th>
<th>Included In Report</th>
</tr>
</thead>
<tbody>
<tr>
<td>Web Content Accessibility Guidelines 2.0</td>
<td>Level A - Yes</td>
</tr>
<tr>
<td></td>
<td>Level AA - Yes</td>
</tr>
<tr>
<td></td>
<td>Level AAA - No</td>
</tr>
<tr>
<td>Web Content Accessibility Guidelines 2.1</td>
<td>Level A - Yes</td>
</tr>
<tr>
<td></td>
<td>Level AA - Yes</td>
</tr>
<tr>
<td></td>
<td>Level AAA - No</td>
</tr>
<tr>
<td>Revised Section 508 standards published January 18, 2017 and corrected January 22, 2018</td>
<td>Yes</td>
</tr>
<tr>
<td>EN 301 549 Accessibility requirements suitable for public procurement of ICT products and services in Europe - V2.1.2 (2018-08)</td>
<td>No</td>
</tr>
</tbody>
</table>

Terms

The terms used in the Conformance Level information are defined as follows:

- **Supports**: The functionality of the product has at least one method that meets the criterion without known defects or meets with equivalent facilitation.
- **Partially Supports**: Some functionality of the product does not meet the criterion.
- **Does Not Support**: The majority of product functionality does not meet the criterion.
- **Not Applicable**: The criterion is not relevant to the product.
- **Not Evaluated**: The product has not been evaluated against the criterion. This can be used only in WCAG 2.0 Level AAA.

**WCAG 2.x Report**

Tables 1 and 2 also document conformance with:

Revised Section 508: Chapter 5 – 501.1 Scope, 504.2 Content Creation or Editing, and Chapter 6 – 602.3 Electronic Support Documentation.

**Note:**

When reporting on conformance with the WCAG 2.x Success Criteria, they are scoped for full pages, complete processes, and accessibility-supported ways of using technology as documented in the [WCAG 2.0 Conformance Requirements](#).

### Table 1: Success Criteria, Level A

<table>
<thead>
<tr>
<th>Criteria</th>
<th>Conformance Level</th>
<th>Remarks and Explanations</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>1.1.1 Non-text Content</strong> (Level A)</td>
<td>Partially Supports</td>
<td>Text alternatives are provided for many instances of non-text content, with exceptions that include permanent links for subtopics and sections.</td>
</tr>
<tr>
<td>Also applies to: Revised Section 508</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• 501 (Web)(Software)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• 504.2 (Authoring Tool)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• 602.3 (Support Docs)</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>1.2.1 Audio-only and Video-only (Prerecorded)</strong> (Level A)</td>
<td>Supports</td>
<td>The authors of the input DITA document are responsible for providing a transcript of the media content in the document.</td>
</tr>
<tr>
<td>Also applies to: Revised Section 508</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• 501 (Web)(Software)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• 504.2 (Authoring Tool)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• 602.3 (Support Docs)</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>1.2.2 Captions (Prerecorded)</strong> (Level A)</td>
<td>Supports</td>
<td>The product does not provide prerecorded media that requires captions.</td>
</tr>
<tr>
<td>Also applies to: Revised Section 508</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Criteria</td>
<td>Conformance Level</td>
<td>Remarks and Explanations</td>
</tr>
<tr>
<td>----------</td>
<td>-------------------</td>
<td>--------------------------</td>
</tr>
<tr>
<td>1.2.3 Audio Description or Media Alternative (Prerecorded) (Level A)</td>
<td>Supports</td>
<td>The authors of the input DITA document are responsible for providing an alternative for time-based media or audio description of the prerecorded video content in the document. See: G58: Placing a link to the alternative for time-based media immediately next to the non-text content</td>
</tr>
</tbody>
</table>
| 1.3.1 Info and Relationships (Level A) | Partially Supports | Information, structure, and relationships conveyed through presentation can be programmatically determined or are available in text, with exceptions that include:  
- Some landmarks are not marked with the corresponding role or do not have an associated label.  
- Some link groups are not structured using lists or are not marked as navigation regions.  
The authors of the input DITA document are responsible for:  
- Using semantic elements to mark up structure.  
- Using semantic markup to mark emphasized or special text.  
- Using caption elements to associate data table captions with data tables. |
<p>| 1.3.2 Meaningful Sequence (Level A) | Supports | The product presents content in a meaningful sequence. |</p>
<table>
<thead>
<tr>
<th>Criteria</th>
<th>Conformance Level</th>
<th>Remarks and Explanations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Revised Section 508</td>
<td></td>
<td>Authors should use Unicode right-to-left mark (RLM) or left-to-right mark (LRM) to mix text direction inline.</td>
</tr>
<tr>
<td><strong>1.3.3 Sensory Characteristics</strong> (Level A)</td>
<td>Supports</td>
<td>Authors should ensure that items are referenced in the content in ways that do not depend on sensory perception.</td>
</tr>
<tr>
<td>Also applies to: Revised Section 508</td>
<td></td>
<td></td>
</tr>
<tr>
<td>501 (Web)(Software)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>504.2 (Authoring Tool)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>602.3 (Support Docs)</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>1.4.1 Use of Color</strong> (Level A)</td>
<td>Supports</td>
<td>(Cobalt template) Color is not used as the only visual means of conveying information, indicating an action, prompting a response, or distinguishing a visual element.</td>
</tr>
<tr>
<td>Also applies to: Revised Section 508</td>
<td></td>
<td></td>
</tr>
<tr>
<td>501 (Web)(Software)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>504.2 (Authoring Tool)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>602.3 (Support Docs)</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>1.4.2 Audio Control</strong> (Level A)</td>
<td>Supports</td>
<td>There is no sound that plays automatically.</td>
</tr>
<tr>
<td>Also applies to: Revised Section 508</td>
<td></td>
<td></td>
</tr>
<tr>
<td>501 (Web)(Software)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>504.2 (Authoring Tool)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>602.3 (Support Docs)</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>2.1.1 Keyboard</strong> (Level A)</td>
<td>Partially Supports</td>
<td>Most of the content is operable through a keyboard interface, with exceptions that include:</td>
</tr>
<tr>
<td>Also applies to: Revised Section 508</td>
<td></td>
<td></td>
</tr>
<tr>
<td>501 (Web)(Software)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>504.2 (Authoring Tool)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>602.3 (Support Docs)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• The submenus (the user cannot tab to the submenus).</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• The top-level links in the main page accordion cannot be accessed.</td>
</tr>
<tr>
<td>Criteria</td>
<td>Conformance Level</td>
<td>Remarks and Explanations</td>
</tr>
<tr>
<td>----------</td>
<td>-------------------</td>
<td>--------------------------</td>
</tr>
<tr>
<td><strong>2.1.2 No Keyboard Trap</strong> (Level A)</td>
<td>Supports</td>
<td>The product does not contain content that traps the keyboard focus.</td>
</tr>
<tr>
<td>Also applies to: Revised Section 508</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• 501 (Web)(Software)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• 504.2 (Authoring Tool)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• 602.3 (Support Docs)</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>2.1.4 Character Key Shortcuts</strong> (Level A 2.1 only)</td>
<td>Supports</td>
<td>The product does not include character key shortcuts.</td>
</tr>
<tr>
<td>Also applies to: Revised Section 508 – Does not apply</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>2.2.1 Timing Adjustable</strong> (Level A)</td>
<td>Supports</td>
<td>The product does not include time limits.</td>
</tr>
<tr>
<td>Also applies to: Revised Section 508</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• 501 (Web)(Software)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• 504.2 (Authoring Tool)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• 602.3 (Support Docs)</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>2.2.2 Pause, Stop, Hide</strong> (Level A)</td>
<td>Supports</td>
<td>The product does not include elements that move, blink, scroll, or auto-update.</td>
</tr>
<tr>
<td>Also applies to: Revised Section 508</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• 501 (Web)(Software)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• 504.2 (Authoring Tool)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• 602.3 (Support Docs)</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>2.3.1 Three Flashes or Below Threshold</strong> (Level A)</td>
<td>Supports</td>
<td>The product does not contain flashing content.</td>
</tr>
<tr>
<td>Also applies to: Revised Section 508</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• 501 (Web)(Software)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• 504.2 (Authoring Tool)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• 602.3 (Support Docs)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Criteria</td>
<td>Conformance Level</td>
<td>Remarks and Explanations</td>
</tr>
<tr>
<td>----------</td>
<td>------------------</td>
<td>--------------------------</td>
</tr>
<tr>
<td><strong>2.4.1 Bypass Blocks</strong> (Level A)</td>
<td>Supports</td>
<td>Each page contains a link at the top that goes directly to the main content area. Each page contains ARIA landmarks that identify the available regions.</td>
</tr>
<tr>
<td>Also applies to: Revised Section 508 • 501 (Web)(Software) – Does not apply to non-web software • 504.2 (Authoring Tool) • 602.3 (Support Docs) – Does not apply to non-web docs</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>2.4.2 Page Titled</strong> (Level A)</td>
<td>Supports</td>
<td>Each page contains a non-empty <code>&lt;title&gt;</code> element in the <code>&lt;head&gt;</code> section.</td>
</tr>
<tr>
<td>Also applies to: Revised Section 508 • 501 (Web)(Software) • 504.2 (Authoring Tool) • 602.3 (Support Docs)</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>2.4.3 Focus Order</strong> (Level A)</td>
<td>Supports</td>
<td>Focusable components receive focus in an order that preserves meaning and operability.</td>
</tr>
<tr>
<td>Also applies to: Revised Section 508 • 501 (Web)(Software) • 504.2 (Authoring Tool) • 602.3 (Support Docs)</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>2.4.4 Link Purpose (In Context)</strong> (Level A)</td>
<td>Supports</td>
<td>The purpose of each link can be determined from the link text alone or from the link text together with its programatically-determined link context. The authors can create hypertext links using text that describes the purpose of the hypertext. There is no control that allows the user to choose between short or long link text (G189 / SCR30).</td>
</tr>
<tr>
<td>Also applies to: Revised Section 508 • 501 (Web)(Software) • 504.2 (Authoring Tool) • 602.3 (Support Docs)</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>2.5.1 Pointer Gestures</strong> (Level A 2.1 only)</td>
<td>Supports</td>
<td>The WebHelp Responsive output does not rely on path-based or multipoint gestures.</td>
</tr>
</tbody>
</table>


<table>
<thead>
<tr>
<th>Criteria</th>
<th>Conformance Level</th>
<th>Remarks and Explanations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Also applies to:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Revised Section 508 – Does not apply</td>
<td></td>
<td>gestures and does not provide controls that require complex gestures.</td>
</tr>
<tr>
<td><strong>2.5.2 Pointer Cancellation</strong> (Level A 2.1 only)</td>
<td>Supports</td>
<td>The product has operations that are activated on the pointer up event.</td>
</tr>
<tr>
<td>Also applies to:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Revised Section 508 – Does not apply</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>2.5.3 Label in Name</strong> (Level A 2.1 only)</td>
<td>Supports</td>
<td>The names of the user interface components contain the text that is presented visually.</td>
</tr>
<tr>
<td>Also applies to:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Revised Section 508 – Does not apply</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>2.5.4 Motion Actuation</strong> (Level A 2.1 only)</td>
<td>Supports</td>
<td>The product does not contain functionality that can be operated by device or user motion.</td>
</tr>
<tr>
<td>Also applies to:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Revised Section 508 – Does not apply</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>3.1.1 Language of Page</strong> (Level A)</td>
<td>Supports</td>
<td>The web pages indicate the language of the content when the content language has been specified by authors.</td>
</tr>
<tr>
<td>Also applies to:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Revised Section 508</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• 501 (Web)(Software)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• 504.2 (Authoring Tool)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• 602.3 (Support Docs)</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>3.2.1 On Focus</strong> (Level A)</td>
<td>Supports</td>
<td>No changes of context occur when any component receives focus.</td>
</tr>
<tr>
<td>Also applies to:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Revised Section 508</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• 501 (Web)(Software)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• 504.2 (Authoring Tool)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• 602.3 (Support Docs)</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>3.2.2 On Input</strong> (Level A)</td>
<td>Supports</td>
<td>Changing the setting of any user interface component does not automatically cause a change of context.</td>
</tr>
<tr>
<td>Also applies to:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Revised Section 508</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Criteria</td>
<td>Conformance Level</td>
<td>Remarks and Explanations</td>
</tr>
<tr>
<td>----------</td>
<td>-------------------</td>
<td>--------------------------</td>
</tr>
</tbody>
</table>
| • 501 (Web)(Software)  
  • 504.2 (Authoring Tool)  
  • 602.3 (Support Docs) | | |
| **3.3.1 Error Identification** (Level A) | Partially Supports | If a search operation is performed leaving the search input empty, an error message is automatically displayed to the user, but no `aria-invalid` information is provided. |
| Also applies to: Revised Section 508  
  • 501 (Web)(Software)  
  • 504.2 (Authoring Tool)  
  • 602.3 (Support Docs) | | |
| **3.3.2 Labels or Instructions** (Level A) | Partially Supports | The search input does not have a visible label specified using a label element. |
| Also applies to: Revised Section 508  
  • 501 (Web)(Software)  
  • 504.2 (Authoring Tool)  
  • 602.3 (Support Docs) | | |
| **4.1.1 Parsing** (Level A) | Partially Supports | Several HTML validation errors are reported by the W3C validator. |
| Also applies to: Revised Section 508  
  • 501 (Web)(Software)  
  • 504.2 (Authoring Tool)  
  • 602.3 (Support Docs) | | |
| **4.1.2 Name, Role, Value** (Level A) | Partially Supports | The *Home* link from the breadcrumb does not have an associated `aria-label`. |
| Also applies to: Revised Section 508  
  • 501 (Web)(Software)  
  • 504.2 (Authoring Tool)  
  • 602.3 (Support Docs) | | |
### Table 2: Success Criteria, Level AA

<table>
<thead>
<tr>
<th>Criteria</th>
<th>Conformance Level</th>
<th>Remarks and Explanations</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>1.2.4 Captions (Live)</strong> (Level AA)</td>
<td>Supports</td>
<td>No live audio content is used.</td>
</tr>
<tr>
<td>Also applies to: Revised Section 508</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• 501 (Web)(Software)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• 504.2 (Authoring Tool)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• 602.3 (Support Docs)</td>
<td></td>
</tr>
<tr>
<td><strong>1.2.5 Audio Description (Prerecorded)</strong> (Level AA)</td>
<td>Supports</td>
<td>The authors of the input DITA document can ensure that the output document meets this criterion.</td>
</tr>
<tr>
<td>Also applies to: Revised Section 508</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• 501 (Web)(Software)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• 504.2 (Authoring Tool)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• 602.3 (Support Docs)</td>
<td></td>
</tr>
<tr>
<td><strong>1.3.4 Orientation</strong> (Level AA 2.1 only)</td>
<td>Supports</td>
<td>Content does not restrict its view and operation to a single display orientation.</td>
</tr>
<tr>
<td>Also applies to: Revised Section 508 – Does not apply</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>1.3.5 Identify Input Purpose</strong> (Level AA 2.1 only)</td>
<td>Supports</td>
<td>The content does not contain input fields that collect information about the user.</td>
</tr>
<tr>
<td>Also applies to: Revised Section 508 – Does not apply</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>1.4.3 Contrast (Minimum)</strong> (Level AA)</td>
<td>Partially Supports</td>
<td>The missing words element from the search results page does not have the contrast ratio 4.5:1.</td>
</tr>
<tr>
<td>Also applies to: Revised Section 508</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• 501 (Web)(Software)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• 504.2 (Authoring Tool)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• 602.3 (Support Docs)</td>
<td></td>
</tr>
<tr>
<td><strong>1.4.4 Resize text</strong> (Level AA)</td>
<td>Partially Supports</td>
<td>Text can be resized up to 200 percent without loss of content or functionali-</td>
</tr>
<tr>
<td>Criteria</td>
<td>Conformance Level</td>
<td>Remarks and Explanations</td>
</tr>
<tr>
<td>----------</td>
<td>------------------</td>
<td>--------------------------</td>
</tr>
<tr>
<td>Revised Section 508</td>
<td></td>
<td>ty and without using assistive technology. Some text content has dimensions specified in pixels rather than em units.</td>
</tr>
</tbody>
</table>
| • 501 (Web)(Software)  
• 504.2 (Authoring Tool)  
• 602.3 (Support Docs) | | |

**1.4.5 Images of Text** (Level AA)

Also applies to:

Revised Section 508

• 501 (Web)(Software)  
• 504.2 (Authoring Tool)  
• 602.3 (Support Docs)

<table>
<thead>
<tr>
<th>Supports</th>
<th>The output does not contain images of text. The authors of the input DITA content can ensure that this criterion is met.</th>
</tr>
</thead>
</table>

**1.4.10 Reflow** (Level AA 2.1 only)

Also applies to:

Revised Section 508 – Does not apply

<table>
<thead>
<tr>
<th>Partially Supports</th>
<th>The majority of the content can be presented without loss of information or functionality, and without requiring scrolling in two dimensions. Long URLs determine the page to display the horizontal scroll bar.</th>
</tr>
</thead>
</table>

**1.4.11 Non-text Contrast** (Level AA 2.1 only)

Also applies to:

Revised Section 508 – Does not apply

<table>
<thead>
<tr>
<th>Supports</th>
<th>(Cobalt template) There is no contrast issue regarding user interface components or graphical objects.</th>
</tr>
</thead>
</table>

**1.4.12 Text Spacing** (Level AA 2.1 only)

Also applies to:

Revised Section 508 – Does not apply

<table>
<thead>
<tr>
<th>Supports</th>
<th>There is no loss of content or functionality that occurs by setting line height (line spacing), spacing following paragraphs, letter spacing, and word spacing.</th>
</tr>
</thead>
</table>

**1.4.13 Content on Hover or Focus** (Level AA 2.1 only)

Also applies to:

Revised Section 508 – Does not apply

<table>
<thead>
<tr>
<th>Partially Supports</th>
<th>Tooltips and submenus are not dismissible. Also, the tooltips are not hoverable.</th>
</tr>
</thead>
</table>

**2.4.5 Multiple Ways** (Level AA)

Also applies to:

Revised Section 508 – Does not apply

<p>| Supports | There is a search form provided that will go to a page that contains the search term and links to the corre- |</p>
<table>
<thead>
<tr>
<th>Criteria</th>
<th>Conformance Level</th>
<th>Remarks and Explanations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Revised Section 508</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• 501 (Web)(Software) – Does not apply to non-web software</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• 504.2 (Authoring Tool)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• 602.3 (Support Docs) – Does not apply to non-web docs</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>2.4.6 Headings and Labels</strong></td>
<td><strong>Supports</strong></td>
<td>Headings and labels describe the topic or purpose. DITA authors can ensure that this criterion is met.</td>
</tr>
<tr>
<td>Also applies to:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Revised Section 508</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• 501 (Web)(Software)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• 504.2 (Authoring Tool)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• 602.3 (Support Docs)</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>2.4.7 Focus Visible</strong></td>
<td><strong>Partially Supports</strong></td>
<td>Placing focus on a focusable element using the mouse doesn't render a visible focus indicator. Also, the search button does not have a visible focus indicator.</td>
</tr>
<tr>
<td>Also applies to:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Revised Section 508</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• 501 (Web)(Software)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• 504.2 (Authoring Tool)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• 602.3 (Support Docs)</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>3.1.2 Language of Parts</strong></td>
<td><strong>Supports</strong></td>
<td>DITA authors can ensure that this criterion is met.</td>
</tr>
<tr>
<td>Also applies to:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Revised Section 508</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• 501 (Web)(Software)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• 504.2 (Authoring Tool)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• 602.3 (Support Docs)</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>3.2.3 Consistent Navigation</strong></td>
<td><strong>Supports</strong></td>
<td>Repeated components appear in the same relative in each page.</td>
</tr>
<tr>
<td>Also applies to:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Revised Section 508</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Criteria</td>
<td>Conformance Level</td>
<td>Remarks and Explanations</td>
</tr>
<tr>
<td>----------</td>
<td>------------------</td>
<td>-------------------------</td>
</tr>
<tr>
<td>3.2.4 Consistent Identification (Level AA)</td>
<td>Partially Supports</td>
<td>The output uses labels, names, and text alternatives consistently for items that have the same functionality. Text alternatives are provided for many instances of non-text content, with exceptions that include: - Permalinks for subtopics and sections. - Enlarge images action. The <em>Home</em> link from the breadcrumb does not have an associated <em>aria-label</em>.</td>
</tr>
<tr>
<td>3.3.3 Error Suggestion (Level AA)</td>
<td>Does Not Support</td>
<td>The Search input does not have the <em>aria-required</em> information set and does not contain a text description specifying that it is a required field.</td>
</tr>
<tr>
<td>3.3.4 Error Prevention (Legal, Financial, Data) (Level AA)</td>
<td>Supports</td>
<td>The Web pages do not cause legal commitments or financial transactions for the user to occur, that modify or delete user-controllable data in data storage systems, or that submit user test responses.</td>
</tr>
<tr>
<td>4.1.3 Status Messages (Level AA 2.1 only)</td>
<td>Supports</td>
<td>The pages do not contain status messages as defined by this criterion.</td>
</tr>
<tr>
<td>Criteria</td>
<td>Conformance Level</td>
<td>Remarks and Explanations</td>
</tr>
<tr>
<td>------------------------------------------------------------------------</td>
<td>-------------------</td>
<td>--------------------------</td>
</tr>
<tr>
<td>Revised Section 508 – Does not apply</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Table 3: Success Criteria, Level AAA**

<table>
<thead>
<tr>
<th>Criteria</th>
<th>Conformance Level</th>
<th>Remarks and Explanations</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>1.2.6 Sign Language (Prerecorded)</strong> (Level AAA)</td>
<td>Not Evaluated</td>
<td></td>
</tr>
<tr>
<td>Revised Section 508 – Does not apply</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>1.2.7 Extended Audio Description (Prerecorded)</strong> (Level AAA)</td>
<td>Not Evaluated</td>
<td></td>
</tr>
<tr>
<td>Revised Section 508 – Does not apply</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>1.2.8 Media Alternative (Prerecorded)</strong> (Level AAA)</td>
<td>Not Evaluated</td>
<td></td>
</tr>
<tr>
<td>Revised Section 508 – Does not apply</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>1.2.9 Audio-only (Live)</strong> (Level AAA)</td>
<td>Not Evaluated</td>
<td></td>
</tr>
<tr>
<td>Revised Section 508 – Does not apply</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>1.3.6 Identify Purpose</strong> (Level AAA 2.1 only)</td>
<td>Not Evaluated</td>
<td></td>
</tr>
<tr>
<td>Revised Section 508 – Does not apply</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>1.4.6 Contrast Enhanced</strong> (Level AAA)</td>
<td>Not Evaluated</td>
<td></td>
</tr>
<tr>
<td>Revised Section 508 – Does not apply</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>1.4.7 Low or No Background Audio</strong> (Level AAA)</td>
<td>Not Evaluated</td>
<td></td>
</tr>
<tr>
<td>Revised Section 508 – Does not apply</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>1.4.8 Visual Presentation</strong> (Level AAA)</td>
<td>Not Evaluated</td>
<td></td>
</tr>
<tr>
<td>Revised Section 508 – Does not apply</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>1.4.9 Images of Text (No Exception) Control</strong> (Level AAA)</td>
<td>Not Evaluated</td>
<td></td>
</tr>
<tr>
<td>Revised Section 508 – Does not apply</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>2.1.3 Keyboard (No Exception)</strong> (Level AAA)</td>
<td>Not Evaluated</td>
<td></td>
</tr>
<tr>
<td>Revised Section 508 – Does not apply</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Criteria</td>
<td>Conformance Level</td>
<td>Remarks and Explanations</td>
</tr>
<tr>
<td>----------------------------------------------</td>
<td>-------------------</td>
<td>--------------------------</td>
</tr>
<tr>
<td>Revised Section 508 – Does not apply</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>2.2.3 No Timing</strong> (Level AAA)</td>
<td></td>
<td>Not Evaluated</td>
</tr>
<tr>
<td>Revised Section 508 – Does not apply</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>2.2.4 Interruptions</strong> (Level AAA)</td>
<td></td>
<td>Not Evaluated</td>
</tr>
<tr>
<td>Revised Section 508 – Does not apply</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>2.2.5 Re-authenticating</strong> (Level AAA)</td>
<td></td>
<td>Not Evaluated</td>
</tr>
<tr>
<td>Revised Section 508 – Does not apply</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>2.2.6 Timeouts</strong> (Level AAA 2.1 only)</td>
<td></td>
<td>Not Evaluated</td>
</tr>
<tr>
<td>Revised Section 508 – Does not apply</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>2.3.2 Three Flashes</strong> (Level AAA)</td>
<td></td>
<td>Not Evaluated</td>
</tr>
<tr>
<td>Revised Section 508 – Does not apply</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>2.3.3 Animation from Interactions</strong> (Level AAA 2.1 only)</td>
<td></td>
<td>Not Evaluated</td>
</tr>
<tr>
<td>Revised Section 508 – Does not apply</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>2.4.8 Location</strong> (Level AAA)</td>
<td></td>
<td>Not Evaluated</td>
</tr>
<tr>
<td>Revised Section 508 – Does not apply</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>2.4.9 Link Purpose (Link Only)</strong> (Level AAA)</td>
<td></td>
<td>Not Evaluated</td>
</tr>
<tr>
<td>Revised Section 508 – Does not apply</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>2.4.10 Section Headings</strong> (Level AAA)</td>
<td></td>
<td>Not Evaluated</td>
</tr>
<tr>
<td>Revised Section 508 – Does not apply</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>2.5.5 Target Size</strong> (Level AAA 2.1 only)</td>
<td></td>
<td>Not Evaluated</td>
</tr>
<tr>
<td>Revised Section 508 – Does not apply</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>2.5.6 Concurrent Input Mechanisms</strong> (Level AAA 2.1 only)</td>
<td></td>
<td>Not Evaluated</td>
</tr>
<tr>
<td>Revised Section 508 – Does not apply</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>3.1.3 Unusual Words</strong> (Level AAA)</td>
<td></td>
<td>Not Evaluated</td>
</tr>
<tr>
<td>Revised Section 508 – Does not apply</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>3.1.4 Abbreviations</strong> (Level AAA)</td>
<td></td>
<td>Not Evaluated</td>
</tr>
</tbody>
</table>
### Revised Section 508 Report

N/A

## Chapter 3: Functional Performance Criteria (FPC)

<table>
<thead>
<tr>
<th>Criteria</th>
<th>Conformance Level</th>
<th>Remarks and Explanations</th>
</tr>
</thead>
<tbody>
<tr>
<td>302.1 Without Vision</td>
<td>Partially Supports</td>
<td>Most of the content is accessible without vision with exceptions that include:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Some components do not have text alternatives or labels.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Some landmarks are not marked with the corresponding role or do not have an associat-</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Some link groups are not structured using lists or are not marked as navigation re-</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Some components do not have text alternatives or labels.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Some landmarks are not marked with the corresponding role or do not have an associat-</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Some link groups are not structured using lists or are not marked as navigation re-</td>
</tr>
</tbody>
</table>

<p>| 302.2 With Limited Vision            | Partially Supports | Most of the content is accessible with limited vision with exceptions that include:   |</p>
<table>
<thead>
<tr>
<th>Criteria</th>
<th>Conformance Level</th>
<th>Remarks and Explanations</th>
</tr>
</thead>
<tbody>
<tr>
<td>302.3 Without Perception of Color</td>
<td>Supports</td>
<td>(Cobalt template) Color is not used as the only visual means of conveying information, indicating an action, prompting a response, or distinguishing a visual element.</td>
</tr>
<tr>
<td>302.4 Without Hearing</td>
<td>Supports</td>
<td>The authors can create content that does not require hearing abilities for use.</td>
</tr>
<tr>
<td>302.5 With Limited Hearing</td>
<td>Supports</td>
<td>The authors can create content that does not require hearing abilities for use.</td>
</tr>
<tr>
<td>302.6 Without Speech</td>
<td>Supports</td>
<td>The output does not require speech for use.</td>
</tr>
<tr>
<td>302.7 With Limited Manipulation</td>
<td>Supports</td>
<td>The WebHelp Responsive output does not rely on path-based or multipoint gestures and does not provide controls that require complex gestures.</td>
</tr>
<tr>
<td>302.8 With Limited Reach and Strength</td>
<td>Supports</td>
<td>The WebHelp Responsive output does not rely on path-based or multipoint gestures and does not provide controls that require complex gestures.</td>
</tr>
<tr>
<td>302.9 With Limited Language, Cognitive, and Learning Abilities</td>
<td>Supports</td>
<td>The authors can create content that can be used by users with limited language, cognitive, and learning abilities.</td>
</tr>
</tbody>
</table>

**Chapter 4: Hardware**

Notes: Not Applicable - Oxygen XML WebHelp Responsive is not a hardware product.
Chapter 5: Software

Notes: Oxygen XML WebHelp Responsive is a web application, not a software product. However, the web application includes authoring functionality, hence Chapter 5: Software 504 Authoring Tools applies to this product.

501 General

<table>
<thead>
<tr>
<th>Criteria</th>
<th>Conformance Level</th>
<th>Remarks and Explanations</th>
</tr>
</thead>
<tbody>
<tr>
<td>501.1 Scope – Incorporation of WCAG 2.0 AA</td>
<td>See WCAG 2.x section (on page 1585)</td>
<td>See information in WCAG section</td>
</tr>
</tbody>
</table>

502 Interoperability with Assistive Technology

<table>
<thead>
<tr>
<th>Criteria</th>
<th>Conformance Level</th>
<th>Remarks and Explanations</th>
</tr>
</thead>
<tbody>
<tr>
<td>502.2.1 User Control of Accessibility Features</td>
<td>Not Applicable</td>
<td>The product is not platform software.</td>
</tr>
<tr>
<td>502.2.2 No Disruption of Accessibility Features</td>
<td>Supports</td>
<td>The product does not disrupt platform features that are defined in the platform documentation as accessibility features.</td>
</tr>
</tbody>
</table>

502.3 Accessibility Services

<table>
<thead>
<tr>
<th>Criteria</th>
<th>Conformance Level</th>
<th>Remarks and Explanations</th>
</tr>
</thead>
<tbody>
<tr>
<td>502.3.1 Object Information</td>
<td>Partially Supports</td>
<td>The majority of object roles, state(s), properties, boundary, name, and description are programmatically determinable. The Home link from the breadcrumb does not have an associated aria-label.</td>
</tr>
<tr>
<td>502.3.2 Modification of Object Information</td>
<td>Supports</td>
<td>States and properties that can be set by the user can be set programmatically.</td>
</tr>
<tr>
<td>502.3.3 Row, Column, and Headers</td>
<td>Supports</td>
<td>The headers associated with the rows or columns of a table can be programmatically determined.</td>
</tr>
<tr>
<td>502.3.4 Values</td>
<td>Supports</td>
<td>The current values of an object can be programmatically determined.</td>
</tr>
<tr>
<td>Criteria</td>
<td>Conformance Level</td>
<td>Remarks and Explanations</td>
</tr>
<tr>
<td>----------</td>
<td>-------------------</td>
<td>--------------------------</td>
</tr>
<tr>
<td><strong>502.3.5 Modification of Values</strong></td>
<td>Supports</td>
<td>Values that can be set by the user are capable of being set programmatically.</td>
</tr>
<tr>
<td><strong>502.3.6 Label Relationships</strong></td>
<td>Partially Supports</td>
<td>Information, structure, and relationships conveyed through presentation can be programmatically determined or are available in text. See <a href="#">WCAG 1.3.1</a> (on page 1586).</td>
</tr>
<tr>
<td><strong>502.3.7 Hierarchical Relationships</strong></td>
<td>Supports</td>
<td>The content is hierarchically structured using language-specific elements and their relationships can be programmatically determined.</td>
</tr>
<tr>
<td><strong>502.3.8 Text</strong></td>
<td>Supports</td>
<td>The content of text objects, text attributes, and the boundary of text rendered to the screen shall be programmatically determinable.</td>
</tr>
<tr>
<td><strong>502.3.9 Modification of Text</strong></td>
<td>Supports</td>
<td>The editable text (search input) can be set programmatically.</td>
</tr>
<tr>
<td><strong>502.3.10 List of Actions</strong></td>
<td>Not Applicable</td>
<td>There are no custom actions available that can be executed on the content.</td>
</tr>
<tr>
<td><strong>502.3.11 Actions on Objects</strong></td>
<td>Not Applicable</td>
<td>There are no custom actions available that can be executed on the content.</td>
</tr>
<tr>
<td><strong>502.3.12 Focus Cursor</strong></td>
<td>Not Applicable</td>
<td>The product is a web application and is isolated from the underlying platform software (web browser).</td>
</tr>
<tr>
<td><strong>502.3.13 Modification of Focus Cursor</strong></td>
<td>Not Applicable</td>
<td>The product is a web application and is isolated from the underlying platform software (web browser).</td>
</tr>
<tr>
<td><strong>502.3.14 Event Notification</strong></td>
<td>Not Applicable</td>
<td>There are no automatic focus changes, caret movement, selection changes, or added components in the content.</td>
</tr>
<tr>
<td><strong>502.4 Platform Accessibility Features</strong></td>
<td>Not Applicable</td>
<td>This product is not platform software.</td>
</tr>
</tbody>
</table>

**503 Applications**
### 503.4 User Controls for Captions and Audio Description

<table>
<thead>
<tr>
<th>Criteria</th>
<th>Conformance Level</th>
<th>Remarks and Explanations</th>
</tr>
</thead>
<tbody>
<tr>
<td>503.4.1 Caption Controls</td>
<td>Not Applicable</td>
<td>The product does not provide controls for volume adjustment.</td>
</tr>
<tr>
<td>503.4.2 Audio Description Controls</td>
<td>Not Applicable</td>
<td>The product does not provide controls for program selection.</td>
</tr>
</tbody>
</table>

### 504 Authoring Tools

<table>
<thead>
<tr>
<th>Criteria</th>
<th>Conformance Level</th>
<th>Remarks and Explanations</th>
</tr>
</thead>
<tbody>
<tr>
<td>504.2 Content Creation or Editing (if not authoring tool, enter &quot;not applicable&quot;)</td>
<td>Not Applicable</td>
<td>The product is not an authoring tool. See the WCAG 2.x section (on page 1585) See information in WCAG section</td>
</tr>
<tr>
<td>504.2.1 Preservation of Information Provided for Accessibility in Format Conversion</td>
<td>Not Applicable</td>
<td>The product is not an authoring tool.</td>
</tr>
<tr>
<td>504.2.2 PDF Export</td>
<td>Not Applicable</td>
<td>The product is not an authoring tool.</td>
</tr>
<tr>
<td>504.3 Prompts</td>
<td>Not Applicable</td>
<td>The product is not an authoring tool.</td>
</tr>
<tr>
<td>504.4 Templates</td>
<td>Not Applicable</td>
<td>The product is not an authoring tool.</td>
</tr>
</tbody>
</table>

### Chapter 6: Support Documentation and Services

#### 601.1 Scope

#### 602 Support Documentation

<table>
<thead>
<tr>
<th>Criteria</th>
<th>Conformance Level</th>
<th>Remarks and Explanations</th>
</tr>
</thead>
<tbody>
<tr>
<td>602.2 Accessibility and Compatibility Features</td>
<td>Partially Supports</td>
<td>The product documentation is distributed in the WebHelp Responsive format. See the Chapter 3 (on page</td>
</tr>
</tbody>
</table>
### 602.3 Electronic Support Documentation

See the WCAG 2.x section (on page 1585) and Chapter 5 (on page 1600) sections.

### 602.4 Alternate Formats for Non-Electronic Support Documentation

Not Applicable

Documentation is not provided in non-electronic formats.

### 603 Support Services

<table>
<thead>
<tr>
<th>Criteria</th>
<th>Conformance Level</th>
<th>Remarks and Explanations</th>
</tr>
</thead>
<tbody>
<tr>
<td>603.2 Information on Accessibility and Compatibility Features</td>
<td>Supports</td>
<td>The support services cover the accessibility features.</td>
</tr>
<tr>
<td>603.3 Accommodation of Communication Needs</td>
<td>Supports</td>
<td>Support services are available by phone or e-mail.</td>
</tr>
</tbody>
</table>

### Legal Disclaimer

This report describes Oxygen XML WebHelp's ability to support the stated VPAT Standards/Guidelines, subject to Syncro Soft's interpretation of the same. This accessibility report is provided for informational purposes only, and the contents hereof are subject to change without notice. SYNCRO SOFT MAKES NO WARRANTIES, EXPRESS OR IMPLIED, IN THIS DOCUMENT. For more information regarding the accessibility status, please contact us at sales@oxygenxml.com.

© 2019 Syncro Soft SRL. All rights reserved.

### Publishing Templates

An Oxygen Publishing Template defines all aspects of the layout and styles for output obtained from the following transformation scenarios:

- WebHelp Responsive
- DITA Map PDF - based on HTML5 & CSS

It is a self-contained customization package stored as a ZIP archive or folder that can easily be shared with others. It provides the primary method for customizing the output.

Some possible customization methods include:

- Add additional template resources to customize the output (such as logos, Favicons, or CSS files).
- Extend the default processing by specifying one or more XSLT extension points.
- Specify one or more transformation parameters to customize the output.
Customize various aspects of the output through simple CSS styling.

For WebHelp Responsive output, change the layout of the main page or topic pages by customizing which components will be displayed and where they will be positioned in the page.

The following graphics are possible sample structures for Oxygen Publishing Template packages:

**Figure 489. Oxygen Publishing Template Package (WebHelp Responsive)**

```
publishing template
  CSS
    oxygen-skin.css
  JS
  fonts
  HTML-Fragments
    webhelp.fragment.footer.html
    webhelp.fragment.before.logo_and_title.html
  XSLT-Extensions
    topic_page_extension.xsl
  page-templates
    wt_index.html
    wt_search.html
    wt_topic.html
    wt_terms.html
  template_descriptor.opt
```

**Resources (CSS, JS, Fonts)**

**HTML fragments**

**XSLT Extensions**

**HTML Page Layout Files**

**Template Descriptor**

**Figure 490. Oxygen Publishing Template Package (PDF)**

```
publishing template
  CSS
    oxygen-skin.css
  XSLT-Extensions
    topic_page_extension.xsl
  template_descriptor.opt
```

**Resources (CSS)**

**XSLT Extensions**

**Template Descriptor**

For information about creating and customizing publishing templates, and how to adjust the WebHelp and PDF output through CSS styling and other customization methods, watch our Webinar: Creating Custom Publishing Templates for WebHelp and PDF Output. The Webinar slides and sample project are also available from that webpage.

**Related Information:**

- How to Create a Publishing Template (on page 1642)
- How to Edit a Packed Publishing Template (on page 1644)
Publishing Templates Gallery

Oxygen XML Editor comes bundled with a variety of built-in templates. You can use one of them to publish your documentation or as a starting point for a new publishing template.

Built-in Templates

There are two categories of templates, Tiles and Tree. You can see the built-in templates in the Templates tab when editing a WebHelp Responsive transformation scenario in Oxygen XML Editor/Author. Each one also includes an Online preview icon in the bottom-right corner that opens a webpage in your default browser that provides a sample of how the main page will look when that particular template is used to generate the output.

Tiles Templates

The main page in the WebHelp output presents a tile for each main topic (chapter) of the documentation.

Tree Templates

The main page in the WebHelp output presents a tree-like table of contents.
Built-in Templates Location

All built-in templates are stored in the following directory: DITA-OT-DIR/plugins/com.oxygenxml.webhelp.responsive/templates.

Custom Templates

You can use a built-in template as a starting point for creating your own custom template (on page 1791). You can store all of your custom templates in a particular directory. Then, go to Options > Preferences > DITA > Publishing and add your directory to the list, and all the templates stored in that directory will be displayed in the preview pane in the transformation scenario's Template tab along with all the built-in templates.

Sharing Publishing Template

To share a publishing template with others, following these steps:

1. Copy your template in a new folder.
2. Go to Options > Preferences > DITA > Publishing and add that new folder to the list.
3. Switch the option as the bottom of that preferences page to Project Options.
4. Share your project file (.xpr).

Publishing Template Package Contents for WebHelp Responsive Customizations

An Oxygen Publishing Template package for WebHelp output must contain a template descriptor file and at least one CSS file, and may contain other resources (such as graphics, XHTML files, XSLT files, etc.). All the template resources can be stored in either a ZIP archive or in a folder. It is recommended to use a ZIP archive because it is easier to share with others.
Template Descriptor File

Each publishing template includes a descriptor file that defines the meta-data associated with the template. It is an XML file that defines all the resources included in a template (such as CSS files, images, JS files, and transformation parameters).

The template descriptor file must have the .opt file extension and must be located in the template’s root folder.

A template descriptor might look like this:

```xml
<publishing-template>
  <name>Flowers</name>

  <webhelp>
    <tags>
      <tag>tree</tag>
      <tag>light</tag>
    </tags>
    <preview-image file="flowers-tree.png"/>

    <!-- Resources (CSS, favicon, logo and others) -->
    <resources>
      <!-- Main CSS file -->
      <css file="flowers.css"/>

      <!-- Resources to copy to the output folder -->
      <fileset>
        <include name="resources/**/*"/>
        <exclude name="resources/**/.svn"/>
        <exclude name="resources/**/.git"/>
      </fileset>
    </resources>

    <!-- Parameters -->
    <parameters>
      <parameter name="webhelp.show.main.page.tiles" value="no"/>
      <parameter name="webhelp.show.main.page.toc" value="yes"/>
      <parameter name="webhelp.top.menu.depth" value="3"/>
    </parameters>
  </webhelp>
</publishing-template>```
Tip:
It is recommended to edit the template descriptor in Oxygen XML Editor/Author because it provides content completion and validation support.

Template Name and Description

Each template descriptor file requires a `<name>` element. This information is displayed as the name of the template in the transformation scenario dialog box.

Optionally, you can include a `<description>` and it displayed when the user hovers over the template in the transformation scenario dialog box.

```xml
<publishing-template>
  <name>Lorem Ipsum</name>
  <description>Lorem ipsum dolor sit amet, consectetur adipiscing elit</description>
  ...
</publishing-template>
```

Template Author

Optionally, you can include author information in the descriptor file and it displayed when the user hovers over the template in the transformation scenario dialog box. This information might be useful if users run into an issue or have questions about a certain template.

If you include the `<author>` element, a `<name>` is required and optionally you can include `<email>`, `<organization>`, and `<organizationUrl>` information.

```xml
<publishing-template>
  ...
  <author>
    <name>John Doe</name>
    <email>jdoe@example.com</email>
    <organization>ACME</organization>
    <organizationUrl>http://www.example.com/jdoe</organizationUrl>
  </author>
  ...
</publishing-template>
```

Webhelp Element

The `<webhelp>` element contains various details that define the WebHelp Responsive output. It is a required element if you intend on using a WebHelp Responsive transformation scenario. The elements that are allowed in this `<webhelp>` section specify the template tags (on page 1609), template preview image (on page 1609), resources (on page 1610) (such as CSS, JS, fonts, logos), transformation parameters (on page 1612), HTML fragment extensions (on page 1613) (used to add fragments to placeholders), XSLT extensions (on page 1613), or HTML page layout files (on page 1622).
Template Tags

The `<tags>` section provides meta information about the template (such as layout type or color theme). Each `tag` is displayed at the top of the Templates tab window in the transformation scenario dialog box and they help the user filter and find particular templates.

```
<publishing-template>
  ...
  <webhelp>
    <tags>
      <tag>tree</tag>
      <tag>dark</tag>
    </tags>
  </webhelp>
</publishing-template>
```

Template Preview Image

The `<preview-image>` element is used to specify an image that will be displayed in the transformation scenario dialog box. It provides a visual representation of the template to help the user select the right template. The image dimensions should be 200 x 115 pixels and the supported image formats are: JPEG, PNG, or GIF.

You can also include an `<online-preview-url>` element to specify the URL of a published sample of your template. This will display an Online preview icon in the bottom-right corner the image in the transformation scenario dialog box and if the user clicks that icon, it will open the specified URL in their default browser.
Template Resources

The `<resources>` section of the descriptor file specifies a set of resources (CSS, JS, fonts, logos, graphics, etc.) that are used to customize various components in the generated output. These resources will be copied to the output folder during the transformation process. At least one CSS file must be included, while the other types of resources are optional.

⚠️ Warning:
All paths set in the `<file>` attribute must be relative.

This section is defined using the `resources` element and the types of resources that can be specified include:

- **CSS files** - One or more CSS files that will define the styles of all generated HTML pages. They are referenced using the `<css>` element.
- **Favicon** - You can specify the path to an image for the favicon associated with your website. It is referenced using the `<favicon>` element.
- **Logo** - You can specify the path to a logo image that will be displayed in the left side of the output header. It is referenced using the `<logo>` element. Optionally, you can also specify:
  - `<target-url>` - will redirect the user to the specified URL if they click the logo in the output.
  - `<alt>` - provides an alternate text for the logo image.
- **JavaScript AMD module** - The path to a JavaScript module that uses the AMD (Asynchronous Module Definition) format. This module will be loaded in the output HTML pages using the RequireJS library. It can be referenced using the `<js-amd-module>` element. For more information, see How to Insert JavaScript AMD Modules (on page 1659).
- **Additional Resources (graphics, JS, fonts, folders)** - For other resources (such as images referenced in CSS, JavaScript, fonts, entire folders, etc.) that need to be included in the output, you need to instruct the transformation to include them in the output folder. You can specify one or more sets of additional resources to be copied to the output folder by using the `<fileset>` element and you can use one or more `<include>` and `<exclude>` elements. This semantic is similar to the ANT FileSet.

```xml
<publishing-template>
  
  
  <webhelp>
    
  </webhelp>

  <resources>
    <css file="css/custom_styles.css"/>
  </resources>

</publishing-template>
```
<css file="css/custom_fonts.css"/>

<favicon file="images/favicon.png"/>

<logo
    file="images/logo.png"
    target-url="http://www.example.com"
    alt="Alternate text for the logo image"/>

<js-amd-module file="js/template-main.js"/>

<fileset>
    <include name="common/**/*"/>
    <include name="JS/**/*"/>
    <exclude name="***/*.svn"/>
    <exclude name="***/*.git"/>
</fileset>
</resources>

**Note:**
All relative paths specified in the descriptor file are relative to the template root folder.

The resources specified in the template descriptor are copied to the following output folder: 
```
[WebHelp_OUTPUT_DIR]/oxygen-webhelp/template
```
The following graphic illustrates the mapping between the template resources and the location where they will be copied to the output folder:

**Figure 491. Template Resources Mapping**

Related Information:
[How to Add a Favicon in WebHelp Systems](on page 1673)
Transformation Parameters

You can also set one or more WebHelp transformation parameters in the descriptor file.

```xml
<publishing-template>
  ...
  <webhelp>
    ...
    <parameters>
      <parameter
        name="webhelp.show.main.page.toc"
        value="yes"/>
      <parameter
        name="webhelp.top.menu.depth"
        value="3"/>
      <parameter
        name="webhelp.fragment.welcome"
        value="html-fragment/webhelp.fragment.welcome.html"
        type="filePath"/>
    </parameters>
  </webhelp>
</publishing-template>
```

The following information can be specified in the `<parameter>` element:

**Parameter name**

The name of the parameter. It may be one of the WebHelp Responsive transformation parameters *(on page 1718)* or a DITA-OT HTML-based output parameter.

**Note:**

It is not recommended to specify an input/output parameter in the descriptor file (such as the input Map, DITAVAL file, or temporary directory).

**Attention:**

JVM arguments like `-Xmx` cannot be specified as a transformation parameter.

**Parameter Value**

The value of the parameter. It should be a relative path to the template root folder for file paths parameters.

**Parameter Type**

The type of the parameter: `string` or `filepath`. The `string` value is default.
After creating a publishing template (on page 1791) and adding it to the templates gallery (on page 1644), when you select the template in the transformation scenario dialog box, the Parameters tab will automatically be updated to include the parameters defined in the descriptor file. These parameters are displayed in italics.

**XSLT Extension Points**

The publishing templates can include one or more supported XSLT extension points (on page 1731). They are helpful when you want to change the structure of the HTML pages that are primarily generated from XSLT processing. They can be specified using the `<xslt>` element in the descriptor file using the following structure:

```
<publishing-template>
  ...
  <webhelp>
    ...
    <xslt>
      <extension
        id="com.oxygenxml.webhelp.xsl.dita2webhelp"
        file="xsl/customDita2webhelp.xsl"/>
      <extension
        id="com.oxygenxml.webhelp.xsl.createMainPage"
        file="xsl/customMainPage.xsl"/>
    </xslt>
  </webhelp>
</publishing-template>
```

For a full list of the supported extension points, see: XSLT-Import and XSLT-Parameter Extension Points (on page 1731).

**Note:**

You can read the value of a WebHelp transformation parameter from your XSLT extension stylesheets by using the `getParameter(param.name)` function from the `http://www.oxygenxml.com/functions` namespace.

**HTML Fragment Placeholders**

The HTML pages contain component placeholders that can be used to insert custom HTML fragments either by specifying a well-formed XHTML fragment or referencing a path to a file that contains a well-formed XHTML fragment (for details on how the file or fragment needs to be constructed, see How to Insert Custom HTML Content (on page 1654)).

These fragments and their placeholder location are defined in the descriptor file using a `<fragment>` element inside the `<html-fragments>` section. You can specify one or more HTML fragment extension points in the descriptor file using the following structure:

```
<publishing-template>
  ...
  <webhelp>
</publishing-template>
```
Some of these placeholders are left empty in the default output configurations, but you can use them to insert custom content.

Each placeholder has an associated parameter value in the transformation. Some of the placeholder parameters are global and can be used in all type of pages (main page, topic page, search results page, index terms page), while others are applicable for certain type of pages. The following diagram illustrates the predefined placeholders that are global (can be used in any of the types of pages).

**Figure 492. Global Predefined Placeholders Diagram**

1. Header *(on page 1615)*
2. After Header *(on page 1615)*
3. Before Body *(on page 1615)*
4. Before Logo and Title *(on page 1615)*
5. After Logo and Title *(on page 1615)*
6. Before Top Menu *(on page 1615)*
Global Placeholder Parameters

The following placeholder parameters can be used in any of the type of pages (main page, topic page, search results page, index terms page). The parameter values can be either a well-formed XHTML fragment or a path to a file that contains a well-formed XHTML fragment:

- \texttt{1 = webhelp.fragment.head} - Displays the specified XHTML fragment in the header section.
- \texttt{2 = webhelp.fragment.after.header} - Displays the specified XHTML fragment after the header section.
- \texttt{3 = webhelp.fragment.before.body} - Displays the specified XHTML fragment before the body.
- \texttt{4 = webhelp.fragment.before.logo_and_title} - Displays the specified XHTML fragment before the logo and title.
- \texttt{5 = webhelp.fragment.after.logo_and_title} - Displays the specified XHTML fragment after the logo and title.
- \texttt{6 = webhelp.fragment.before.top_menu} - Displays the specified XHTML fragment before the top menu.
- \texttt{7 = webhelp.fragment.after.top_menu} - Displays the specified XHTML fragment after the top menu.
- \texttt{8 = webhelp.fragment.before.search.input} - Displays the specified XHTML fragment before the search input component.
- \texttt{9 = webhelp.fragment.after.search.input} - Displays the specified XHTML fragment after the search input component.
- \texttt{10 = webhelp.fragment.before.main.content.area} - Displays the specified XHTML fragment before the main content area.
- \texttt{11 = webhelp.fragment.after.main.content.area} - Displays the specified XHTML fragment after the main content area.
- \texttt{12 = webhelp.fragment.footer} - Displays the specified XHTML fragment in the footer section.
- \texttt{13 = webhelp.fragment.after.body} - Displays the specified XHTML fragment after the body.

Main Page Placeholder Parameters

The following placeholder parameters can be used in the main page. The parameter values can be either a well-formed XHTML fragment or a path to a file that contains a well-formed XHTML fragment:

- \texttt{webhelp.fragment.head.main.page} - Displays the specified XHTML fragment in the header section.
- \texttt{webhelp.fragment.after.header.main.page} - Displays the specified XHTML fragment after the header section.
- \texttt{webhelp.fragment.before.body.main.page} - Displays the specified XHTML fragment before the body.
• **webhelp.fragment.before.search.input.main.page** - Displays the specified XHTML fragment before the search input component.
• **webhelp.fragment.after.search.input.main.page** - Displays the specified XHTML fragment after the search input component.
• **webhelp.fragment.before.main.content.area.main.page** - Displays the specified XHTML fragment before the main content area
• **webhelp.fragment.after.main.content.area.main.page** - Displays the specified XHTML fragment after the main content area.
• **webhelp.fragment.after.body.main.page** - Displays the specified XHTML fragment after the body.
• **webhelp.fragment.before.toc_or_tiles** - Displays the specified XHTML fragment before the main table of contents or tiles component on the main page.
• **webhelp.fragment.after.toc_or_tiles** - Displays the specified XHTML fragment after the main table of contents or tiles component on the main page.

### Topic Page Placeholder Parameters

The following placeholder parameters can be used in the **topic page**. The parameter values can be either a well-formed XHTML fragment or a path to a file that contains a well-formed XHTML fragment:

• **webhelp.fragment.head.topic.page** - Displays the specified XHTML fragment in the header section.
• **webhelp.fragment.after.header.topic.page** - Displays the specified XHTML fragment after the header section.
• **webhelp.fragment.before.body.topic.page** - Displays the specified XHTML fragment before the body.
• **webhelp.fragment.before.search.input.topic.page** - Displays the specified XHTML fragment before the search input component.
• **webhelp.fragment.after.search.input.topic.page** - Displays the specified XHTML fragment after the search input component.
• **webhelp.fragment.before.main.content.area.topic.page** - Displays the specified XHTML fragment before the main content area
• **webhelp.fragment.after.main.content.area.topic.page** - Displays the specified XHTML fragment after the main content area.
• **webhelp.fragment.after.body.topic.page** - Displays the specified XHTML fragment after the body.
• **webhelp.fragment.before.topic.toolbar** - Displays the specified XHTML fragment before the toolbar buttons above the topic content in the topic page.
• **webhelp.fragment.after.topic.toolbar** - Displays the specified XHTML fragment after the toolbar buttons above the topic content in the topic page.
• **webhelp.fragment.before.topic.breadcrumb** - Displays the specified XHTML fragment before the breadcrumb component in the topic page.
• **webhelp.fragment.after.topic.breadcrumb** - Displays the specified XHTML fragment after the breadcrumb component in the topic page.
• **webhelp.fragment.before.publication.toc** - Displays the specified XHTML fragment before the publication's table of contents component in the topic page.
- `webhelp.fragment.after.publication.toc` - Displays the specified XHTML fragment after the publication's table of contents component in the topic page.
- `webhelp.fragment.before.topic.content` - Displays the specified XHTML fragment before the topic's main content in the topic page.
- `webhelp.fragment.after.topic.content` - Displays the specified XHTML fragment after the topic's main content in the topic page.
- `webhelp.fragment.before.feedback` - Displays the specified XHTML fragment before the Oxygen Feedback commenting component in the topic page.
- `webhelp.fragment.after.feedback` - Displays the specified XHTML fragment after the Oxygen Feedback commenting component in the topic page.
- `webhelp.fragment.before.topic.toc` - Displays the specified XHTML fragment before the topic's table of contents component in the topic page.
- `webhelp.fragment.after.topic.toc` - Displays the specified XHTML fragment after the topic's table of contents component in the topic page.

Search Results Page Placeholder Parameters

The following placeholder parameters can be used in the search results page. The parameter values can be either a well-formed XHTML fragment or a path to a file that contains a well-formed XHTML fragment:

- `webhelp.fragment.head.search.page` - Displays the specified XHTML fragment in the header section.
- `webhelp.fragment.after.header.search.page` - Displays the specified XHTML fragment after the header section.
- `webhelp.fragment.before.body.search.page` - Displays the specified XHTML fragment before the body.
- `webhelp.fragment.before.search.input.search.page` - Displays the specified XHTML fragment before the search input component.
- `webhelp.fragment.after.search.input.search.page` - Displays the specified XHTML fragment after the search input component.
- `webhelp.fragment.before.main.content.area.search.page` - Displays the specified XHTML fragment before the main content area.
- `webhelp.fragment.after.main.content.area.search.page` - Displays the specified XHTML fragment after the main content area.
- `webhelp.fragment.after.body.search.page` - Displays the specified XHTML fragment after the body.
- `webhelp.google.search.script` - Replaces the search input component with a Google search component.

Index Terms Page Placeholder Parameters

The following placeholder parameters can be used in the search results page. The parameter values can be either a well-formed XHTML fragment or a path to a file that contains a well-formed XHTML fragment:

- `webhelp.fragment.head.terms.page` - Displays the specified XHTML fragment in the header section.
- `webhelp.fragment.after.header.terms.page` - Displays the specified XHTML fragment after the header section.
Using String Values in Placeholder Parameter Values

If you use strings for values of HTML fragment placeholder parameter values, the string values are written to files in the transformation's temporary directory. The values of the associated parameters reference the paths of the temporary files. This means that the HTML fragments will have a uniform processing in the WebHelp's XSLT Module.

**Example:**

Suppose the placeholder parameter has the following string value:

```
# String value
webhelp.fragment.welcome = <p>This is an HTML paragraph.</p>
```

A new file that contains the parameter's value is created:

```
[temp-dir]/whr-html-fragments/webhelp_fragment_welcome.xml
<p>This is an HTML paragraph.</p>
```

The parameter's value then becomes:

```
# Absolute file path as value
webhelp.fragment.welcome= [temp-dir]/whr-html-fragments/webhelp_fragment_welcome.xml
```

**Related Information:**

[How to Insert Custom HTML Content (on page 1654)](#)

**WebHelp Responsive Macros**

You can use the `whc:macro` layout component to specify a macro value (a variable that will be expanded when the output files are generated).

A macro has the following syntax:

```
${macro-name}
```

or
A macro name can accept any alphanumeric characters, as well as the following characters: - (minus), _ (underscore), . (dot), : (colon). The value of a parameter may contain any character except the } (close curly bracket) character.

Implementations

The following macros are supported:

i18n
For localizing a string.

```xml
${i18n(string.id)}
```

param
Returns the value of a transformation parameter.

```xml
${param(webhelp.show.main.page.tiles)}
```

env
Returns the value of an environment variable.

```xml
${env(JAVA_HOME)}
```

system-property
Returns the value of a system property.

```xml
${system-property(os.name)}
```

timestamp
Can be used to format the current date and time. Accepts a string (as a parameter) that determines how the date and time will be formatted (format string or picture string as it is known in the XSLT specification). The format string must comply with the rules of the XSLT format dateTime function specification.

```xml
${timestamp([h1]:[m01] [P] [M01]/[D01]/[Y0001])}
```

path
Returns the path associated with the specified path ID. The following paths IDs are supported:

- **oxygen-webhelp-output-dir** - The path to the output directory. The path is relative to the current HTML file.
- **oxygen-webhelp-assets-dir** - The path to the oxygen-webhelp subdirectory from the output directory. The path is relative to the current HTML file.
- **oxygen-webhelp-template-dir** - The path to the template directory. The path is relative to the current HTML file.

```xml
${path(oxygen-webhelp-template-dir)}
```
Note:
New paths IDs can be added by overriding the `wh-macro-custom-path` template from `com.oxygenxml.webhelp.responsive\xsl\template\macroExpander.xsl`:

```xml
<!-- Extension template for expanding a custom path macro. -->
<xsl:template name="wh-macro-custom-path">
  <xsl:param name="pathId"/>
  <xsl:value-of select="$pathId"/>
</xsl:template>
```

**map-xpath**

Can be used to execute an XPath expression over the DITA map file from the temporary directory.

**Tip:**
Available in all template layout HTML pages.

```
${map-xpath(/map/title)}
```

**topic-xpath**

Can be used to execute an XPath expression over the current topic.

**Tip:**
Available only in the topic HTML page template (`wt_topic.html`).

```
${topic-xpath(string-join(//shortdesc//text(), ' '))}
```

**oxygen-webhelp-build-number**

Returns the current WebHelp distribution ID (build number).

```
${oxygen-webhelp-build-number}
```

**Extensibility**

To add new macros, you can add an XSLT extension to overwrite the `wh-macro-extension` template from the `com.oxygenxml.webhelp.responsive\xsl\template\macroExpander.xsl` file.

```xml
<!-- Extension template for expanding custom macro constructs -->
<xsl:template name="wh-macro-extension">
  <xsl:param name="name"/>
  <xsl:param name="params"/>
  <xsl:param name="contextNode"/>
  <xsl:param name="matchedString"/>
</xsl:template>
```
The `wh-macro-extension` template has the following parameters:

- **name** - The name of the current macro.
- **params** - List of parameters of the current macro as a string sequence. The current macros parsing mechanism only allows macros with a maximum of one parameter. Consequently, this list will contain at most one element.
- **contextNode** - The current element or attribute where the macro was declared.
- **matchedString** - The entire value of the matched macro as specified in the HTML template page.

### Combining WebHelp Responsive and PDF Customizations in a Template Package

An Oxygen Publishing Template package can contain both a WebHelp Responsive and PDF customization in the same template package and you can use that same template in both types of transformations. The template descriptor file can define the customization for both types by including both a `<webhelp>` and `<pdf>` element and some of the resources can be reused. Resources referenced in elements in the `<webhelp>` element will only be used for WebHelp transformations, and resources referenced in the elements in the `<pdf>` element will only be used in PDF transformations.
Related Information:
Publishing Template Package Contents for PDF Customizations (on page 1785)

**HTML Page Layout Files**

The HTML page layout files define the default layout of the generated pages in the output for the built-in template. There are four types of pages (main, search, topic, index) and each type of page is a simple HTML file. Each page type has various components that appear by default and each component has a corresponding element and when that element is included in the HTML file, the corresponding components will appear in the output.

⚠️ **Warning:**

It is no longer recommended for you to customize these files because if you upgrade to a newer version of Oxygen, those files may no longer produce the desired results and if new components have been added, you won't have access to them. Instead, use any of the other methods described in Publishing Template Package Contents for WebHelp Responsive Customizations (on page 1606).

If you do choose to customize these HTML files, each type of page is defined inside an `<html-page-layout-files>` element in the descriptor file.
If you do use the `html-page-layout-files` element, you must specify all four types of pages (main, search, topic, index). When not specified, the files from the DITA-OT-DIR/plugins/com.oxygenxml.webhelp.responsive/oxygen-webhelp/page-templates folder will be used to define the layout of each type of page.

**HTML Page Components**

Each type of page contains various components that control the layout of that page. The rendering of each component depends on the context where it is placed and its content depends on the transformed DITA map (on page 3218).

Some of the components can be used in all four types of pages, while some are only available for certain pages. For instance, the Publication Title component can be used in all pages, but the Navigation Breadcrumb component can only be used in the Topic Page.

To include a component in the output of a particular type of page, you have to reference a specific element in that particular HTML file. All the elements associated with a component should belong to the `http://www.oxygenxml.com/webhelp/components` namespace.

Every component can contain custom content or reference another component. To specify where the component content will be located in the output, you can use the `<whc:component_content>` element as a descendant of the component element. It can specify the location as before, after, or it can wrap the component content. The following snippet contains an example of each:

```xml
<whc:webhelp_search_input class="navbar-form wh_main_page_search"
    role="form">
    <div class="custom-content-before">Enter search terms here:</div>
    <div class="custom-wrapper">
        <whc:component_content/>
    </div>
    <div class="custom-content-after">Results will be displayed in a new window.</div>
</whc:webhelp_search_input>
```
Main Page

The Main Page is the home page generated in the WebHelp Responsive output. The name of the HTML file that defines this page is `wt_index.html` and it is located in the following directory: `DITA-OT-DIR/plugins/com.oxygenxml.webhelp.responsive/oxygen-webhelp/page-templates`.

The main function of the home page is to display top-level information and provide links that help you easily navigate to any of the top-level topics of the publication. These links can be rendered in either a Tiles or Tree style of layout. The HTML page produced for the home page also consists of various other components, such as a logo, title, menu, search field, or index link.

Figure 493. Examples of Main Page Components for a Tiles Style of Layout

1. Publication Logo (on page 1625)
2. Publication Title (on page 1625)
3. Search Input (on page 1626)
4. Main Menu (on page 1626)
5. Index Terms Link (on page 1627)
6. Topic Tiles (on page 1626)
7. Print Link (on page 1626)
Figure 494. Examples of Main Page Components for a Tree Style of Layout

1. Publication Logo (on page 1625)
2. Publication Title (on page 1625)
3. Search Input (on page 1626)
4. Main Menu (on page 1626)
5. Index Terms Link (on page 1627)
6. Table of Contents (on page 1627)
7. Print Link (on page 1626)

The following components can be referenced in the Main Page (wt_index.html) file:

**Publication Title** (webhelp_publication_title)

This component generates the publication title in the output. To generate this component, the `<whc:webhelp_publication_title>` element must be specified in the HTML file as in the following example:

```html
<whc:webhelp_publication_title
```

In the output, you will find an element with the class: `wh_publication_title`.

**Publication Logo** (webhelp_logo)

This component generates a logo image in the output. To generate this component, the `<whc:webhelp_logo>` element must be specified in the HTML file as in the following example:
In addition, you must also specify the path of the logo image in the webhelp.logo.image transformation parameter (in the Parameters tab in the transformation scenario). You can set the webhelp.logo.image.target.url parameter to generate a link to a URL when you click the logo image.

In the output, you will find an element with the class: wh_logo.

Search Input (webhelp_search_input)

This component is used to generate the input widget associated with search function in the output. To generate this component, the <whc:webhelp_search_input> element must be specified in the HTML file as in the following example:

```
<whc:webhelp_search_input
```

In the output, you will find an element with the class: wh_search_input.

Print Link (webhelp_print_link)

This component is used to generate a print icon that opens the print dialog box for your particular browser. To generate this component, the <whc:webhelp_print_link> element must be specified in the HTML file as in the following example:

```
<whc:webhelp_print_link
```

In the output, you will find an element with the class: wh_print_link.

Main Menu (webhelp_top_menu)

This component generates a menu with all the documentation topics. To generate this component, the <whc:webhelp_top_menu> element must be specified in the HTML file as in the following example:

```
<whc:webhelp_top_menu
```

In the output, you will find an element with the class: wh_top_menu.

You can control the maximum level of topics that will be included in the menu using the webhelp.top.menu.depth transformation parameter (in the Parameters tab of the transformation scenario).

For information about customizing the menu, see How to Customize the Menu (on page 1665).

Main Page Topic Tiles (webhelp_tiles)
This component generates the tiles section in the main page. This section will contain a tile for each root topic of the published documentation. Each topic tile has three sections that correspond to the topic title, short description, and image. To generate this component, the `<whc:webhelp_tiles>` element must be specified in the HTML file as in the following example:

```
<whc:webhelp_tiles
```

In the output, you will find an element with the class: `wh_tiles`.

If you want to control the HTML structure that is generated for a WebHelp tile you can also specify the template for a tile by using the `<whc:webhelp_tile>` component, as in the following example:

```
<whc:webhelp_tile
class="col-md-4">
    <!-- Place holder for tile's image -->
    <whc:webhelp_tile_image/>

    <div
        class="wh_tile_text">
        <!-- Place holder for tile's title -->
        <whc:webhelp_tile_title/>

        <!-- Place holder for tile's shordesc -->
        <whc:webhelp_tile_shortdesc/>
    </div>
</whc:webhelp_tile>
```

For information about customizing the tiles, see How to Configure the Tiles on the WebHelp Responsive Main Page (on page 1670).

**Main Page Table of Contents (webhelp_main_page_toc)**

This component generates a simplified Table of Contents. It is simplified because it contains only two levels from the documentation hierarchy. To generate this component, the `<whc:webhelp_main_page_toc>` element must be specified in the HTML file as in the following example:

```
<whc:webhelp_main_page_toc
```

In the output, you will find an element with the class: `wh_main_page_toc`.

**Index Terms Link (webhelp_indexterms_link)**

This component can be used to generate a link to the index terms page (`indexterms.html`). If the published documentation does not contain any index terms, then the link will not be generated. To generate this component, the `<whc:webhelp_indexterms_link>` element must be specified in the HTML file as in the following example:
In the output, you will find an element with the class: `wh_indexterms_link`. This component will contain a link to the `indexterms.html` page.

**Link to Skins Resources (`webhelp_skin_resources`)**

This component can be used to add a link to resources for the current WebHelp skin (such as the CSS file). To generate this component, the `<whc:webhelp_skin_resources>` element must be specified in the HTML file as in the following example:

```xml
<whc:webhelp_skin_resources/>
```

In the output, you will find a link to the skin resources.

**Topic Page**

The *Topic Page* is the page generated for each DITA topic in the WebHelp Responsive output. The name of the HTML file that defines this page is `wt_topic.html` and it is located in the following directory: `DITA-OT-DIR/plugins/com.oxygenxml.webhelp.responsive/oxygen-webhelp/page-templates`.

The HTML pages produced for each topic consist of the topic content along with various other additional components, such as a title, menu, navigation breadcrumb, print icon, or side table of contents.
Figure 495. Examples of Topic Page Components

1. Publication Logo (on page 1630)
2. Publication Title (on page 1629)
3. Search Input (on page 1630)
4. Main Menu (on page 1632)
5. Index Terms Link (on page 1632)
6. Expand/Collapse All Sections (on page 1632)
7. Navigation Links (on page 1630)
8. Print Link (on page 1631)
9. Breadcrumb (on page 1630)
10. Publication Table of Contents (on page 1631)
11. Topic Content (on page 1631)
12. Topic Table of Contents (on page 1631)

The following components can be referenced in the Topic Page (wt_topic.html) file:

Publication Title (webhelp_publication_title)
This component generates the publication title in the output. To generate this component, the `<whc:webhelp_publication_title>` element must be specified in the HTML file as in the following example:

```xml
<whc:webhelp_publication_title
```

In the output, you will find an element with the class: `wh_publication_title`.

Publication Logo (webhelp_logo)

This component generates a logo image in the output. To generate this component, the `<whc:webhelp_logo>` element must be specified in the HTML file as in the following example:

```xml
<whc:webhelp_logo
```

In addition, you must also specify the path of the logo image in the `webhelp.logo.image` transformation parameter (in the Parameters tab in the transformation scenario). You can set the `webhelp.logo.image.target.url` parameter to generate a link to a URL when you click the logo image.

In the output, you will find an element with the class: `wh_logo`.

Search Input (webhelp_search_input)

This component is used to generate the input widget associated with search function in the output. To generate this component, the `<whc:webhelp_search_input>` element must be specified in the HTML file as in the following example:

```xml
<whc:webhelp_search_input
```

In the output, you will find an element with the class: `wh_search_input`.

Topic Breadcrumb (webhelp_breadcrumb)

This component generates a breadcrumb that displays the path of the current topic. To generate this component, the `<whc:webhelp_breadcrumb>` element must be specified in the HTML file as in the following example:

```xml
<whc:webhelp_breadcrumb
```

In the output, you will find an element with the class: `wh_breadcrumb`. This component will contain a list with items that correspond to the topics in the path. The first item in the list has a link to the main page with the `home` class. The last item in the list corresponds to the current topic and has the `active` class set.

Navigation Links (webhelp_navigation_links)
This component generates navigation links to the next and previous topics. To generate this component, the `<whc:webhelp_navigation_links>` element must be specified in the HTML file as in the following example:

```xml
<whc:webhelp_navigation_links
```

In the output, you will find an element with the class: `wh_navigation_links`. This component will contain the links to the next and previous topics.

**Print Link (webhelp_print_link)**

This component is used to generate a print icon that opens the print dialog box for your particular browser. To generate this component, the `<whc:webhelp_print_link>` element must be specified in the HTML file as in the following example:

```xml
<whc:webhelp_print_link
```

In the output, you will find an element with the class: `wh_print_link`.

**Topic Content (webhelp_topic_content)**

This component generates the content of a topic and it represent the content of the HTML files as they are produced by the DITA-OT processor. To generate this component, the `<whc:webhelp_topic_content>` element must be specified in the HTML file as in the following example:

```xml
<whc:webhelp_topic_content
```

In the output, you will find an element with the class: `wh_topic_content`.

**Publication TOC (webhelp_publication_toc)**

This component generates a mini table of contents for the current topic (on the left side). It will contain links to the children of the current topic, its siblings, and all of its ancestors. To generate this component, the `<whc:webhelp_publication_toc>` element must be specified in the HTML file as in the following example:

```xml
<whc:webhelp_publication_toc
```

In the output, you will find an element with the class: `wh_publication_toc`. This component will contain links to the topics referenced in the DITA map. It also includes an expand/collapse button (either `<` to collapse or the `>` to expand).

**Topic TOC (webhelp_topic_toc)**

This component generates a topic table of contents for the current topic (on the right side) with a heading named **On this page**. It contains links to each section within the current topic and
the section corresponding to the current scroll position is highlighted. The topic must contain at least two `<section>` elements and each `<section>` must have an `@id` attribute. To generate this component, the `<whc:webhelp_topic_toc>` element must be specified in the HTML file as in the following example:

```xml
<whc:webhelp_topic_toc
```

In the output, you will find an element with the class: `wh_topic_toc`. This component will contain links to the sections within the current topic. It also includes an expand/collapse button (either ➔ to collapse or the ➚ to expand).

**Expand/Collapse Sections** *(webhelp_expand Collapse_sections)*

This component is used to generate an icon that expands or collapses sections listed in the side table of contents within a topic. To generate this component, the `<whc:webhelp_expand Collapse_sections>` element must be specified in the HTML file as in the following example:

```xml
<whc:webhelp_expand Collapse_sections
```

In the output, you will find an element with the class: `webhelp_expand Collapse_sections`.

**Topic Feedback** *(webhelp_feedback)*

This component generates a placeholder for where the comments section will be presented. To generate this component, the `<whc:webhelp_feedback>` element must be specified in the HTML file as in the following example:

```xml
<whc:webhelp_feedback
```

**Main Menu** *(webhelp_top_menu)*

This component generates a menu with all the documentation topics. To generate this component, the `<whc:webhelp_top_menu>` element must be specified in the HTML file as in the following example:

```xml
<whc:webhelp_top_menu
```

In the output, you will find an element with the class: `wh_top_menu`.

You can control the maximum level of topics that will be included in the menu using the `webhelp.top.menu.depth` transformation parameter (in the Parameters tab of the transformation scenario).

For information about customizing the menu, see How to Customize the Menu *(on page 1665)*.

**Index Terms Link** *(webhelp_indexterms_link)*
This component can be used to generate a link to the index terms page (indexterms.html). If the published documentation does not contain any index terms, then the link will not be generated. To generate this component, the `<whc:webhelp_indexterms_link>` element must be specified in the HTML file as in the following example:

```xml
<whc:webhelp_indexterms_link
```

In the output, you will find an element with the class `wh_indexterms_link`. This component will contain a link to the indexterms.html page.

**Child Links** *(webhelp_child_links)*

For all topics with subtopics (child topics), this component generates a list of links to each child topic. To generate this component, the `<whc:webhelp_child_links>` element must be specified in the HTML file as in the following example:

```xml
<whc:webhelp_child_links
```

**Related Links** *(webhelp_related_links)*

For all topics that contain related links, this component generates a list of related links that will appear in the output. To generate this component, the `<whc:webhelp_related_links>` element must be specified in the HTML file as in the following example:

```xml
<whc:webhelp_related_links
```

**Link to Skins Resources** *(webhelp_skin_resources)*

This component can be used to add a link to resources for the current WebHelp skin (such as the CSS file). To generate this component, the `<whc:webhelp_skin_resources>` element must be specified in the HTML file as in the following example:

```xml
<whc:webhelp_skin_resources/>
```

In the output, you will find a link to the skin resources.

**Search Results Page**

The *Search Results Page* is the page generated that presents search results in the WebHelp Responsive output. The name of the HTML file that defines this page is `wt_search.html` and it is located in the following directory: `DITA-OT-DIR/plugins/com.oxygenxml.webhelp.responsive/oxygen-webhelp/page-templates`.

The HTML page that is produced consists of a search results component along with various other additional components, such as a title, menu, or index link.
The following components can be referenced in the Search Results Page (\texttt{wt\_search.html}) file:

**Publication Title** (\texttt{webhelp\_publication\_title})

This component generates the publication title in the output. To generate this component, the \texttt{<whc:webhelp\_publication\_title>} element must be specified in the HTML file as in the following example:

\begin{verbatim}
\end{verbatim}

In the output, you will find an element with the class \texttt{wh\_publication\_title}.

**Publication Logo** (\texttt{webhelp\_logo})
This component generates a logo image in the output. To generate this component, the `<whc:webhelp_logo>` element must be specified in the HTML file as in the following example:

```xml
<whc:webhelp_logo
```

In addition, you must also specify the path of the logo image in the `webhelp.logo.image` transformation parameter (in the Parameters tab in the transformation scenario). You can set the `webhelp.logo.image.target.url` parameter to generate a link to a URL when you click the logo image.

In the output, you will find an element with the class: `wh_logo`.

**Search Input (webhelp_search_input)**

This component is used to generate the input widget associated with search function in the output. To generate this component, the `<whc:webhelp_search_input>` element must be specified in the HTML file as in the following example:

```xml
<whc:webhelp_search_input
```

In the output, you will find an element with the class: `wh_search_input`.

**Search Results (webhelp_search_results)**

This component is used to generate a placeholder to signal where the search results will be presented in the output. To generate this component, the `<whc:webhelp_search_results>` element must be specified in the HTML file as in the following example:

```xml
<whc:webhelp_search_results
```

In the output, you will find an element with the class: `wh_search_results`.

**Print Link (webhelp_print_link)**

This component is used to generate a print icon that opens the print dialog box for your particular browser. To generate this component, the `<whc:webhelp_print_link>` element must be specified in the HTML file as in the following example:

```xml
<whc:webhelp_print_link
```

In the output, you will find an element with the class: `wh_print_link`.

**Main Menu (webhelp_top_menu)**

This component generates a menu with all the documentation topics. To generate this component, the `<whc:webhelp_top_menu>` element must be specified in the HTML file as in the following example:
In the output, you will find an element with the class: `wh_top_menu`.

You can control the maximum level of topics that will be included in the menu using the `webhelp.top.menu.depth` transformation parameter (in the Parameters tab of the transformation scenario).

For information about customizing the menu, see How to Customize the Menu (on page 1665).

**Index Terms Link (`webhelp_indexterms_link`)**

This component can be used to generate a link to the index terms page (`indexterms.html`). If the published documentation does not contain any index terms, then the link will not be generated. To generate this component, the `<whc:webhelp_indexterms_link>` element must be specified in the HTML file as in the following example:

```
<whc:webhelp_indexterms_link
```

In the output, you will find an element with the class: `wh_indexterms_link`. This component will contain a link to the `indexterms.html` page.

**Link to Skins Resources (`webhelp_skin_resources`)**

This component can be used to add a link to resources for the current WebHelp skin (such as the CSS file). To generate this component, the `<whc:webhelp_skin_resources>` element must be specified in the HTML file as in the following example:

```
<whc:webhelp_skin_resources/>
```

In the output, you will find a link to the skin resources.

**Index Terms Page**

The Index Terms Page is the page generated that presents index terms in the WebHelp Responsive output. The name of the HTML file that defines this page is `wt_terms.html` and it is located in the following directory: `DITA-OT-DIR/plugins/com.oxygenxml.webhelp.responsive/oxygen-webhelp/page-templates`.

The HTML page that is produced consists of an index terms section along with various other additional components, such as a title, menu, or search field.

An alphabet that contains the first letter of the documentation index terms is generated at the top of the index page. Each letter represents a link to a specific indices section.
The following components can be referenced in the Index Terms Page (wt_terms.html) file:

**Publication Title (webhelp_publication_title)**

This component generates the publication title in the output. To generate this component, the `<whc:webhelp_publication_title>` element must be specified in the HTML file as in the following example:

```xml
<whc:webhelp_publication_title
```

In the output, you will find an element with the class: `wh_publication_title`.

**Publication Logo (webhelp_logo)**

This component generates a logo image in the output. To generate this component, the `<whc:webhelp_logo>` element must be specified in the HTML file as in the following example:
In addition, you must also specify the path of the logo image in the webhelp.logo.image transformation parameter (in the Parameters tab in the transformation scenario). You can set the webhelp.logo.image.target.url parameter to generate a link to a URL when you click the logo image.

In the output, you will find an element with the class: wh_logo.

**Search Input (webhelp_search_input)**

This component is used to generate the input widget associated with search function in the output. To generate this component, the `<whc:webhelp_search_input>` element must be specified in the HTML file as in the following example:

```xml
<whc:webhelp_search_input
```

In the output, you will find an element with the class: wh_search_input.

**Print Link (webhelp_print_link)**

This component is used to generate a print icon that opens the print dialog box for your particular browser. To generate this component, the `<whc:webhelp_print_link>` element must be specified in the HTML file as in the following example:

```xml
<whc:webhelp_print_link
```

In the output, you will find an element with the class: wh_print_link.

**Main Menu (webhelp_top_menu)**

This component generates a menu with all the documentation topics. To generate this component, the `<whc:webhelp_top_menu>` element must be specified in the HTML file as in the following example:

```xml
<whc:webhelp_top_menu
```

In the output, you will find an element with the class: wh_top_menu.

You can control the maximum level of topics that will be included in the menu using the webhelp.top.menu.depth transformation parameter (in the Parameters tab of the transformation scenario).

For information about customizing the menu, see How to Customize the Menu (on page 1665).

**Index Terms Link (webhelp_indexterms_link)**
This component can be used to generate a link to the index terms page (indexterms.html). If the published documentation does not contain any index terms, then the link will not be generated. To generate this component, the `<whc:webhelp_indexterms_link>` element must be specified in the HTML file as in the following example:

```xml
<whc:webhelp_indexterms_link
```

In the output, you will find an element with the class: `wh_indexterms_link`. This component will contain a link to the indexterms.html page.

**Link to Skins Resources (webhelp_skin_resources)**

This component can be used to add a link to resources for the current WebHelp skin (such as the CSS file). To generate this component, the `<whc:webhelp_skin_resources>` element must be specified in the HTML file as in the following example:

```xml
<whc:webhelp_skin_resources/>
```

In the output, you will find a link to the skin resources.

**Generating WebHelp Responsive Output**

The publishing process can be initiated from a transformation scenario within Oxygen XML Editor/Author, from a command line outside Oxygen XML Editor/Author, or from an integration server.

**Running WebHelp Responsive from Oxygen XML Editor/Author**

To publish a DITA map (on page 3218) as WebHelp Responsive output, follow these steps:

1. Select the Configure Transformation Scenario(s) action from the DITA Maps Manager toolbar.
2. Select the DITA Map WebHelp Responsive scenario from the DITA Map section.
3. If you want to configure the transformation, click the Edit button.

**Step Result:** This opens an Edit scenario configuration dialog box that allows you to configure various options in the following tabs:

- **Templates Tab** - This tab contains a set of built-in skins that you can use for the layout of your WebHelp system output.
- **Parameters Tab** - This tab includes numerous transformation parameters that can be set to customize your WebHelp system output.
- **Feedback Tab** - This tab is for those who want to add the Oxygen Feedback comments component at the bottom of each WebHelp page so that you can interact with your readers.
- **Filters Tab** - This tab allows you to filter certain content elements from the generated output.
Advanced Tab - This tab allows you to specify some advanced options for the transformation scenario.

Output Tab - This tab allows you to configure options that are related to the location where the output is generated.

4. Click **Apply associated** to process the transformation.

**Result:** When the DITA Map WebHelp Responsive transformation is complete, the output is automatically opened in your default browser.

**Automating the WebHelp Responsive Output for DITA**

DITA-based WebHelp output can be generated from an automated publishing process using a command line outside of Oxygen XML Editor/Author or an automatic publishing system, such as Jenkins or Travis. However, to do this, you must purchase an additional Oxygen XML WebHelp license.

**Related Information:**
- Generating WebHelp Responsive Output for DITA

**Deploying an Oxygen Feedback Comments Component**

You can add a comments component in your WebHelp Responsive output to provide a simple and efficient way for your community to interact and offer feedback. The comments component is contributed by Oxygen Feedback, a modern comment management system that can be integrated with your WebHelp Responsive output to provide a comments area at the bottom of each WebHelp page where readers can add new comments or reply to existing ones.

Oxygen Feedback includes a modern, user-friendly administration interface where you can moderate comments, manage users, view statistics, and configure settings. It is very easy to integrate and there are no requirements for installing additional software. You simply need to create an Oxygen Feedback site configuration in the administration interface, copy the HTML installation fragment that is generated at the end of the creation process, and paste the generated fragment in the Feedback tab in the WebHelp Responsive transformation scenario dialog box (on page 3113).

An add-on is also available that contributes a Feedback Comments Manager view in Oxygen XML Editor/Author where the documentation team can see all the comments added in your WebHelp output. This means they can react to user feedback by making corrections and updating the source content without leaving the application.

**Adding the Feedback System to WebHelp Responsive Documentation**

**Prerequisite**

To install and manage Oxygen Feedback, you will need to obtain a license for the product. This requires that you choose a subscription plan during the installation procedure. To see the subscription plans prior to installing the product, go to: https://www.oxygenxml.com/oxygen_feedback/buy_feedback.html.
Installation Procedure

1. Log in to your Feedback account from the administration login page ([https://feedback.oxygenxml.com/login](https://feedback.oxygenxml.com/login)). You can click on Log in with Google or Log in with Facebook to create an account using your Google or Facebook credentials, or click the Sign Up tab to create an account using your name and email address.

2. Click the Add site button to create a site configuration. If you have not already selected a subscription plan, you will be directed to a page where you can choose from several options.

3. In the Settings page, enter a Name and Description for the site configuration. There are some optional settings that can be adjusted according to your needs. For more details, see the Site Settings topic. Click Continue.

4. In the Initial version page, enter the Base URL for your website (you can add additional URLs by clicking the Add button). You can also specify an Initial version if you want it to be something other than 1.0. If you do not plan to have multiple versions, leave the version as 1.0. For more details, see the Initial Version topic. Click Continue.

5. In the Installation page, choose a site generation option:
   a. If you will generate the documentation using a transformation scenario in Oxygen XML Editor/Author, select the Oxygen XML Editor option and continue with these steps:
      i. Copy the generated HTML fragment and click Finish.
      ii. In Oxygen XML Editor/Author, open the Configure Transformation Scenario(s) dialog box.
      iii. Select and duplicate the DITA Map WebHelp Responsive scenario.
      iv. Go to the Feedback tab.
      v. Click the Edit button and paste the generated installation fragment.
   b. If you will generate the documentation using a command-line script, select the Oxygen XML WebHelp option and continue with these steps:
      i. Copy the generated HTML fragment and click Finish.
      ii. Create an XML file (for example, feedback-install.xml) with the generated installation fragment.
      iii. Use the webhelp.fragment.feedback parameter in your command-line script to specify the path to the file you just created. For example:

         ```
         dita.bat -Dwebhelp.fragment.feedback=c:\path\to\feedback-install.xml
         ```

6. [Optional] If you want the Oxygen Feedback comments component to fill the entire page width, contribute a custom CSS file (use the args.css parameter to reference it) that contains the following style rule:

   ```
   div.footer {
     float: none;
   }
   ```

For more details about Oxygen Feedback, how to configure settings, moderate comments, view statistics, and much more, see the Oxygen Feedback user guide.
Also, to see a demonstration of Oxygen Feedback being integrated into WebHelp Responsive output, watch our Webinar: DITA Publishing and Feedback with Oxygen Tools.

**Customizing WebHelp Responsive Output**

Oxygen XML Editor provides support for customizing the WebHelp Responsive output to suit your specific needs. The WebHelp Responsive output is based upon the Bootstrap responsive front-end framework and is available for DITA document types.

To change the overall appearance of your WebHelp Responsive output, you can use several different customization methods or a combination of methods. If you are familiar with CSS and coding, you can style your WebHelp output through your own custom stylesheets. You can also customize your output by modifying existing templates, create your own layout pages, or by configuring certain options and parameters in the transformation scenario.

This section includes topics that explain various ways to customize your WebHelp Responsive system output, such as how to configure the tiles on the main page, add logos in the title area, integrate with social media, localizing the interface, and much more.

For an in-depth look at WebHelp Responsive features and some customization tips, watch our Webinar: DITA Publishing and Feedback with Oxygen Tools.

**Working with Publishing Templates**

An Oxygen Publishing Template (on page 3221) defines all aspects of the layout and styles of the WebHelp Responsive output. It is a self-contained customization package stored as a ZIP archive or folder that can easily be shared with others. It provides the primary method for customizing the output. The recommended method for customizing the WebHelp Responsive output is to use a custom publishing template.

This section contains topics about how to create, edit, publish, and share publishing templates.

Related Information:

Publishing Template Package Contents for WebHelp Responsive Customizations (on page 1606)

**How to Create a Publishing Template**

To create a customization, you can start from scratch or from an existing template, and then adapt it according to your needs.

**Creating a Publishing Template Starting from Scratch**

To create a new Oxygen Publishing Template, follow these steps:

1. Create a folder that will contain all the template files.

2. In Oxygen XML Editor/Author, open the new document wizard (use File > New or the New toolbar button), then choose the Publishing Template Descriptor template.
3. Save the .opt file into your customization directory.
4. Open the .opt file in the editor and customize it to suit your needs.

Creating a Publishing Template Starting from an Existing Template

If you are using a WebHelp Responsive or DITA Map PDF - based on HTML5 & CSS transformation, the easiest way to create a new Oxygen Publishing Template (on page 3221) is to select an existing template in the transformation scenario dialog box and use the Save template as button to save that template into a new template package that can be used as a starting point.

To create a new Oxygen Publishing Template, follow these steps:

1. Open the transformation scenario dialog box and select the publishing template you want to export and use as a starting point.

2. Optional: You can set one or more transformation parameters from the Parameters tab and the edited parameters will be exported along with the selected template. You will see which parameters will be exported in the dialog box that is displayed after the next step.

3. Click the Save template as button.

Step Result: This opens a template package configuration dialog box that contains some options and displays the parameters that will be exported to your template package.
4. Specify a name for the new template.
5. **Optional:** Specify a template description.
6. **Optional:** The same publishing template package can contain both a WebHelp Responsive and PDF customization and you can use the same template in both types of transformations (DITA Map WebHelp Responsive or DITA Map to PDF - based on HTML5 & CSS). You can use the Include WebHelp customization and Include PDF customization options to specify whether your custom template will include both types of customizations.
7. **Optional:** For WebHelp Responsive customizations, you can select the Include HTML Page Layout Files option if you want to copy the default HTML Page Layout Files (on page 1622) in your template package. They are helpful if you want to change the structure of the generated HTML pages.
8. In the Save as field, specify the name and path of the ZIP file where the template will be saved.

**Step Result:** A new ZIP archive will be created on disk in the specified location with the specified name.
9. Open the .opt file in the editor and customize it to suit your needs.

For more information about creating and customizing publishing templates, watch our video demonstration:

https://www.youtube.com/embed/zNmXfKWXw08

**Related Information:**
- Publishing Template Package Contents for PDF Customizations (on page 1785)
- Publishing Template Package Contents for WebHelp Responsive Customizations (on page 1606)

**How to Edit a Packed Publishing Template**

To edit an existing *Oxygen Publishing Template* (on page 3221) package, follow these steps:

1. Unzip the ZIP archive associated with the *Oxygen Publishing Template* in a separate folder.
2. Link the folder associated with the template in the Project view.
3. Using the Project view, you can modify the resources (CSS, JS, fonts) within the *Oxygen Publishing Template* folder to fit your needs.
4. Open the publishing template descriptor file (.opt extension) in the editor and modify it to suit your needs.
5. **Optional:** Once you finish your customization, you can archive the folder as a ZIP file.

**Related Information:**
- Publishing Template Package Contents for PDF Customizations (on page 1785)
- Publishing Template Package Contents for WebHelp Responsive Customizations (on page 1606)

**How to Add a Publishing Template to the Publishing Templates Gallery**

To add the publishing template to your templates gallery, follow these steps:
1. Open the transformation scenario dialog box by editing a WebHelp Responsive transformation.

2. In the Templates tab, click the Configure Publishing Templates Gallery link to.
   This will open the preferences page.

3. Click the Add button and specify the location of your template directory.
   Your template directory is now added to the Additional Publishing Templates Galleries list.

4. Click OK to return to the transformation scenario dialog box.
   All the templates contained in your template directory will be displayed in the preview pane along with all the built-in templates.

How to Use a Publishing Template from a Command Line

Before you run the transformation, you need to know if the publishing template has a single template descriptor file or multiple descriptor files (on page 1607). If you don't know, open the ZIP archive or folder and check for files with the .opt extension.

Using a Publishing Template with a Single Descriptor

A template with a single descriptor is used for a single customization.

To run from a command line, you need to use the webhelp.publishing.template parameter (on page 1718). This parameter specifies the path to the ZIP archive (or root folder) that contains your custom WebHelp Responsive template.

**Command-Line Example:**

- **Windows:**

  ```
  dita.bat
  --format=webhelp-responsive
  --input=c:\path\to\mySample.ditamap
  --output=c:\path\to\output
  -Dwebhelp.publishing.template=custom-template
  ```

- **Linux/macOS:**

  ```
  dita
  --format=webhelp-responsive
  --input=/path/to/mySample.ditamap
  --output=/path/to/output
  -Dwebhelp.publishing.template=custom-template
  ```

**Tip:**

You can also start the dita process by passing it a DITA OT Project File. Inside the project file you can specify as parameters for the webhelp-responsive transformation type the WebHelp-related parameters.
Using a Publishing Template with Multiple Descriptors

A template with multiple descriptors contains multiple customizations.

Because the publishing template is self-contained, it is used to reuse resources that are common to multiple publications.

To run from a command line, you need to use the `webhelp.publishing.template` (on page 1718) and `webhelp.publishing.template.descriptor` (on page 1718) parameters.

The `webhelp.publishing.template` (on page 1718) parameter specifies the path to the ZIP archive (or root folder) while the `webhelp.publishing.template.descriptor` (on page 1718) parameter specifies the name of the descriptor you want to use.

**Command-Line Example:**

- **Windows:**
  ```
dita.bat
   --format=webhelp-responsive
   --input=c:\path\to\mySample.ditamap
   --output=c:\path\to\output
   -Dwebhelp.publishing.template=custom-template
   -Dwebhelp.publishing.template.descriptor=flowers.opt
  ```

- **Linux/macOS:**
  ```
dita
   --format=webhelp-responsive
   --input=/path/to/mySample.ditamap
   --output=/path/to/output
   -Dwebhelp.publishing.template=custom-template
   -Dwebhelp.publishing.template.descriptor=flowers.opt
  ```

**Tip:**
You can also start the `dita` process by passing it a DITA OT Project File. Inside the project file you can specify as parameters for the `webhelp-responsive` transformation type the WebHelp-related parameters.

How to Share a Publishing Template

To share a publishing template with others, following these steps:

1. Copy your template in a new folder in your project.
2. Go to **Options > Preferences > DITA > Publishing** and add that new folder to the list.
3. Switch the option as the bottom of that preferences page to **Project Options**.
4. Share your project file (.xpr).
Troubleshooting: Errors Encountered when Loading Templates

When the Templates tab of a WebHelp Responsive transformation scenario dialog box is opened, all templates (built-in and custom) are loaded and validated. Specifically, certain elements in the template descriptor file (on page 1607) are checked for validity. If errors are encountered that prevents the template from loading, the following message will be displayed toward the bottom of the dialog box:

⚠ Some templates could not be loaded. More details

If you click the More details link, a window will open with more information about the encountered error. For example, it might offer a hint that the element is missing from the expected descriptor file structure (on page 1607).

Also, if a template could be loaded, but certain elements could not be found in the descriptor file (on page 1607), a warning icon (⚠️) will be displayed on the template's image (in the Templates tab of the transformation dialog box). For example, this happens if a valid preview-image element (on page 1609) cannot be found.

Converting Old Templates to Newer Versions

WebHelp templates that were created in older versions of Oxygen XML Editor can be converted to the Publishing Template format that was introduced in Oxygen XML Editor version 20.0. This section contains several procedures for converting old templates depending on the version they were created in.

Convert Version 24.0 Publishing Templates to Version 24.1

If you have a custom Publishing Template that was created in Oxygen XML Editor version 24.0, the following conversion procedure is required for the template to be compatible with Oxygen XML Editor version 24.1:

1. In the Project view, add the root directory for your custom Publishing Template (you can use a linked folder (on page 402) and the easiest way to do this is to drag and drop the folder).

   ⚠ Note:
   If your template is stored as a ZIP archive, you first need to unzip it.

2. Expand your template directory, right-click the page-templates subfolder, and select Refactoring > XML Refactoring.
3. In the XML Refactoring dialog box, scroll to the Publishing Template section and select Migrate HTML Page Layout Files to v24.1, then click Next.
4. The Scope should be left as Selected project resources.
5. You can use the Preview button to open a comparison panel where you can review all the changes that will be made by the refactoring operation before applying the changes.
6. Click Finish to perform the conversion.

Result: The converted Publishing Template can now be used in version 24.1.
Convert Version 23 Publishing Templates to Version 24

If you have a custom Publishing Template that was created in Oxygen XML Editor version 23.0 or 23.1, the following conversion procedure is required for the template to be compatible with Oxygen XML Editor version 24:

1. In the Project view, add the root directory for your custom Publishing Template (you can use a linked folder (on page 402) and the easiest way to do this is to drag and drop the folder).

   ![Note:](image)
   
   If your template is stored as a ZIP archive, you first need to unzip it.

2. Expand your template directory, right-click the page-templates subfolder, and select Refactoring > XML Refactoring.
3. In the XML Refactoring dialog box, scroll to the Publishing Template section and select Migrate HTML Page Layout Files to v24, then click Next.
4. The Scope should be left as Selected project resources.
5. You can use the Preview button to open a comparison panel where you can review all the changes that will be made by the refactoring operation before applying the changes.
6. Click Finish to perform the conversion.

Result: The converted Publishing Template can now be used in version 24.
Convert Version 22 Publishing Templates to Version 23

If you have a custom Publishing Template that was created in Oxygen XML Editor version 22.0 or 22.1, it is not necessary to convert it to version 23 because there were no structural changes made for the HTML layout files (on page 1622) between the two versions.

Related information
Convert Version 24.0 Publishing Templates to Version 24.1 (on page 1647)
Convert Version 23 Publishing Templates to Version 24 (on page 1648)
Convert Version 21 Publishing Templates to Version 22 (on page 1649)
Convert Version 20 Publishing Templates to Version 21 (on page 1650)
Convert Version 19 (and Older) Publishing Templates to Version 20 (on page 1650)

Convert Version 21 Publishing Templates to Version 22

If you have a custom Publishing Template that was created in Oxygen XML Editor version 21.0 or 21.1, the following conversion procedure is required for the template to be compatible with Oxygen XML Editor version 22:

1. In the Project view, add the root directory for your custom Publishing Template (you can use a linked folder (on page 402) and the easiest way to do this is to drag and drop the folder).

   **Note:**
   If your template is stored as a ZIP archive, you first need to unzip it.

2. Expand your template directory, right-click the page-templates subfolder, and select Refactoring > XML Refactoring.
3. In the XML Refactoring dialog box, scroll to the Publishing Template section and select Migrate HTML Page Layout Files to v22, then click Next.
4. The Scope should be left as Selected project resources.
5. You can use the Preview button to open a comparison panel where you can review all the changes that will be made by the refactoring operation before applying the changes.
6. Click Finish to perform the conversion.

Result: The converted Publishing Template can now be used in version 22.

Related Information:
Convert Version 24.0 Publishing Templates to Version 24.1 (on page 1647)
Convert Version 23 Publishing Templates to Version 24 (on page 1648)
Convert Version 22 Publishing Templates to Version 23 (on page 1649)
Convert Version 20 Publishing Templates to Version 21

If you have a custom Publishing Template that was created in Oxygen XML Editor version 20.0 or 20.1, the following conversion procedure is required for the template to be compatible with Oxygen XML Editor version 21.0 or 21.1:

1. In the Project view, add the root directory for your custom Publishing Template (you can use a linked folder and the easiest way to do this is to drag and drop the folder).

   **Note:**
   If your template is stored as a ZIP archive, you first need to unzip it.

2. Expand your template directory, right-click the page-templates subfolder, and select Refactoring > XML Refactoring.

3. Convert Version 20 Publishing Templates to Version 21

4. In the XML Refactoring dialog box, scroll to the Publishing Template section and select Migrate HTML Page Layout Files to v21, then click Next.

5. The Scope should be left as Selected project resources.

6. You can use the Preview button to open a comparison panel where you can review all the changes that will be made by the refactoring operation before applying the changes.

7. Click Finish to perform the conversion.

**Result:** The converted Publishing Template can now be used in version 21.0 or 21.1.

---

Related information

- Convert Version 24.0 Publishing Templates to Version 24.1
- Convert Version 23 Publishing Templates to Version 24
- Convert Version 22 Publishing Templates to Version 23
- Convert Version 21 Publishing Templates to Version 22
- Convert Version 19 (and Older) Publishing Templates to Version 20

---

Convert Version 19 (and Older) Publishing Templates to Version 20

With the introduction of the Publishing Template concept in Oxygen XML Editor version 20.0, the old WebHelp output template formats (version 19.1 and older) are no longer supported. However, they can be easily converted to the new format (version 20.0 or 20.1) by following this procedure:
1. Set Oxygen XML Editor to point to the DITA-OT distribution that contains your old template:
   a. Go to Options > Preferences > DITA.
   b. Select Custom in the DITA Open Toolkit section and specify the DITA-OT directory that contains your old template.

2. Edit a WebHelp Responsive transformation scenario.

3. Select your old custom template in the Templates tab.

4. Click on the Save Template as button, complete the required fields, and save the template.

5. Reset the option set in step 1 to its previous value.

6. Edit the WebHelp Responsive transformation scenario again.

7. This time, use the Choose custom template button to select your converted template.

8. Save the scenario and use it to generate the WebHelp Responsive output.

**Result:** The converted template is now in a Publishing Template format for version 20.

### Related information

- Convert Version 24.0 Publishing Templates to Version 24.1 *(on page 1647)*
- Convert Version 23 Publishing Templates to Version 24 *(on page 1648)*
- Convert Version 22 Publishing Templates to Version 23 *(on page 1649)*
- Convert Version 21 Publishing Templates to Version 22 *(on page 1649)*
- Convert Version 20 Publishing Templates to Version 21 *(on page 1650)*

---

**Convert Version 19 (and Older) Publishing Templates to Version 21**

If you have a custom template that was created in Oxygen XML Editor version 19.1 or older and you want to convert it to be compatible with Oxygen XML Editor version 21.0 or 21.1, you need to apply two conversion procedures:

1. Convert the old template to a version 20 Publishing Template by using this procedure: Convert Version 19 (and Older) Publishing Templates to Version 20 *(on page 1650).*

2. Convert that version 20 Publishing Template to version 21 by using this procedure: Convert Version 20 Publishing Templates to Version 21 *(on page 1650).*

**Result:** The converted Publishing Template can now be used in version 21.0 or 21.1.

### Related information

- Convert Version 24.0 Publishing Templates to Version 24.1 *(on page 1647)*
- Convert Version 23 Publishing Templates to Version 24 *(on page 1648)*
- Convert Version 22 Publishing Templates to Version 23 *(on page 1649)*
- Convert Version 21 Publishing Templates to Version 22 *(on page 1649)*
- Convert Version 20 Publishing Templates to Version 21 *(on page 1650)*
Changing the Layout and Styles

This section contains topics that explain how to customize the output using CSS, inserting HTML fragments, changing the layout of the main page, and more.

How to Use CSS Styling to Customize the Output

The most common way to customize WebHelp Responsive output is to use custom CSS styling. This method can be used to make small, simple styling changes or more advanced, precise changes. To implement the styling in your WebHelp output, you simply need to create the custom CSS file and reference it in your transformation scenario (using an Oxygen Publishing Template (on page 3221) or a transformation parameter). This custom file will be the final CSS to be applied so its content will override the styles in the other pre-existing CSS files.

Using CSS Inspector to Identify Content for Custom CSS File

You can use your browser’s CSS inspector to identify the pertinent code in the current CSS files and you can even make changes directly in the CSS inspector to test the results so that you know exactly what content to use in your custom CSS file.

In most popular browsers (such as Chrome, Firefox, and Edge), you can access the CSS inspector by using F12 or by selecting Inspect Element (or simply Inspect) from the contextual menu.

Tip:
When using Safari on macOS, you must first enable the Develop menu by going to the Advanced settings and selecting Show Develop menu in menu bar. Then you can select Show Web Inspector from the Develop menu or click Command + Option + I.

Create the Custom CSS

As a practical example, the following procedure would change the background color of the footer bar in the WebHelp output:

1. Use the browser’s CSS inspector to identify the current CSS code that styles the footer bar. In this particular case, the pertinent code that would be identified is:

   ```css
   .wh_footer {
       font-size: 15px;
       line-height: 1.7em;
       background-color: #000;
   }
   ```

2. If you want to test the color you want to apply as the background of this particular element, use the browser’s CSS inspector to change the value of the `background-color` attribute. After you find a suitable color, copy that new code.

3. Create a custom CSS file and paste or enter the copied code. For example:
4. Save the custom CSS file at a location of your convenience.
5. Reference the CSS file in a WebHelp Responsive transformation using an Oxygen Publishing Template (on page 1653) or the args.css parameter (on page 1653).

Referencing the CSS Using a Publishing Template

1. If you have not already created a Publishing Template, see How to Create a Publishing Template (on page 1791).
2. Using the Project view, copy your custom CSS in a folder inside the publishing template root folder (for example, in the custom_footer_template/resources folder).
3. Open the template descriptor file (on page 1607) associated with your publishing template and add your custom CSS in the resources section.

```xml
<publishing-template>
  ...
  <webhelp>
    ...
      <resources>
        ...
      </resources>
      <css file="resources/MyCustom.css"/>
  </webhelp>
  ...
</publishing-template>
```

4. Open the DITA Map WebHelp Responsive transformation scenario.
5. Click the Choose Custom Publishing Template link and select your template.
6. Click OK to save the changes to the transformation scenario.
7. Run the transformation scenario.

Result: Your custom CSS will be applied as a final layer on top of any existing CSS rules and the output will reflect the changes you made.

Referencing the CSS Using the args.css Parameter

1. Edit the DITA Map WebHelp Responsive transformation scenario and open the Parameters tab.
2. Set the args.css parameter to the path of your custom CSS file.
3. Set the args.copycss parameter to yes to automatically copy your custom CSS in the output folder when the transformation scenario is processed.
4. Click OK to save the changes to the transformation scenario.
5. Run the transformation scenario.

Result: Your custom CSS will be applied as a final layer on top of any existing CSS rules and the output will reflect the changes you made.
How to Insert Custom HTML Content

You can add custom HTML content in the WebHelp Responsive output by inserting it in a well-formed XML file (or specifying it in a well-formed XHTML fragment) that will be referenced in the transformation (either from an Oxygen Publishing Template or using one of the HTML fragment placeholder parameters). This content may include references to additional JavaScript, CSS, and other types of resources, or such resources can be inserted inline within the HTML content that is inserted in the XML file.

The XML File

There are several things to consider regarding this XML file:

- **Well-Formedness** - If the content of the file is not XML Well-formed, the transformation will automatically convert non-well-formed HTML content to a well-formed XML equivalent (assuming the webhelp.enable.html.fragments.cleanup transformation parameter is set to true).

  For example, if the HTML content includes several `<script>` or `<link>` elements, the XML fragment would have multiple root elements and to make it well-formed, it would be wrapped in an `<html>` element. This element tag will be filtered out and only its children will be copied to the output documents.

  Similarly, you can wrap your content in `<head>`, `<body>`, `<html/head>`, or `<html/body>` elements.

  **Note:**

  The converted fragments are stored in a file located in the whr-html-fragments subfolder of the transformation's temporary directory.

  **Tip:**

  If you do not want the transformation to automatically convert non-well-formed content into well-formed XML content, you can set the webhelp.enable.html.fragments.cleanup transformation parameter to false. This will instead cause the transformation to fail if at least one HTML fragment is not well-formed.

- **Referencing Resources in the XML File** - You can include references to local resources (such as JavaScript or CSS files) by using the built-in `${oxygen-webhelp-output-dir}` macro to specify their paths relative to the output directory:

  ```html
  <html>
  <script type="text/javascript" src="${oxygen-webhelp-output-dir}/js/test.js"/>
  <link rel="stylesheet" type="text/css"
       href="${oxygen-webhelp-output-dir}/css/test.css"/>
  </html>
  ```

  If you want that the path of your resource to be relative to the templates directory, you can use the `${oxygen-webhelp-template-dir}` macro.
To copy the referenced resources to the output directory, follow the procedure in: How to Copy Additional Resources to Output Directory (on page 1704).

• Inline JavaScript or CSS Content:

JavaScript:

```html
<script type="text/javascript">
/* Include JavaScript code here. */

function myFunction() {
    return true;
}
</script>
```

CSS:

```html
<style>
/* Include CSS style rules here. */

*{
    color:red
}
</style>
```

**Note:**

If you have special characters (e.g. &amp;, &lt;) that break the well-formedness of the XML fragment, it is important to place the content inside an XML comment.

Otherwise, the WebHelp transformation automatically wraps inline JavaScript or CSS content in an XML comment. Also, if the commented content contains constructs that are not allowed in an XML comment, those constructs are escaped.

**[Important]** XML comment tags (both the start and end tags) must be on lines by themselves. If they are on the same line as any of the script's content, it will likely result in a JavaScript error.

```html
<script type="text/javascript">
    <!--
    /* Include JavaScript code here. */
    
    function myFunction() {
        return true;
    }
</script>
```
Using WebHelp Macros

The XML file can use WebHelp macros, which are variables that will be expanded when the content of the HTML fragment file will be copied in the final output.

There are two possibilities for using macros:

- **Directly in attribute values** - For example, if you want to reference a JavaScript file from the Publishing Template directory, you can use the following construct:

  ```html
  <script type="text/javascript" src="${path(oxygen-webhelp-template-dir)}/"></script>
  ```

- **In text content** - Using the `<whc:macro>` template component:

  ```html
  <script type="text/javascript">
    var outDirPath = '<whc:macro value="${path(oxygen-webhelp-output-dir)}" xmlns:whc="http://www.oxygenxml.com/webhelp/components" />';
    console.log("The output directory path is:", outDirPath);
  </script>
  ```

**Note:**

When using the `<whc:macro>` element, you should also include the `xmlns:whc="http://www.oxygenxml.com/webhelp/components"` namespace declaration for the `whc` prefix. This is necessary for the XML fragment to be well-formed.

The following *macros* are supported:

**i18n**

For localizing a string.

```html
${i18n(string.id)}
```

**param**

Returns the value of a transformation parameter.

```html
${param(webhelp.show.main.page.tiles)}
```

**env**

Returns the value of an environment variable.

```html
${env(JAVA_HOME)}
```

**system-property**

Returns the value of a system property.
timestamp

Can be used to format the current date and time. Accepts a string (as a parameter) that determines how the date and time will be formatted (format string or picture string as it is known in the XSLT specification). The format string must comply with the rules of the XSLT formatDateTime function specification.

${timestamp([h1]:[m0]:[s0]:[P]/[M]/[D]/[Y000])}

path

Returns the path associated with the specified path ID. The following paths IDs are supported:

- **oxygen-webhelp-output-dir** - The path to the output directory. The path is relative to the current HTML file.
- **oxygen-webhelp-assets-dir** - The path to the oxygen-webhelp subdirectory from the output directory. The path is relative to the current HTML file.
- **oxygen-webhelp-template-dir** - The path to the template directory. The path is relative to the current HTML file.

${path(oxygen-webhelp-template-dir)}

Note:

New paths IDs can be added by overriding the wh-macro-custom-path template from com.oxygenxml.webhelp.responsive\xsl\template\macroExpander.xsl:

<!-- Extension template for expanding a custom path macro. -->
<xsl:template name="wh-macro-custom-path">
  <xsl:param name="pathId" />
  <xsl:value-of select="$pathId" />
</xsl:template>

map-xpath

Can be used to execute an XPath expression over the DITA map file from the temporary directory.

Tip:

Available in all template layout HTML pages.

${map-xpath(/map/title)}

topic-xpath

Can be used to execute an XPath expression over the current topic.
Tip:
Available only in the topic HTML page template (wt_topic.html).

```xml
${topic-xpath(string-join(//shortdesc//text(), ' '))}
```

**oxygen-webhelp-build-number**

Returns the current WebHelp distribution ID (build number).

```xml
${oxygen-webhelp-build-number}
```

Referencing the HTML fragment using a Publishing Template

1. If you have not already created a Publishing Template, see Working with Publishing Templates (on page 1642).
2. Insert the HTML content in a file that is XML well-formed (for example, custom-html.xml).
3. Using the **Project** view, copy your custom XML file in a folder inside publishing the template root folder (for example, in the custom_footer_template/html-fragments folder).
4. Open the template descriptor file (on page 1607) associated with your publishing template and add a reference to the custom HTML fragment in the **html-fragments** section.

```xml
<publishing-template>
  ...
  <webhelp>
    ...
    <html-fragments>
      <fragment
        file="html-fragments/custom-html.xml"
        placeholder="webhelp.fragment.head"/>
    </html-fragments>
  </webhelp>
  ...
</publishing-template>
```

**Note:**
If you want to insert the content in another location within the output document, you can reference the XML file from any other HTML Fragment extension points (on page 1613).

5. Open the **DITA Map WebHelp Responsive** transformation scenario.
6. Click the **Choose Custom Publishing Template** link and select your template.
7. Click **OK** to save the changes to the transformation scenario.
8. Run the transformation scenario.

**Results:** Your additional content will be included at the end of the `<head>` element of your output document.

Referencing the HTML Fragment using a Transformation Parameter

1. Insert the HTML content in a well-formed XML file.
2. Edit the **DITA Map WebHelp Responsive** transformation scenario and open the **Parameters** tab.
3. Edit the value of the `webhelp.fragment.head` parameter and set it to the absolute path of your XML file.

![Note:
If you want to insert the content in another location within the output document, you can reference the XML file from any other HTML Fragment extension points (on page 1613).

4. Click **OK** to save the changes to the transformation scenario.

5. Run the transformation scenario.

**Results:** Your additional content will be included at the end of the `<head>` element of your output document.

**Related Information:**
- HTML Fragment Placeholders (on page 1613)
- Publishing Template Package Contents for WebHelp Responsive Customizations (on page 1606)

**How to Insert JavaScript AMD Modules**

In the WebHelp Responsive output, you may want to include a JavaScript module that uses the Asynchronous Module Definition (AMD) format. Unlike the traditional JavaScript resources that are loaded using `<script>` tags, these modules are loaded using the **RequireJS** library. For the traditional JavaScript libraries, the standard procedure to contribute your script to the output would be to use an HTML fragment file as described here (on page 1654).

Although following the procedure that uses HTML fragments (on page 1654) would result in having your JS file included in the output, loading the JS code in the browser will result in an error. Since your JS module uses the **AMD API**, it cannot be loaded using a `<script>` element. For example, because many jQuery plugins use the AMD format, it will be difficult for you to use those libraries in your custom WebHelp output.

Normally, a JavaScript AMD module can be loaded in one of the following ways:

- **As a top-level script**, using the `data-main` attribute of the `<script>` element used to load the **RequireJS** library.

  ```html
  <script data-main="js/template-main.js" src="js/require.js"></script>
  ``

  However, since WebHelp already loads its internal JS modules using **RequireJS**, a top-level script is already specified and you cannot specify another top-level script in the same page. Therefore, this approach cannot be used to load your custom JS module in Oxygen XML WebHelp, leaving you with only the following option.

- **As a dependency module**, using a `require()` call from the top-level (main) script or from one of its dependency modules. This means that you would have to alter one of the WebHelp core JS libraries and inject a `require()` call to load your custom module:
Contributing JavaScript AMD Modules Using a Publishing Template

Oxygen XML WebHelp provides the ability to contribute a custom JavaScript Asynchronous Module Definition (AMD) resource in the output by referencing it in the Publishing Template descriptor file (on page 1607). This module is automatically copied to the output directory and it is automatically loaded in the output HTML pages using a require() call. Thus, you can include your scripts in the output without altering WebHelp's core JavaScript libraries.

This module may contain all of your custom functionality or can be used to load other AMD JavaScript modules. The additional sub-modules can be stored either locally in your custom Publishing Template or on a remote web server.

Important:
To enable loading of a JS module you should set the webhelp.enable.template.js.module.loading parameter to yes (the default value is no) in the Publishing Template descriptor file or in the transformation scenario.

The JavaScript Modules

The JS Modules sample template contains a main JavaScript module (template-main.js) that loads other modules stored in the template package or in a remote location on the Internet.

```javascript
define(['require'], function (require) {
  require(['./red', './italic', './tables']);
});
```

Besides the main JavaScript example, the template contains 3 other sub-modules:

- **red.js** - Changes the font color of the publication title.

```javascript
define(['jquery'], function ($) {
  $(document).ready(function () {
    // Make the title red
    $('a').attr('style', 'color:red');
  });
});
```

- **italic.js** - Changes the font style of your publication title.
```javascript
define(['jquery'], function ($) {
    $(document).ready(function () {
        // Make the title italic
        $('>.wh_publication_title a').wrapInner('<i></i>');
    });
});
```

- **tables.js** - Loads the DataTables jQuery plugin from a CDN.

```javascript
define(['jquery',
        'https://cdn.datatables.net/1.10.16/js/jquery.dataTables.min.js'], function ($) {
    $(document).ready(function () {
        $('.table').DataTable();
    });
});
```

The JavaScript modules are stored in the Publishing Template package as follows:

```
[template-dir]
[js]
    template-main.js
    italic.js
    red.js
    tables.js
```

---

**Notes:**

- The main module should be referenced in the Publishing Template descriptor file *(on page 1607)* by specifying its path relative to the base directory of the template.

```xml
<js-amd-module file="js/template-main.js"/>
```

- The main JS module is copied automatically to the output directory, but the sub-modules are not. To instruct the Publishing Template engine to copy those modules to the output directory you should include a `<fileset>` section in the Publishing Template descriptor file *(on page 1607).*

```xml
<fileset>
    <include name="js/*.js"/>
</fileset>
```

- The main module can reference other modules bundled in the publishing template package as follows:
Relative to template directory - Prefix their paths with the ID of the template directory:
`template-base-dir` (for example: `template-base-dir/js/italic`).

Relative to the name of the current JS module - Use `.\` to prefix the paths of the referenced modules.

The `.js` extension should be omitted for local modules, because the RequireJS library will add it automatically.

Related Information:
How to Insert Custom HTML Content (on page 1654)

How to Change Numbering Styles for Ordered Lists

Ordered lists ( `<ol>` ) are usually numbered in XHTML output using numerals. If you want to change the numbering to alphabetical, follow these steps:

1. Define a custom `@outputclass` value and set it as an attribute of the ordered list, as in the following example:

```xml
<ol outputclass="number-alpha">
  <li>A</li>
  <li>B</li>
  <li>C</li>
</ol>
```

2. Add the following code snippet in a custom CSS file:

```css
ol.number-alpha{
  list-style-type:lower-alpha;
}
```

3. Reference the CSS file in a WebHelp Responsive transformation using an Oxygen Publishing Template (on page 1662) or the `args.css` parameter (on page 1663).

Referencing the Custom CSS from a Publishing Template

1. If you have not already created a Publishing Template, see How to Create a Publishing Template (on page 1791).
2. Using the Project view, copy your custom CSS in a folder inside the publishing template root folder (for example, in the `custom_footer_template/resources` folder).
3. Open the `template descriptor file` (on page 1607) associated with your publishing template and add your custom CSS in the `resources` section.

```xml
<publishing-template>
  ...
  <webhelp>
```
4. Open the DITA Map WebHelp Responsive transformation scenario.
5. Click the Choose Custom Publishing Template link and select your template.
6. Click OK to save the changes to the transformation scenario.
7. Run the transformation scenario.

Result: Your custom CSS will be applied as a final layer on top of any existing CSS rules and the output will reflect the changes you made.

Referencing the CSS Using the args.css Parameter

1. Edit the DITA Map WebHelp Responsive transformation scenario and open the Parameters tab.
2. Set the args.css parameter to the path of your custom CSS file.
3. Set the args.copycss parameter to yes to automatically copy your custom CSS in the output folder when the transformation scenario is processed.
4. Click OK to save the changes to the transformation scenario.
5. Run the transformation scenario.

Result: Your custom CSS will be applied as a final layer on top of any existing CSS rules and the output will reflect the changes you made.

Adding Syntax Highlights for Codeblocks in the Output

Syntax Highlighting makes it easier to read the semantics of the structured content by displaying each type of code (language) in different colors and fonts. The application provides the ability to add syntax highlights in codeblocks for DITA to PDF or HTML-based output through the use of the \@outputclass attribute and a variety of predefined values are available.

To provide syntax highlighting in the codeblocks that appear in the output, add the \@outputclass attribute on the \<codeblock> element and set its value to one of the predefined language values. The Content Completion Assistant offers a list of the possible values when adding the \@outputclass attribute in Text mode but there are also two very simple ways to set the value in Author mode:

- Select the \<codeblock> element in the editor and in the Attributes view, click on the Value cell for the \@outputclass attribute and select one of the predefined values (for example, language-xml).
- Select the \<codeblock> element in the editor and use the Alt + Enter keyboard shortcut to open the in-place attributes editor window. Then select one of the predefined values from the Value drop-down menu.

The predefined values that can be selected are:
Attention:
It is recommended that you do not add inline elements in the codeblocks when using this `@outputclass` attribute, as it may lead to improper highlighting.

Tip:
Starting with version 24.0, the language values can also be set without using the `language-` prefix.

Example:
This codeblock:

```xml
<codeblock outputclass="language-xml">
  <p>This code is an example of how to use a coderef.</p>
  <coderef href="MyExternalCode.xsl"/></codeblock>
</codeblock>
```

would result in this in the output:

```xml
<p>This code is an example of how to use a coderef.</p>
<codeblock><coderef href="MyExternalCode.xsl"/></codeblock>
```
How to Show or Hide Navigation Links in Topic Pages

The topic pages (on page 1565) in WebHelp Responsive output can contain navigation links (Previous / Next arrows) that can be used to navigate to the previous or next topic.

How to Control Which Topic Pages Include Navigation Links

The navigation links are controlled by the @collection-type attribute. For example, if you set collection-type="sequence" on a parent topic reference in your DITA map, navigation links will be generated in the output for all of its child topics (from children to parent, and from child to previous sibling and next sibling).

```xml
<map id="example_map" title="Example Map">
  <topicref href="../topics/ParentTopic.dita" collection-type="sequence">
    <topicref href="/topics/Childtopic.dita"/>
  </topicref>
</map>
```

How to Generate Navigation Links for All Topics (Ignoring the Collection Type Attribute)

You can use the webhelp.default.collection.type.sequence parameter in the transformation and set its value to yes to generate navigation links for all topics, regardless of whether or not the collection-type attribute is present.

How to Hide All Navigation Links

To hide all navigation links, use the webhelp.show.navigation.links parameter in the transformation and set its value to no.

How to Change the Main Page Layout

This section contains topics that explain how to customize the layout of the main page in the WebHelp Responsive output.

How to Customize the Menu

By default, the menu component is displayed in all WebHelp Responsive pages. However, you might want to hide it completely, or only display some of its menu entries.

How to Hide Some of the Menu Entries

There are two methods for doing this. One of them involves editing the DITA map (on page 3218) and marking the topics that do not need to be included in the menu, and another one that uses a small CSS customization.

Editing the DITA Map

To edit the metadata in the DITA map to control which topics will not be displayed in the menu, follow these steps:
1. Open the DITA map in the Text editing mode of Oxygen XML Editor.
2. Add the following metadata information in the topicref element (or any of its specializations) for each topic you do not want to be displayed in the menu:

   ```xml
   <topicmeta>
     <data name="wh-menu">
       <data name="hide" value="yes"/>
     </data>
   </topicmeta>
   ```

**Customizing the CSS**

To customize the CSS to control which topics will not be displayed in the menu, follow these steps:

1. Make sure you set an ID on the topic that you do not want to include in the menu.
2. Create a new CSS file that contains a rule that hides the menu entry generated for the topic (identified by the topic ID growing-flowers in the following example). The CSS file should have content that is similar to this:

   ```css
   .wh_top_menu *[data-id='growing-flowers'] { 
     display:none;
   }
   ```

3. Reference the CSS file in a WebHelp Responsive transformation using an Oxygen Publishing Template (on page 1653) or the args.css parameter (on page 1653).

**How to Hide the Entire Menu**

If you do not want to include a main menu in the pages of the WebHelp Responsive output, you can instruct the transformation scenario to skip the menu generation completely.

**Using a Publishing Template**

To hide the menu using an Oxygen Publishing Template (on page 1603), follow this procedure:

1. If you have not already created a Publishing Template, see How to Create a Publishing Template (on page 1791).
2. Open the template descriptor file (on page 1607) associated with your publishing template and add the webhelp.show.top.menu parameter in the parameters section with its value set to no.
3. Open the *DITA Map WebHelp Responsive* transformation scenario.
4. Click the **Choose Custom Publishing Template** link and select your template.
5. Click **OK** to save the changes to the transformation scenario.
6. Run the transformation scenario.

**Using a Transformation Scenario in Oxygen XML Editor/Author**

To hide the menu using a transformation scenario from within *Oxygen XML Editor/Author*, follow this procedure:

1. Edit the *DITA Map WebHelp Responsive* transformation scenario and choose a template.
2. Open the **Parameters** tab and set the `webhelp.show.top.menu` parameter to **no**.
3. Click **OK** to save the changes to the transformation scenario.
4. Run the transformation scenario.

**How to Add a Welcome Message in the WebHelp Responsive Main Page**

The main page of the WebHelp Responsive output contains a set of *empty placeholders* (*on page 1613*) that can be used to display customized text fragments. These placeholders are available to you through WebHelp Responsive transformation scenario parameters. For example, the placeholder identified through the `webhelp.fragment.welcome` parameter displays text content above the search box in the main page.

**Using a Publishing Template**

To add a customized welcome message in the main page of the WebHelp Responsive output using an *Oxygen Publishing Template* (*on page 1603*), follow this procedure:

1. If you have not already created a Publishing Template, see How to Create a Publishing Template (*on page 1791*).
2. Open the **template descriptor file** (*on page 1607*) associated with your publishing template and add the `webhelp.fragment.welcome` parameter in the **parameters** section with its value set to one of the following:
   - A small well-formed XHTML fragment (such as: `<i>Welcome to the User Guide</i>`).
   - A path to a file that contains well-formed XHTML content.

```xml
<publishing-template>
    ...
    <webhelp>
        ...
        <parameters>
            <parameter name="webhelp.fragment.welcome" value="c:\myMessage.xhtml"/>
        </parameters>
    </webhelp>
</publishing-template>
```

3. Open the *DITA Map WebHelp Responsive* transformation scenario.
4. Click the **Choose Custom Publishing Template** link and select your template.
5. Click OK to save the changes to the transformation scenario.  
6. Run the transformation scenario.

**Result:** In the WebHelp output, your custom message will be displayed above the search box in the main page.

### Using a Transformation Scenario in Oxygen XML Editor/Author

**Important:** Running WebHelp transformations from a script outside of Oxygen XML Editor/Author requires an additional license and some additional setup:

- You must have a valid license for the Oxygen XML WebHelp Plugin (https://www.oxygenxml.com/buy_webhelp.html).
- The Oxygen XML WebHelp Plugin must be installed and integrated.

To add a customized welcome message in the main page of the WebHelp Responsive output using a transformation scenario from within Oxygen XML Editor/Author, follow this procedure:

1. Edit the DITA Map WebHelp Responsive transformation scenario and choose a template.  
2. Open the Parameters tab and set the webhelp.fragment.welcome parameter with its value set to one of the following:
   - A small well-formed XHTML fragment (such as `<i>Welcome to the User Guide</i>`).
   - A path to a file that contains well-formed XHTML content.
3. Click OK to save the changes to the transformation scenario.
4. Run the transformation scenario.

**Result:** In the WebHelp output, your custom message will be displayed above the search box in the main page.

### How to Create a Custom Footer

The main page of the WebHelp Responsive output contains a set of empty placeholders (on page 1613) that can be used to display customized text fragments. These placeholders are available to you through WebHelp Responsive transformation scenario parameters. For example, the placeholder identified through the `webhelp.fragment.footer` parameter displays the custom content at the bottom of the page.

### Using a Publishing Template

To create a custom footer in the WebHelp Responsive output using an Oxygen Publishing Template (on page 1603), follow this procedure:

1. If you have not already created a Publishing Template, see How to Create a Publishing Template (on page 1791).
2. Open the template descriptor file (on page 1607) associated with your publishing template and add the `webhelp.fragment.footer` parameter in the parameters section with its value set to one of the following:
A small well-formed XHTML fragment.
- A path to a file that contains well-formed XHTML content.

```xml
<publishing-template>
  ...
  <webhelp>
    ...
    <parameters>
      <parameter name="webhelp.fragment.footer" value="c:\myFooter.xhtml"/>
    </parameters>
  </webhelp>
</publishing-template>
```

**Important:**
This parameter should only be used if you are using a valid, purchased license of Oxygen XML Editor (do not use it with a trial license).

3. Open the *DITA Map WebHelp Responsive* transformation scenario.
4. Click the *Choose Custom Publishing Template* link and select your template.
5. Click **OK** to save the changes to the transformation scenario.
6. Run the transformation scenario.

**Result:** In the WebHelp output, your custom footer will be displayed at the bottom of the page.

### Using a Transformation Scenario in Oxygen XML Editor/Author

**Important:**
Running WebHelp transformations from a script outside of *Oxygen XML Editor/Author* requires an additional license and some additional setup:

- You must have a valid license for the *Oxygen XML WebHelp Plugin* ([https://www.oxygenxml.com/buy_webhelp.html](https://www.oxygenxml.com/buy_webhelp.html)).
- The *Oxygen XML WebHelp Plugin* must be installed and integrated.

To create a custom footer in the WebHelp Responsive output using a transformation scenario from within *Oxygen XML Editor/Author*, follow this procedure:

1. Edit the *DITA Map WebHelp Responsive* transformation scenario and choose a *template*.
2. Open the *Parameters* tab and set the *webhelp.fragment.footer* parameter with its value set to one of the following:
   - A small well-formed XHTML fragment.
   - A path to a file that contains well-formed XHTML content.
3. Click **OK** to save the changes to the transformation scenario.
4. Run the transformation scenario.
Result: In the WebHelp output, your custom footer will be displayed at the bottom of the page.

How to Configure the Tiles on the WebHelp Responsive Main Page

The *tiles* version of the main page of the WebHelp Responsive output displays a tile for each topic found on the first level of the *DITA map* (on page 3218). However, you might want to customize the way they look or even to hide some of them.

Depending on your particular setup, you can choose to customize the tiles either by setting metadata information in the *DITA map* or by customizing the CSS that is associated with the *DITA map*.

How to Hide Some of the Tiles

If your documentation is very large or there is a large number of topics on the first level, you might want to hide some of the tiles. Also, this might be useful if you only want to display the topics in the first page that are most relevant to your intended audience.

There are two methods for doing this. One of them involves editing the *DITA map* and marking the topics that do not need to be displayed as tiles, and another one that uses a small CSS customization level to hide some tiles identified by the ID of the topic.

**Editing the DITA Map**

To edit the metadata in the *DITA map* to control which topics on the first level of the *DITA map* will not be displayed as a tile, follow these steps:

1. Open the *DITA map* in the **Text** editing mode of Oxygen XML Editor.
2. Add the following metadata information in the `<topicref>` element (or any of its specializations) for each first-level topic that you do not want to be displayed as a tile:

```xml
<topicmeta>
  <data name="wh-tile">
    <data name="hide" value="yes"/>
  </data>
</topicmeta>
```

**Customizing the CSS**

To customize the CSS to control which topics on the first level of the *DITA map* will not be displayed as a tile, follow these steps:

1. Make sure you set an ID on the topic you want to hide.
2. Create a new CSS file that contains a rule that hides the tile generated for the topic (identified in the following example by the topic ID `growing-flowers`). The CSS file should have content that is similar to this:
3. Reference the CSS file in a WebHelp Responsive transformation using an Oxygen Publishing Template (on page 1653) or the args.css parameter (on page 1653).

How to Add an Image to the Tiles

There are two methods that you can use to add an image to a tile. One of them involves editing the DITA map, and the other uses a CSS customization.

Editing the DITA Map

To edit the metadata in the DITA map to set an image to be displayed in a tile, follow these steps:

1. Open the DITA map in the Text editing mode of Oxygen XML Editor.
2. Add the following metadata information in the <topicref> element (or any of its specializations) for each first-level topic that will have an image displayed in the corresponding tile:

   ```xml
   <topicmeta>
     <data name="wh-tile">
       <data name="image" href="img/tile-image.png" format="png">
         <data name="attr-width" value="64"/>
         <data name="attr-height" value="64"/>
       </data>
     </data>
   </topicmeta>
   ```

   Note: The @attr-width and @attr-height attributes can be used to control the size of the image, but they are optional.

Customizing the CSS

To customize the CSS to set an image to be displayed in a tile, follow these steps:

1. Make sure you set an ID on the topic that you want the tile to include an image.
2. Create a new CSS file that contains a rule that associates an image with a specific tile. The CSS file should have content that is similar to this:

   ```css
   .wh_tile [data-id='growing-flowers'] { 
     display:none;
   }
   ```

3. Reference the CSS file in a WebHelp Responsive transformation using an Oxygen Publishing Template (on page 1653) or the args.css parameter (on page 1653).
Adding Graphics and Media Resources

This section contains topics that explain how to add media resources to the published output or the output directory.

How to Add a Logo Image in the Title Area

You can customize WebHelp Responsive output to include a logo in the title area. It will be displayed before the publication title. You can also specify a URL that can be used to send users to a specific website when they click the logo image.

This customization can be done using an Oxygen Publishing Template or using a transformation scenario from within Oxygen XML Editor/Author.

Using a Publishing Template

To add a logo in the title area of your WebHelp output using an Oxygen Publishing Template (on page 1603), follow this procedure:

1. If you have not already created a Publishing Template, see How to Create a Publishing Template (on page 1791).
2. Open the template descriptor file (on page 1607) associated with your publishing template and add the `<logo>` element in the `<resources>` section and set the `@file` attribute value to the path of your logo.
3. If you also want to add a link to your website when you click the logo image, set its URL in the `@target-url` attribute.

```xml
<publishing-template>
  ...
  <webhelp>
    ...
    <resources>
      <logo
          file="images/logo.png"
          target-url="http://www.example.com"
          alt="Alternate text for the logo image"/>
    </resources>
  </webhelp>
</publishing-template>
```
4. Open the DITA Map WebHelp Responsive transformation scenario.
5. Click the Choose Custom Publishing Template link and select your template.
6. Click OK to save the changes to the transformation scenario.
7. Run the transformation scenario.

Using a Transformation Scenario in Oxygen XML Editor/Author

To add a logo in the title area of your WebHelp output using a transformation scenario from within Oxygen XML Editor/Author, follow this procedure:
1. Edit the *DITA Map WebHelp Responsive* transformation scenario and choose a template.
2. Open the **Parameters** tab and set the `webhelp.logo.image` parameter to the path of your logo.
3. If you also want to add a link to your website when you click the logo image, set its URL in the `webhelp.logo.image.target.url` parameter.
4. Click OK to save the changes to the transformation scenario.
5. Run the transformation scenario.

**How to Add a Favicon in WebHelp Systems**

You can add a custom *favicon* to your WebHelp output by simply using a parameter in the transformation scenario to point to your *favicon* image.

This customization can be done using an *Oxygen Publishing Template* or using a transformation scenario from within *Oxygen XML Editor/Author*.

**Using a Publishing Template**

To add a *favicon* to your WebHelp output using an *Oxygen Publishing Template (on page 1603)*, follow this procedure:

1. If you have not already created a Publishing Template, see *Working with Publishing Templates (on page 1642)*.
2. Open the *template descriptor file (on page 1607)* associated with your publishing template and add the `<favicon>` element in the *resources* section. The path to the image is relative to the template root folder.
   ```xml
   <publishing-template>
     ...
     <webhelp>
       ...
       <resources>
         ...
         <favicon file="images/favicon.png"/>
       </resources>
     </webhelp>
     ...
   </publishing-template>
   ```
3. Open the *DITA Map WebHelp Responsive* transformation scenario.
4. Click the **Choose Custom Publishing Template** link and select your template.
5. Click OK to save the changes to the transformation scenario.
6. Run the transformation scenario.

**Result:** Browsers that provide *favicon* support display the *favicon* (typically in the browser's address bar, in the list of bookmarks, and in the history).

**Using a Transformation Scenario in Oxygen XML Editor/Author**

To add a *favicon* to your WebHelp output using a transformation scenario from within *Oxygen XML Editor/Author*, follow this procedure:
1. Edit the DITA Map WebHelp Responsive transformation scenario and choose a template.
2. Open the Parameters tab and set the webhelp.favicon parameter to the path of your image.
3. Click OK to save the changes to the transformation scenario.
4. Run the transformation scenario.

How to Add Video and Audio Objects in DITA WebHelp Output

You can insert references to video and audio media resources (such as videos, audio clips, or embedded HTML frames) in your DITA topics and then publish them to WebHelp output. The media objects can be played directly in all HTML5-based outputs, including WebHelp systems.

To add media objects in the WebHelp output generated from DITA documents, follow the procedures below.

**Adding Videos to DITA WebHelp Output**

1. Edit the DITA topic and insert a reference to the video through one of the following methods:
   - Use the Insert Media Object toolbar action (on page 2977).
   - Drag (or copy) the video file from your system explorer or the Project view (on page 402) and drop (or paste) it into your document.
   - Manually add an `<object>` element, as in one of the following examples:

   ```xml
   <object outputclass="video" type="video/mp4" data="MyVideo.mp4"/>
   ``

   or, instead of the `@data` attribute, you can specify the video using a parameter like this:

   ```xml
   <object outputclass="video">
     <param name="src" value="videos/MyVideo.mp4"/>
   </object>
   ```

2. Apply a DITA to WebHelp transformation to obtain the output.

**Result:** The transformation converts the `<object>` element to an HTML5 `<video>` element.

```xml
<video controls="controls">
  <source type="video/mp4" src="MyVideo.mp4"></source>
</video>
```

**Adding Audio Clips to DITA WebHelp Output**

1. Edit the DITA topic and insert a reference to the audio clip through one of the following methods:
   - Use the Insert Media Object toolbar action (on page 2977).
   - Drag (or copy) the audio file from your system explorer or the Project view (on page 402) and drop (or paste) it into your document.
   - Manually add an `<object>` element, as in one of the following examples:

   ```xml
   <object outputclass="audio" type="audio/mpeg" data="MyClip.mp3"/>
   ``

   or, instead of the `@data` attribute, you can specify the video using a parameter like this:
2. Apply a **DITA to WebHelp** transformation to obtain the output.

**Result:** The transformation converts the `<object>` element to an HTML5 `<audio>` element.

```xml
<object outputclass="audio">
  <param name="src" value="audio/MyClip.mp3"/>
</object>
```

---

### Adding Embedded HTML Frames (such as YouTube videos) to DITA WebHelp Output

1. Edit the DITA topic and insert a reference to the embedded object by using the `Insert Media Object` toolbar action (on page 2977) or by manually adding an `<object>` element, as in one of the following examples:

```xml
<object outputclass="iframe" data="https://www.youtube.com/embed/m_vv2s5Trn4"/>
```

or, instead of the `@data` attribute, you can specify the object using a parameter like this:

```xml
<object outputclass="iframe">
  <param name="src" value="http://www.youtube.com/embed/m_vv2s5Trn4"/>
</object>
```

2. If you want the video to be allowed to play in full screen mode once the document is converted to XHTML output, also add an `allowfullscreen` parameter and set its value to `true`:

```xml
<object outputclass="iframe" data="https://www.youtube.com/embed/m_vv2s5Trn4"/>
  <param name="allowfullscreen" value="true"/>
</object>
```

**Tip:**

If you copy the embed code from the source and paste it into the `Insert Media` dialog box (see the specific instructions: [here](on page 2979)), the `allowfullscreen` parameter will automatically be added and all you have to do is set the value to `true`.

3. Apply a **DITA to WebHelp** transformation to obtain the output.

**Result:** The transformation converts the `<object>` element to an HTML5 `<iframe>` element.

```xml
<iframe controls="controls" src="https://www.youtube.com/embed/m_vv2s5Trn4"/>
```

For more information, see the following video demonstration:

https://www.youtube.com/embed/lIX11gS4WaU
How to Add MathML Equations in WebHelp Output

Currently, only Firefox can render MathML equations embedded in the HTML code. MathJax is a solution to properly view MathML equations embedded in HTML content in a variety of browsers.

If you have DocBook content that has embedded MathML equations and you want to properly view the equations in published HTML output types (such as WebHelp), you need to add a reference to the MathJax script in the head element of all HTML files that have the equation embedded.

For example:

```html
<script type="text/javascript" src="https://cdnjs.cloudflare.com/ajax/libs/mathjax/2.7.1/MathJax.js?config=TeX-AMS-MML_HTMLorMM L"></script>
```

Result: The equation should now be properly rendered in the WebHelp output for other browsers.

Searching the Output

This section contains topics that explain how to use some of the search features in WebHelp Responsive output.

How to Change Element Scoring in Search Results

The WebHelp Search feature is enhanced with a rating mechanism that computes scores for every page that matches the search criteria. HTML tag elements are assigned a scoring value and these values are evaluated for the search results. The WebHelp directory includes a properties file that defines the scoring values for tag elements and this file can be edited to customize the values according to your needs.

To edit the scoring values of HTML tag element for enhancing WebHelp search results, follow these steps:

1. Edit the scoring properties file for DITA. The properties file includes instructions and examples to help you with your customization. The file is located in: DITA-OT-DIR\plugins \com.oxygenxml.webhelp.responsive\indexer\scoring.properties. The values that can be edited in the scoring.properties file:

```plaintext
h1 = 10
h2 = 9
h3 = 8
h4 = 7
h5 = 6
```
2. Save your changes to the file.
3. Re-run your WebHelp transformation.

How to Exclude Certain DITA Topics from Search Results

There are several ways to exclude certain DITA resources from your WebHelp system's search results. This is useful if you have topics in your DITA map (on page 3218) structure that you do not want to be included in search results for your WebHelp system. The first method involves setting a parameter in the WebHelp transformation scenario and the second involves setting an attribute for each DITA topic reference that you want to exclude.

Transformation Parameter Method

To exclude DITA topics from WebHelp search results using a transformation parameter, follow these steps:

1. Create a simple text file that will contain your excluded file patterns. Each pattern must be on a new line. The patterns are considered to be relative to the output directory and they accept wildcards such as ‘*’ (matches zero or more characters) or ‘?’ (matches one character). For more information about the patterns, see https://ant.apache.org/manual/dirtasks.html#patterns.

   **Example:** Suppose that in your project, you want to exclude all files located in the resources directory and all files located in the topics directory that have a .bak file extension. You could create a simple text file (for example, named exclude.properties), and add the following lines:

   ```
   resources/*
   topics/*.bak
   ```

2. Set the webhelp.search.custom.excludes.file parameter to specify the path to the file that contains the excluded file patterns (for example, exclude.properties in step 1). The parameter can be specified in the parameters section of the template descriptor file (on page 1612) associated with your
publishing template or in the Parameters tab of the transformation scenario dialog box in Oxygen XML Editor/Author.

3. Run the transformation.

Search Attribute Method

The WebHelp Search engine does not index DITA topics that have the @search attribute set to no.

To exclude DITA topics from WebHelp search results using this attribute, follow these steps:

1. Edit the DITA map and for any <topicref> that you want to exclude from search results, set the @search attribute to no. For example:

   ```xml
   <topicref href="../topics/internal-topic1.dita" search="no"/>
   ```

2. Save your changes to the DITA map.

3. Run your WebHelp system transformation.

How to Optimize Search Results

A DITA Map WebHelp transformation scenario produces a sitemap.xml file that is used by search engines to aid crawling and indexing mechanisms. A sitemap lists all pages of a WebHelp system and allows web admins to provide additional information about each page, such as the date it was last updated, change frequency, and importance of each page in relation to other pages in your WebHelp deployment.

**Important:**

If the webhelp.sitemap.base.url parameter is specified, the loc element will contain the value of this parameter plus the relative path to the page. If the webhelp.sitemap.base.url parameter is not specified, the loc element will only contain the relative path of the page.

You can also set these additional parameters:

- **webhelp.sitemap.change.frequency** - Specifies how frequently the WebHelp pages are likely to change (accepted values are: always, hourly, daily, weekly, monthly, yearly, and never).
- **webhelp.sitemap.priority** - Specifies the priority of each page (a value ranging from 0.0 to 1.0).

The structure of the sitemap.xml file looks like this:

```xml
<urlset xmlns="http://www.sitemaps.org/schemas/sitemap/0.9">
  <url>
    <loc>http://www.example.com/topics/introduction.html</loc>
    <lastmod>2014-10-24</lastmod>
    <changefreq>weekly</changefreq>
    <priority>0.5</priority>
  </url>
</urlset>
```
Each page has a `<url>` element structure containing additional information, such as:

- **loc** - The URL of the page. This URL must begin with the protocol (such as `http`), if required by your web server. It is constructed from the value of the `webhelp.sitemap.base.url` parameter from the transformation scenario and the relative path to the page (collected from the `href` attribute of a `topicref` element in the DITA map).

  Note:
  The value must have fewer than 2,048 characters.

- **lastmod** (optional) - The date when the page was last modified. The date format is `YYYY-MM-DD hh:mm:ss`.

- **changefreq** (optional) - Indicates how frequently the page is likely to change. This value provides general information to assist search engines, but may not correlate exactly to how often they crawl the page. Valid values are: `always`, `hourly`, `daily`, `weekly`, `monthly`, `yearly`, and `never`. The first time the `sitemap.xml` file is generated, the value is set based upon the value of the `webhelp.sitemap.change.frequency` parameter in the DITA WebHelp transformation scenario. You can change the value in each `url` element by editing the `sitemap.xml` file.

  Note:
  The value `always` should be used to describe documents that change each time they are accessed. The value `never` should be used to describe archived URLs.

- **priority** (optional) - The priority of this page relative to other pages on your site. Valid values range from 0.0 to 1.0. This value does not affect how your pages are compared to pages on other sites. It only lets the search engines know which pages you deem most important for the crawlers. The first time the `sitemap.xml` file is generated, the value is set based upon the value of the `webhelp.sitemap.priority` parameter in the DITA WebHelp transformation scenario. You can change the value in each `url` element by editing the `sitemap.xml` file.

Creating and Editing the `sitemap.xml` File

Follow these steps to produce a `sitemap.xml` file for your WebHelp system, which can then be edited to fine-tune search engine optimization:
1. **Edit** the transformation scenario you currently use for obtaining your WebHelp output. This opens the **Edit DITA Scenario** dialog box.

2. Open the **Parameters** tab and set a value for the following parameters:
   - **webhelp.sitemap.base.url** - The URL of the location where your WebHelp system is deployed.
     
     **Note:** This parameter is required for Oxygen XML Editor to generate the *sitemap.xml* file.
   - **webhelp.sitemap.change.frequency** - How frequently the WebHelp pages are likely to change (accepted values are: always, hourly, daily, weekly, monthly, yearly, and never).
   - **webhelp.sitemap.priority** - The priority of each page (value ranging from 0.0 to 1.0).

3. Run the transformation scenario.

4. Look for the *sitemap.xml* file in the transformation's output folder. Edit the file to fine-tune the parameters of each page, according to your needs.

### How to Index Japanese Content

To optimize the indexing of Japanese content in WebHelp pages, the **Lucene Kuromoji Japanese analyzer** can be used. This analyzer is included in the **Oxygen XML Editor/Author** installation kit.

**Restriction:**

The Kuromoji analyzer does not work if your WebHelp output is accessed locally. In this scenario, a warning message will be displayed informing you that the Kuromoji analyzer is disabled.

It is possible to hide this warning message by using a transformation parameter named `webhelp.enable.search.kuromoji.js`. By default, its value is `yes`, which means the Kuromoji analyzer is enabled by default. To hide the warning message, set the value of that parameter to `no` using either of the methods listed below. When it is set to `no`, the Kuromoji analyzer is disabled even if you deploy your WebHelp output on a web server.

### Using a Publishing Template

To add a logo in the title area of your WebHelp output using an **Oxygen Publishing Template** (on page 1603), follow this procedure:

1. If you have not already created a Publishing Template, see How to Create a Publishing Template (on page 1791).
2. Open the **template descriptor file** (on page 1607) associated with your publishing template and add the `default.language` parameter in the **parameters** section with its value set to `ja-jp`.

```xml
<publishing-template>
  ...
  <webhelp>
    ...
  </webhelp>
  <parameters>
    ...
  </parameters>
</publishing-template>
```
Using a Transformation Scenario in Oxygen XML Editor/Author

To activate the Japanese indexing in your WebHelp output using a transformation scenario from within Oxygen XML Editor/Author, follow this procedure:

1. Edit a DITA to WebHelp transformation scenario and in the Parameters tab, set the value of the default.language parameter to ja-jp.

2. Run the WebHelp transformation scenario to generate the output.

Localization

This section contains topics that explain how to use the localization support in WebHelp Responsive output.

How to Localize the Interface of WebHelp Responsive Output

Oxygen XML Editor comes with support for the following built-in languages: English, French, German, Japanese, and Chinese. It is possible to edit existing localization strings or add a new language.

Static labels used in the WebHelp output are stored in translation files that have the strings-lang1-lang2.xml name format, where lang1 and lang2 are ISO language codes. For example, the US English labels are kept in the strings-en-us.xml file.

These translation files are collected from two locations:
• **DITA-OT-DIR/plugins/org.dita.base/xsl/common folder** - DITA-OT's default translations (generated text for `<note>`, `<fig>`, and `<table>` elements).

• **DITA-OT-DIR/plugins/com.oxygenxml.webhelp.responsive/oxygen-webhelp/resources/localization folder** - These translations are contributed by the WebHelp plugin and extend the default ones provided by DITA-OT. The labels defined in this folder take precedence over the DITA-OT defaults.

There are two major reasons you may want to use modify the translation files: to modify the existing strings or to translate to a new language.

**Related Information:**

- How to Index Japanese Content *(on page 1680)*
- Customizing Generated Text

### Modifying the Existing Strings

To modify the generated text for WebHelp transformations, you need to create a DITA-OT extension plugin that uses the `dita.xsl.strings` extension point. The following procedure is for changing English labels, but you can adapt it for any language:

1. Create a `com.oxygenxml.webhelp.localization` plugin directory inside the `DITA-OT-DIR/plugins/` location.
2. Create a `plugin.xml` file inside that `com.oxygenxml.webhelp.localization` directory with the following content:

   ```xml
   <plugin id="com.oxygenxml.webhelp.localization">
     <require plugin="com.oxygenxml.webhelp.responsive"/>

     <feature extension="dita.xsl.strings" file="webhelp-extension-strings.xml"/>
   </plugin>
   ```

3. Create a `webhelp-extension-strings.xml` file with the following content:

   ```xml
   <langlist>
     <lang xml:lang="en" filename="strings-en-us.xml"/>
     <lang xml:lang="en-us" filename="strings-en-us.xml"/>
   </langlist>
   ```

4. Copy the strings you want to change from the translation files *(on page 1681)* to the `strings-en-us.xml` file. Make sure you leave the name attribute unchanged because this is the key used to look up the string. A sample content might be:

   ```xml
   <strings xml:lang="en-US">
     <str name="Figure">Fig</str>
   </strings>
   ```
5. Use the Run DITA-OT Integrator transformation scenario (on page 1455) found in the DITA Map section in the Configure Transformation Scenario(s) dialog box (on page 1548).

**Adding a New Language**

To add a new language for WebHelp transformations, you need to create a DITA-OT extension plugin that uses the `dita.xsl.strings` extension point. The following sample procedure is for adding translation files for the Polish language, but you can adapt it for any language:

1. Create a `com.oxygenxml.webhelp.localization` plugin directory inside the `DITA-OT-DIR/plugins/` location.
2. Create a `plugin.xml` file inside that `com.oxygenxml.webhelp.localization` directory with the following content:

   ```xml
   <plugin id="com.oxygenxml.webhelp.localization">
     <require plugin="com.oxygenxml.webhelp.responsive"/>

     <feature extension="dita.xsl.strings" file="webhelp-extension-strings.xml"/>
   </plugin>
   ```

3. Create a `webhelp-extension-strings.xml` file with the following content:

   ```xml
   <langlist>
     <lang xml:lang="pl" filename="strings-pl-pl.xml"/>
     <lang xml:lang="pl-PL" filename="strings-pl-pl.xml"/>
   </langlist>
   ```


5. In the `strings-pl-pl.xml` file, change the @xml:lang attribute on the root element that conforms with the new language.

   ```xml
   <strings xml:lang="pl-PL">
     ...
   </strings>
   ```

6. Translate the content of each `<str>` element (make sure to leave the name attribute unchanged).

   ```xml
   <strings xml:lang="pl-PL">
     ...
     <str name="webhelp.content" js="true" php="false">Polish translation for 'Content'</str>
     <str name="webhelp.search" js="true" php="false">Polish translation for 'Search'</str>
     ...
   </strings>
   ```
7. Copy the common DITA-OT strings defined in the `DITA-OT-DIR/plugins/org.dita.base/ 
xsl/common/strings-en-us.xml` file. It defines a set generated text available for HTML-based 
transformations (such as `<note>`, `<fig>`, and `<table>` elements). Translate the content of each `<str>` 
element.

```xml
<strings xml:lang="pl-PL">
...
<str name="webhelp.content" js="true" php="false">Polish translation for 'Content'</str>
<str name="webhelp.search" js="true" php="false">Polish translation for 'Search'</str>
...
<str name="Figure">Polish translation for 'Figure'</str>
<str name="Table">Polish translation for 'Table'</str>
...
</strings>
```

8. Use the Run DITA-OT Integrator transformation scenario (on page 1455) found in the DITA Map 
section in the Configure Transformation Scenario(s) dialog box (on page 1548).

**How to Activate Support for Right-to-Left (RTL) Languages**

To activate support for RTL (right-to-left) languages in WebHelp output, edit the DITA map (on page 3218) 
and set the `<xml:lang>` attribute on its root element (<map>). The corresponding attribute value can be set for 
following RTL languages:

- ar-eg - Arabic
- he-il - Hebrew
- ur-pk - Urdu

**Integrating Social Media and Google Tools in the WebHelp Output**

This section contains topics that explain how to integrate some of the most popular social media sites in 
WebHelp output.

**How to Add a Facebook Like Button in WebHelp Responsive Output**

It is possible to integrate Facebook™ into your WebHelp Responsive output and you can specify where you 
want the widget to appear in your WebHelp page.

**Using a Publishing Template**

To add a Facebook™ Like widget to your WebHelp output using an Oxygen Publishing Template (on page 
1603), follow this procedure:

1. Go to the Facebook Developers website.
2. Fill in the displayed form, then click the Get Code button.
3. Copy the two code snippets and paste them into a `<div>` element inside an XML file called `facebook-
   widget.xml`. Make sure you follow these rules:
The file must be well-formed.
- The code for each `<script>` element must be included in an XML comment.
- The start and end tags for the XML comment must be on a separate line. The content of the XML file should look like this:

```xml
<div id="facebook">
  <div id="fb-root"/>
  <script>
    <!--
    (function(d, s, id) {
      var js, fjs = d.getElementsByTagName(s)[0];
      if (d.getElementById(id)) return;
      js = d.createElement(s); js.id = id;
      js.src = "//connect.facebook.net/en_US/sdk.js#xfbml=1&version=v2.0";
      fjs.parentNode.insertBefore(js, fjs);
    } document, 'script', 'facebook-jssdk'));
    -->
  </script>
  <div class="fb-like" data-layout="standard" data-action="like"
       data-show-faces="true" data-share="true"/>
</div>
```

4. Open the template descriptor file (on page 1607) associated with your publishing template.
5. Use one of the parameters that begin with `webhelp.fragment` (on page 1613) in the `html-fragments` section of the descriptor file. Set the value of that parameter to reference the `facebook-widget.xml` file that you created earlier.

```xml
<publishing-template>
  ...
  <webhelp>
    ...
    <html-fragments>
      <fragment
        file="HTML-fragments/facebook-widget.xml"
        placeholder="webhelp.fragment.after.toc_or_tiles"/>
    </html-fragments>
  </webhelp>
</publishing-template>
```

6. Open the DITA Map WebHelp Responsive transformation scenario.
7. Click the Choose Custom Publishing Template link and select your template.
8. Click OK to save the changes to the transformation scenario.
9. Run the transformation scenario.
Using a Transformation Scenario in Oxygen XML Editor/Author

To add a Facebook™ Like widget to your WebHelp output using a transformation scenario from within Oxygen XML Editor/Author, follow this procedure:

1. Go to the Facebook Developers website.
2. Fill in the displayed form, then click the Get Code button.
3. Copy the two code snippets and paste them into a `<div>` element inside an XML file called `facebook-widget.xml`. Make sure you follow these rules:
   ◦ The file must be well-formed.
   ◦ The code for each `<script>` element must be included in an XML comment.
   ◦ The start and end tags for the XML comment must be on a separate line. The content of the XML file should look like this:

   ```xml
   <div id="facebook">
     <div id="fb-root"/>
     <script>
       <!--
       (function(d, s, id) {
         var js, fjs = d.getElementsByTagName(s)[0];
         if (d.getElementById(id)) return;
         js = d.createElement(s); js.id = id;
         js.src = "//connect.facebook.net/en_US/sdk.js#xfbml=1&version=v2.0";
         fjs.parentNode.insertBefore(js, fjs);
       }(document, 'script', 'facebook-jssdk'));
       -->
     </script>
   <div class="fb-like" data-layout="standard" data-action="like"
    data-show-faces="true" data-share="true"/>
     </div>
   </div>
   ```

4. Edit the DITA Map WebHelp Responsive transformation scenario and choose a template.
5. Switch to the Parameters tab. Depending on where you want to display the button, edit one of the parameters that begin with `webhelp.fragment` (on page 1613). Set that parameter to reference the `facebook-widget.xml` file that you created earlier.
6. Click Ok and run the transformation scenario.

How to Add Tweet Button in WebHelp Responsive Output

It is possible to integrate Twitter into your WebHelp Responsive output and you can specify where you want the widget to appear in your WebHelp page.

Using a Publishing Template

To add a Twitter™ Tweet widget to your WebHelp Responsive output using an Oxygen Publishing Template (on page 1603), follow this procedure:
1. Go to the Tweet button generator page.
2. Fill in the displayed form. The Preview and code area displays the code that you will need.
3. Copy the code snippet displayed in the Preview and code area and paste it into a `<div>` element inside an XML file called `tweet-button.xml`. Make sure you follow these rules:
   - The file must be well-formed.
   - The code for each `<script>` element must be included in an XML comment.
   - The start and end tags for the XML comment must be on a separate line.

   The content of the XML file should look like this:

   ```xml
   <div id="twitter">
     <a href="https://twitter.com/share" class="twitter-share-button">Tweet</a>
     <script>
       !function(d, s, id) {
         var js, fjs = d.getElementsByTagName(s)[0], p = /^http:/.test(d.location) ? 'http' : 'https';
         if (!d.getElementById(id)) {
           js = d.createElement(s); js.id = id;
           js.src = p + '://platform.twitter.com/widgets.js';
           fjs.parentNode.insertBefore(js, fjs);
         }
       }(document, 'script', 'twitter-wjs');
     </script>
   </div>
   ```

4. Open the template descriptor file (on page 1607) associated with your publishing template.
5. Use one of the parameters that begin with `webhelp.fragment` (on page 1613) in the `html-fragments` section of the descriptor file. Set the value of that parameter to reference the `tweet-button.xml` file that you created earlier.

   ```xml
   <publishing-template>
     ...
     <webhelp>
       ...
     </webhelp>
     <html-fragments>
       <fragment
         file="HTML-fragments/tweet-button.xml"
         placeholder="webhelp.fragment.after.toc_or_tiles"/>
   ```
6. Open the **DITA Map WebHelp Responsive** transformation scenario.
7. Click the **Choose Custom Publishing Template** link and select your template.
8. Click **OK** to save the changes to the transformation scenario.
9. Run the transformation scenario.

### Using a Transformation Scenario in Oxygen XML Editor/Author

To add a Twitter™ **Tweet** widget to your WebHelp Responsive output using a transformation scenario from within **Oxygen XML Editor/Author**, follow this procedure:

1. Go to the **Tweet button generator** page.
2. Fill in the displayed form. The **Preview and code** area displays the code that you will need.
3. Copy the code snippet displayed in the **Preview and code** area and paste it into a `<div>` element inside an XML file called `tweet-button.xml`. Make sure you follow these rules:
   - The file must be well-formed.
   - The code for each `<script>` element must be included in an XML comment.
   - The start and end tags for the XML comment must be on a separate line.

The content of the XML file should look like this:

```xml
<div id="twitter">
  <a href="https://twitter.com/share" class="twitter-share-button">Tweet</a>
  <script type="text/javascript">
    function(d, s, id) {
      var js, fjs = d.getElementsByTagName(s)[0], p = /^http:/.test(d.location) ? 'http' : 'https';
      if (! d.getElementById(id)) {
        js = d.createElement(s);
        js.id = id;
        js.src = p + '//platform.twitter.com/widgets.js';
        fjs.parentNode.insertBefore(js, fjs);
      }
      (document, 'script', 'twitter-wjs');
    }
    (document, 'script', 'twitter-wjs');
  </script>
</div>
```

4. Edit the **DITA Map WebHelp Responsive** transformation scenario and choose a **template**.
5. Switch to the **Parameters** tab. Depending on where you want to display the button, edit **one of the parameters that begin with** webhelp.fragment (on page 1613). Set that parameter to reference the tweet-button.xml file that you created earlier.

6. Click **Ok** and run the transformation scenario.

### How to Integrate Google Analytics in WebHelp Responsive Output

You can use **Google Analytics** to track and report site data for your **WebHelp Responsive** output.

#### Using a Publishing Template

To integrate **Google Analytics** into your WebHelp Responsive output using an **Oxygen Publishing Template** (on page 1603), follow this procedure:

1. Create a new **Google Analytics account** (if you do not already have one) and log on.
2. Choose the Analytics solution that best fits the needs of your website.
3. Follow the on-screen instructions to obtain a **Tracking Code** that contains your **Tracking ID**. A **Tracking Code** looks like this:

   ```
   <script>(function(i,s,o,g,r,a,m){i['GoogleAnalyticsObject']=r;i[r]=i[r]||function(){
   (i[r].q=i[r].q||[]).push(arguments)},i[r].l=1*new Date();a=s.createElement(o),
m=s.getElementsByTagName(o)[0];a.async=1;a.src=g;m.parentNode.insertBefore(a,m)
 })(window,document,'script','//www.google-analytics.com/analytics.js','ga');
   
   ga('create', 'UA-XXXXXXXX-X', 'auto');

   ga('send', 'pageview');
   </script>
   ```

4. Save the Tracking Code (obtained in the previous step) in a new XML file called googleAnalytics.xml. Note that the file should only contain the tracking code.

5. Open the **template descriptor file** (on page 1607) associated with your publishing template.

6. Use the **webhelp.fragment.after.body parameter** (on page 1719) in the **html-fragments** section of the descriptor file. Set the value of that parameter to reference the googleAnalytics.xml file that you created earlier. The content of this file will be copied at the end of all generated output pages, right before the ending `<body>` element. This ensures that the page is loaded before the Google Analytics servers are contacted, thus reducing page loading time.

   ```
   <publishing-template>
   ...
   <webhelp>
   ...
   
   <html-fragments>

   <fragment

   file="HTML-fragments/googleAnalytics.xml"
   placeholder="webhelp.fragment.after.body"/>
   ```
7. Open the **DITA Map WebHelp Responsive** transformation scenario.
8. Click the **Choose Custom Publishing Template** link and select your template.
9. Click **OK** to save the changes to the transformation scenario.
10. Run the transformation scenario.

**Using a Transformation Scenario in Oxygen XML Editor/Author**

To integrate **Google Analytics** into your WebHelp Responsive output using a transformation scenario from within **Oxygen XML Editor/Author**, follow this procedure:

1. Create a new **Google Analytics account** (if you do not already have one) and log on.
2. Choose the Analytics solution that best fits the needs of your website.
3. Follow the on-screen instructions to obtain a **Tracking Code** that contains your **Tracking ID**. A **Tracking Code** looks like this:

```html
<script>
(function(i,s,o,g,r,a,m){i['GoogleAnalyticsObject']=r;i[r]=i[r]|||
function(){
(i[r].q=i[r].q||[]).push(arguments)},i[r].l=1
*new Date();a=s.createElement(o),
m=s.getElementsByTagName(o)[0];a.async=1;a.src=g;m.parentNode.insertBefore(a,m)
}(window,document,'script','//www.google-analytics.com/analytics.js','ga'));

ga('create', 'UA-XXXXXXXX-X', 'auto');
ga('send', 'pageview');
</script>
```

4. Save the Tracking Code (obtained in the previous step) in a new XML file called `googleAnalytics.xml`. Note that the file should only contain the tracking code.
5. Edit the **DITA Map WebHelp Responsive** transformation scenario and choose a **template**.
6. Switch to the **Parameters** tab. Edit the `webhelp.fragment.after.body` parameter **(on page 1719)** and set it to reference the `googleAnalytics.xml` file that you created earlier. The content of this file will be copied at the end of all generated output pages, right before the ending `<body>` element. This ensures that the page is loaded before the Google Analytics servers are contacted, thus reducing page loading time.
7. Click **Ok** and run the transformation scenario.

**How to Integrate Google Search in WebHelp Responsive Output**

It is possible to integrate the **Google Search Engine** into your **WebHelp Responsive** output and you can specify where you want the results to appear in your WebHelp page.

**Using a Publishing Template**

To integrate the **Google Search Engine** into your WebHelp Responsive output using an **Oxygen Publishing Template (on page 1603)**, follow this procedure:
1. Go to the Google Custom Search Engine page using your Google account.
2. Select the Create a custom search engine button.
3. Follow the on-screen instructions to create a search engine component for your site.

⚠️ Important:
For the Layout, you must select Results only for the Google Search Engine to work with Oxygen XML WebHelp Responsive.

4. At the end of this process you should obtain a code snippet that looks like this:

```html
<script>
(function() {
  var cx = '000888210889775888983:8mn4x_mf-yg';
  var gsc = document.createElement('script');
  gsc.type = 'text/javascript';
  gsc.async = true;
  var s = document.getElementsByTagName('script')[0];
  s.parentNode.insertBefore(gsc, s);
})();
</script>
```

5. Save the script into a well-formed HTML file called googlecse.html.
6. Open the template descriptor file (on page 1607) associated with your publishing template and add the webhelp.google.search.script parameter in the parameters section with its value set to reference the googlecse.html file that you created earlier.

```xml
<publishing-template>
  ...
  <webhelp>
    ...
    <parameters>
      <parameter
        name="webhelp.google.search.script"
        value="resources/googlecse.html"
        type="filePath"/>
    </parameters>
  </webhelp>
</publishing-template>
```

7. You can also use the webhelp.google.search.results parameter to choose where to display the search results.
Using a Transformation Scenario in Oxygen XML Editor/Author

To integrate the Google Search Engine into your WebHelp Responsive output using a transformation scenario from within Oxygen XML Editor/Author, follow this procedure:

1. Go to the Google Custom Search Engine page using your Google account.
2. Select the Create a custom search engine button.
3. Follow the on-screen instructions to create a search engine for your site.

Important:

For the Layout, you must select Results only for the Google Search Engine to work with Oxygen XML WebHelp Responsive.

4. At the end of this process you should obtain a code snippet that looks like this:

```html
<script>
(function() {
    var cx = '000888210889775888983:8mn4x-mf-yg';
    var gcse = document.createElement('script');
    gcse.type = 'text/javascript';
    gcse.async = true;
    var s = document.getElementsByTagName('script')[0];
    s.parentNode.insertBefore(gcse, s);
})();
</script>
```

5. Save the script into a well-formed HTML file called googlecse.html.
6. Edit the DITA Map WebHelp Responsive transformation scenario and choose a template.
7. Switch to the Parameters tab and edit the webhelp.google.search.script parameter to reference the googlecse.html file that you created earlier.
8. You can also use the webhelp.google.search.results parameter to choose where to display the search results.
   a. Create an HTML file with the following content:
      `<div class="gcse-searchresults-only" data-autoSearchOnLoad="true" data-queryParameterName="searchQuery"></div>`
      (you must use the HTML5 version for the GCSE). For more information about other supported attributes, see Google Custom Search: Supported Attributes.
   b. Set the value of the webhelp.google.search.results parameter to the file path of the file you just created. If this parameter is not specified, the following code is used:
      `<div class="gcse-searchresults-only" data-autoSearchOnLoad="true" data-queryParameterName="searchQuery"></div>`.
9. Click Ok and run the transformation scenario.

**Ant Extensions for WebHelp Responsive**

The WebHelp Responsive plugin provides extension points that allow you to implement custom Ant targets to perform additional operations before and after certain processing stages. The following extension points are available in WebHelp Responsive:

- **whr-init-pre**
  Runs a custom Ant target before the whr-init processing stage.

- **whr-init-post**
  Runs a custom Ant target after the whr-init processing stage.

- **whr-collect-indexterms-pre**
  Runs a custom Ant target before the whr-collect-indexterms processing stage.

- **whr-collect-indexterms-post**
  Runs a custom Ant target after the whr-collect-indexterms processing stage.

- **whr-toc-xml-pre**
  Runs a custom Ant target before the whr-toc-xml processing stage.

- **whr-toc-xml-post**
  Runs a custom Ant target after the whr-toc-xml processing stage.

- **whr-context-help-map-pre**
  Runs a custom Ant target before the whr-context-help-map processing stage.

- **whr-context-help-map-post**
  Runs a custom Ant target after the whr-context-help-map processing stage.

- **whr-sitemap-pre**
  Runs a custom Ant target before the whr-sitemap processing stage.
To use Ant extension points for WebHelp Responsive, follow these steps:

1. In the DITA-OT-DIR/plugins/ folder, create a folder for this plugin (for example, com.oxygenxml.webhelp.responsive.custom.ant.extensions).
2. Create a plugin.xml file (in the folder you created in step 1) that extends the WebHelp Responsive plugin and specifies an Ant extension point with your custom Ant project file that contains the new build targets. For example:
3. Create the custom_build_file.xml file (in the folder you created in step 1) that contains your custom Ant project implementing one or more extension points:

```xml
<project name="custom.ant.extensions.integrator" basedir="."/>

<target name="custom-whr-init-pre" extensionOf="whr-init-pre">
    <echo>Extension point that executes before whr-init</echo>
</target>

<target name="custom-whr-init-post" extensionOf="whr-init-post">
    <echo>Extension point that executes after whr-init</echo>
</target>
</project>
```

4. Integrate the plugin into the DITA-OT. In the DITA-OT-DIR/bin directory of the DITA Open Toolkit, run one of the following scripts, depending on your operating system:

- **Windows:** DITA-OT-DIR/bin/dita.bat --install
- **Linux/macOS:** sh DOTA-OT-DIR/bin/dita --install

5. Execute a DITA Map to WebHelp Responsive transformation script.

### XSLT Extensions for WebHelp Responsive

Since WebHelp Responsive output is primarily obtained by running XSLT transformations over the DITA input files, one customization method would be to override the default XSLT templates that are used by the WebHelp Responsive transformations.

There are two methods available to override the XSLT stylesheets implied by the WebHelp Responsive transformation:

- **Use XSLT-import extension points from an Oxygen Publishing Template (on page 3221).**

  **Note:**
  Use this method if you want to affect only the transformations that use this publishing template.

- **Use XSLT-import extension points from a DITA-OT extension plugin.**

  **Note:**
  This method will affect all the outputs generated with the WebHelp system.
How to Use XSLT Extension Points from a Publishing Template

This example demonstrates how to use WebHelp XSLT-import Extension Points from an Oxygen Publishing Template (on page 1783).

Use Case 1: Add Copyright Information Extracted from a DITA Bookmap

Suppose you want to customize the WebHelp Responsive main page by adding information about the legal rights associated with the book in the footer (for example, copyright dates and owner). This information is specified in the bookmap:

```xml
<bookrights>
  <copyrfirst>
    <year>2002</year>
  </copyrfirst>
  <copyrlast>
    <year>2017</year>
  </copyrlast>
  <bookowner>
    <organization>SyncRO Soft SRL</organization>
  </bookowner>
</bookrights>
```

Figure 499. Example: Copyright Information Added in the WebHelp Footer

The XSLT stylesheet that generates the main page is located in: DITA-OT-DIR\plugins\com.oxygenxml.webhelp.responsive\xsl\mainFiles\createMainPage.xsl. This XSLT stylesheet declares the copy_template mode that processes the main page template (on page 1624) to expand its components. The main page template declares a component for the footer section that looks like this:

```html
<div class="footer-container text-center">
  <whc:include_html href="${webhelp.fragment.footer}"
</div>
```
In the following example, the extension stylesheet will add a template that matches this component. It applies the default processing and adds the copyright information at the end.

```xml
<xsl:template match="*:div[contains(@class, 'footer-container')]">
  <!-- Apply the default processing -->
  <xsl:next-match/>
  <!-- Add a div containing the copyright information -->
  <div class="copyright_info">
    <xsl:choose>
      <!-- Adds the start-end years if they are defined -->
      <xsl:when test="exists($toc/*:topicmeta/*:bookrights/*:copyrfirst) and exists($toc/*:topicmeta/*:bookrights/*:copyrlast)"/>
        <span class="copyright_years">
          © <xsl:value-of select="$toc/*:topicmeta/*:bookrights/*:copyrfirst"/>
          - <xsl:value-of select="$toc/*:topicmeta/*:bookrights/*:copyrlast"/>
        </span>
      </xsl:when>
      <!-- Adds only the first year if last is not defined. -->
      <xsl:when test="exists($toc/*:topicmeta/*:bookrights/*:copyrfirst)"/>
        <span class="copyright_years">
          © <xsl:value-of select="$toc/*:topicmeta/*:bookrights/*:copyrfirst"/>
        </span>
      </xsl:when>
    </xsl:choose>
    <xsl:if test="exists($toc/*:topicmeta/*:bookrights/*:bookowner/*:organization)">
      <span class="organization">
        <xsl:text> </xsl:text>
        <xsl:value-of select="$toc/*:topicmeta/*:bookrights/*:bookowner/*:organization"/>
        <xsl:text>. All rights reserved.</xsl:text>
      </span>
    </xsl:if>
  </div>
</xsl:template>
```

To add this functionality using a Oxygen Publishing Template, follow these steps:

1. If you have not already created a Publishing Template, see How to Create a Publishing Template (on page 1791).
2. Link the folder associated with the publishing template to your current project in the Project view.
   You should have the custom_footer_template folder linked in your project.
3. Using the **Project** view, create an `xslt` folder inside the project root folder. You should have the `custom_footer_template/xsl` folder in your project.

4. Create your customization stylesheet (for example, `custom_mainpage.xsl`) in the `custom_footer_template/xsl` folder. Edit it to override the template that produces the footer section:

```xml
<xsl:template match="*:div[contains(@class, 'footer-container')]" mode="copy_template">
  <!-- Apply the default processing -->
  <xsl:next-match/>

  <!-- Add a div containing the copyright information -->
  <div class="copyright_info">
    <xsl:choose>
      <!-- Adds the start-end years if they are defined -->
      <xsl:when test="exists($toc/*/topicmeta/*/bookrights/*:copyrfirst) and exists($toc/*/topicmeta/*/bookrights/*:copyrlast)">
        <xsl:when test="not($toc/*/topicmeta/*/bookrights/*:copyrlast)">
          ©<xsl:value-of select="$toc/*/topicmeta/*/bookrights/*:copyrfirst"/>
        </xsl:when>
        <xsl:when test="exists($toc/*/topicmeta/*/bookrights/*:copyrlast)">
          ©<xsl:value-of select="$toc/*/topicmeta/*/bookrights/*:copyrlast"/>
        </xsl:when>
      </xsl:when>

      <!-- Adds only the first year if last is not defined. -->
      <xsl:when test="exists($toc/*/topicmeta/*/bookrights/*:copyrfirst)">
        ©<xsl:value-of select="$toc/*/topicmeta/*/bookrights/*:copyrfirst"/>
      </xsl:when>
    </xsl:when>
  </xsl:choose>

  <xsl:if test="exists($toc/*/topicmeta/*/bookrights/*:bookowner/*:organization)">
    <span class="organization">
      <xsl:text>©</xsl:text><xsl:value-of select="$toc/*/topicmeta/*/bookrights/*:bookowner/*:organization"/>
    </span>
  </xsl:if>
</div>
</xsl:template>
```

5. Open the **template descriptor file (on page 1607)** associated with your publishing template and set the XSLT stylesheet created in the previous step with the `com.oxygenxml.webhelp.xsl.createMainPage` XSLT extension point.
6. Open the DITA Map WebHelp Responsive transformation scenario.
7. Click the Choose Custom Publishing Template link and select your template.
8. Click OK to save the changes to the transformation scenario.
9. Run the transformation scenario.

**Use Case 2: Add Generation Time in the Output Footer**

Another possible customization for the main page is to add the generation time in its footer. A transformation parameter is used to control whether or not this customization is active.

![Figure 500. Generation Time Added in the WebHelp Footer](image)

To add this functionality, follow these steps:

1. In the customization stylesheet that you just created (for example, `custom_mainpage.xsl`), modify the template by adding the following XSLT code at the end.

```xml
<xsl:if test="oxyf:getParameter('webhelp.footer.add.generation.time') = 'yes'">  
  <div class="generation_time">  
    Generation date: <xsl:value-of select="format-dateTime(current-dateTime(), '[h1]:[m01] [P] on [M01]/[D01]/[Y0001].')"/>  
  </div>  
</xsl:if>
```

**Note:**
You can read the value of a WebHelp transformation parameter from your XSLT extension stylesheets by using the `setParameter(param.name)` function from the `http://www.oxygenxml.com/functions` namespace.
2. Open the **template descriptor file** *(on page 1607)* associated with your publishing template and set the `webhelp.footer.add.generation.time` parameter to the default value.

```xml
<publishing-template>
  ...
  <webhelp>
    ...
    <parameters>
      <parameter
        name="webhelp.footer.add.generation.time"
        value="yes"/>
  </parameters>
  ...
</webhelp>
</publishing-template>
```

3. Open the **DITA Map WebHelp Responsive** transformation scenario.
4. In the **Parameters** tab, you can change the value of the `webhelp.footer.add.generation.time` parameter.
5. Click **OK** to save the changes to the transformation scenario.
6. Run the transformation scenario.

**How to Use XSLT Extension Points from a DITA-OT Plugin**

In this example, the main page footer is modified by adding copyright information extracted from the DITA bookmap or by adding the output generation time. The first use-case uses an **XSLT-Import** extension point while the second uses an **XSLT-Parameter** extension point.

**Note:**

This customization is available as a GitHub project at: [https://github.com/oxygenxml/com.oxygenxml.webhelp.responsive.custom.footer](https://github.com/oxygenxml/com.oxygenxml.webhelp.responsive.custom.footer).

**Use Case 1: WebHelp **XSLT-Import** extension point to add copyright information extracted from a DITA Bookmap**

Suppose you want to customize the WebHelp Responsive main page by adding information about the legal rights associated with the book in the footer (for example, copyright dates and owner). This information is specified in the bookmap:

```xml
<bookrights>
  <copyrfirst>
    <year>2002</year>
  </copyrfirst>
  <copyrlast>
    <year>2017</year>
  </copyrlast>
  <bookowner>
    <organization>SyncRO Soft SRL</organization>
  </bookowner>
</bookrights>
```
Figure 501. Example: Copyright Information Added in the WebHelp Footer

The XSLT stylesheet that generates the main page is located in: `DITA-OT-DIR\plugins\com.oxygenxml.webhelp.responsive\xsl\mainFiles\createMainPage.xsl`. This XSLT stylesheet declares the `copy_template` mode that processes the main page template to expand its components. The main page template (on page 1624) declares a component for the footer section that looks like this:

```xml
<div class="footer-container text-center">
  <whc:include_html href="${webhelp.fragment.footer}"/>
</div>
```

In the following example, the extension stylesheet will add a template that matches this component. It applies the default processing and adds the copyright information at the end.

```xml
<xsl:template match="*:div[contains(@class, 'footer-container')]
  mode="copy_template">
  <!-- Apply the default processing -->
  <xsl:next-match/>
  <!-- Add a div containing the copyright information -->
  <div class="copyright_info">
    <xsl:choose>
      <!-- Adds the start-end years if they are defined -->
      <xsl:when test="exists($toc/*:topicmeta/*:bookrights/*:copyrfirst) and 
        exists($toc/*:topicmeta/*:bookrights/*:copyrlast)">
        <span class="copyright_years">
          ©<xsl:value-of select="$toc/*:topicmeta/*:bookrights/*:copyrfirst"/>
          -<xsl:value-of select="$toc/*:topicmeta/*:bookrights/*:copyrlast"/>
        </span>
      </xsl:when>
      <!-- Adds only the first year if last is not defined. -->
      <xsl:when test="exists($toc/*:topicmeta/*:bookrights/*:copyrfirst)">
        <span class="copyright_years">
          ©<xsl:value-of select="$toc/*:topicmeta/*:bookrights/*:copyrfirst"/>
        </span>
      </xsl:when>
    </xsl:choose>
  </div>
</xsl:template>
```
You can implement this functionality with a WebHelp extension plugin that uses the `com.oxygenxml.webhelp.xsl.createMainPage` extension point (on page 1732). This extension point allows you to specify a customization stylesheet that will override the template described above.

To add this functionality as a DITA-OT plugin, follow these steps:

1. In the `DITA-OT-DIR\plugins` folder, create a folder for this plugin (for example, `com.oxygenxml.webhelp.responsive.custom.footer`).
2. Create a `plugin.xml` file (in the folder you created in step 1) that specifies the extension point and your customization stylesheet. For example:

```xml
<plugin id="com.oxygenxml.webhelp.responsive.custom.footer">
  <feature extension="com.oxygenxml.webhelp.xsl.createMainPage"
           file="custom_mainpage.xsl"/>
</plugin>
```

3. Create your customization stylesheet (for example, `custom_mainpage.xsl`), and edit it to override the template that produces the footer section:

```xml
<xsl:template match="/div[contains(@class, 'footer-container')]
               mode="copy_template">
  <!-- Apply the default processing -->
  <xsl:next-match/>
  
  <!-- Add a div containing the copyright information -->
  <div class="copyright_info">
    <xsl:choose>
      <!-- Adds the start-end years if they are defined -->
      <xsl:when test="exists($toc::*:topicmeta::*:bookrights::*:copyrfirst) and exists($toc::*:topicmeta::*:bookrights::*:copyrlast)">
        <span class="copyright_years">
          ©
          <xsl:value-of select="$toc::*:topicmeta::*:bookrights::*:copyrfirst"/>
          -
          <xsl:value-of select="$toc::*:topicmeta::*:bookrights::*:copyrlast"/>
        </span>
      </xsl:when>
    </xsl:choose>
  </div>
</xsl:template>
```
4. Use the Run DITA-OT Integrator transformation scenario (on page 1455) found in the DITA Map section in the Configure Transformation Scenario(s) dialog box (on page 1548).

5. Run a DITA Map WebHelp Responsive transformation scenario to obtain the customized side TOC.

Use-Case 2: WebHelp XSLT-Parameter Extension Point to Control if Generation Time is Displayed in the Output

Another possible customization for the main page is to add the generation time in its footer. You can use an XSLT-Parameter extension point to control whether or note this customization is active. In this case, you can use the com.oxygenxml.webhelp.xsl.createMainPage.param extension point (on page 1733).

Figure 502. Generation Time Added in the WebHelp Footer

To add this functionality, follow these steps:

1. Create a DITA-OT plugin structure by following the first 3 steps in the procedure above (on page 1700).
2. In the customization stylesheet that you just created (for example, custom_mainpage.xsl), declare webhelp.footer.add.generation.time as a global parameter and modify the template by adding the following XSLT code at the end.
3. Edit the **plugin.xml** file to specify the `com.oxygenxml.webhelp.xsl.createMainPage.param` extension point and a custom parameter file by adding the following line:

```xml
<feature extension="com.oxygenxml.webhelp.xsl.createMainPage.param" file="params.xml"/>
```

4. Create a custom parameter file (for example, `params.xml`). It should look like this:

```xml
<dummy>
  <param name="webhelp.footer.add.generation.time"
    expression="${webhelp.footer.add.generation.time}"
    if="webhelp.footer.add.generation.time"/>
</dummy>
```

5. Use the **Run DITA-OT Integrator** transformation scenario *(on page 1455)* found in the **DITA Map** section in the **Configure Transformation Scenario(s)** dialog box *(on page 1548)*.

6. Edit a **DITA Map WebHelp Responsive** transformation scenario and in the **Parameters** tab *(on page 3112)*, specify the desired value *(yes or no)* for your custom parameter *(webhelp.footer.add.generation.time)*.

7. Run the transformation scenario.

**Related Information:**

[DITA-OT] XSLT-Import Extension Points

[DITA-OT] XSLT-Parameter Extension Points

**Miscellaneous Customization Topics**

This section contains miscellaneous topics about how to customize the WebHelp Responsive output.

**How to Copy Additional Resources to Output Directory**

You can copy additional resources (such as graphics, JavaScript, CSS, entire folders, or other resources) to the output directory either by using an **Oxygen Publishing Template** *(on page 3221)* or the `webhelp.custom.resources` parameter.

**Copying Additional Resources to the Output Directory using a Publishing Template**

1. If you have not already created a Publishing Template, see **How to Create a Publishing Template** *(on page 1791)*.
2. Add a new `<fileset>` element in the **resources** section of the template descriptor file *(on page 1610)*.
<publishing-template>

... 

<webhelp>

... 

<resources>

<fileset>

<include name="custom-resources/**/**"/>

<exclude name="**/*.git"/>

</fileset>

</resources>

</webhelp>

</publishing-template>

Note:
Relative paths in the descriptor file are relative to the template root folder.

3. Open the DITA Map WebHelp Responsive transformation scenario.
4. Click the Choose Custom Publishing Template link and select your template.
5. Click OK to save the changes to the transformation scenario.
6. Run the transformation scenario.

Results: All files from the custom resources directory will be copied to the WebHelp Output Directory/oxygen-webhelp/template folder.

Copying Additional Resources to the Output Directory using a Transformation Parameter

1. Place all your resources in the same directory.
2. Edit the DITA Map WebHelp Responsive transformation scenario and open the Parameters tab.
3. Edit the value of the webhelp.custom.resources parameter and set it to the absolute path of the directory in step 1.
4. Click OK to save the changes to the transformation scenario.
5. Run the transformation scenario.

Results: All files from the new directory will be copied to the root of the WebHelp output directory.

How to Add an Edit Link to Launch Oxygen XML Web Author

You can embed Edit links in the DITA WebHelp Responsive output that will automatically launch a particular document in Oxygen XML Web Author. A reviewer can then click the link to open the particular file in Oxygen XML Web Author where they can make or propose changes.

Using a Publishing Template

To embed an Edit link in the DITA Map WebHelp Responsive output using an Oxygen Publishing Template (on page 1603), follow this procedure:
1. If you have not already created a Publishing Template, see Working with Publishing Templates (on page 1642).
2. Open the template descriptor file (on page 1607) associated with your publishing template and add the following parameters with their values set to the URLs:
   - `editlink.ditamap.edit.url` - The URL of the DITA map used to publish your content. The easiest way to obtain the URL is to open the map in Web Author and copy the URL from the browser's address bar.
   - `editlink.additional.query.parameters` - Optional query parameters to be appended to each generated edit link. Each parameter must start with & (e.g. `&tags-mode=no-tags`).

```xml
<publishing-template>
  ...
  <webhelp>
  ...
  <parameters>
    <parameter name="editlink.ditamap.edit.url" value="webdav-https://dav.box.com/dav/my.ditamap"/>
  </parameters>
  </webhelp>
</publishing-template>
```
3. Open the DITA Map WebHelp Responsive transformation scenario.
4. Click the Choose Custom Publishing Template link and select your template.
5. Click OK to save the changes to the transformation scenario.
6. Run the transformation scenario.

**Result:** In the WebHelp output, all topics will have an Edit link to the right side of the title and clicking the link will launch that particular document in Oxygen XML Web Author.

For example:

- **Windows:**

  ```
  ```

- **macOS/ Linux:**

  ```
  ```

**Using a Transformation Scenario in Oxygen XML Editor/Author**

To embed an Edit link in the DITA Map WebHelp Responsive output using a transformation scenario from within Oxygen XML Editor/Author, follow this procedure:

1. Edit a DITA Map WebHelp Responsive transformation scenario and open the Parameters tab.
2. Set values for the following parameters:
- `editlink.ditamap.edit.url` - The URL of the Oxygen XML Web Author that have opened the DITA map for editing.
- `editlink.additional.query.parameters` - Optional query parameters to be appended to each generated edit link. Must start with & (e.g.: `&tags-mode=no-tags`).

3. Run the transformation scenario.

**Result:** In the WebHelp output, all topics will have an **Edit** link to the right side of the title and clicking the link will launch that particular document in Oxygen XML Web Author.

**Related information**

- [Web Author Customization Guide: Embedding an Edit Link that will Launch Web Author](#)
- [How to Flag DITA Content in WebHelp Output](#)

### How to Flag DITA Content in WebHelp Output

Flagging content in WebHelp output involves defining a set of images that will be used for marking content across your information set.

To flag DITA content, you need to create a filter file that defines properties that will be applied on elements to be flagged. Generally, flagging is supported for **block elements** *(on page 3216)* (such as paragraphs), but not for phrase-level elements within a paragraph. This ensures that the images that will flag the content are easily scanned by the reader, instead of being buried in the text.

**Using a Publishing Template**

To flag content in DITA Map to WebHelp output using an **Oxygen Publishing Template** *(on page 1603)*, follow this procedure:

1. Create a DITA filter file (DITAVAL) and add it in a directory of your choice (for example, named `myFile.ditaval`).
2. Define the property for the elements you want to be flagged. For example, if you want to flag any element that has the `@audience` attribute set to `programmer`, the content of the DITAVAL file should look like this:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<val>
    <prop att="audience" val="programmer" action="flag"
    img="D:\resource\delta.gif" alt="sample alt text"/>
</val>
```

**Note:**

For an element to be flagged, at least one attribute-value pair needs to have a property declared in the DITAVAL file.
3. Open the template descriptor file (on page 1607) associated with your publishing template and add the `args.filter` parameter in the `parameters` section with its value set to the path of the DITAVAL file you created.

   ```xml
   <publishing-template>
     ...
     <webhelp>
     ...
     <parameters>
       <parameter name="args.filter" value="resources/myFile.ditaval"/>
     </parameters>
   </webhelp>
   </publishing-template>
   
   4. Open the DITA Map WebHelp Responsive transformation scenario.
   5. Click the Choose Custom Publishing Template link and select your template.
   6. Click OK to save the changes to the transformation scenario.
   7. Run the transformation scenario.

Using a Transformation Scenario in Oxygen XML Editor/Author

To flag content in the DITA Map to WebHelp output using a transformation scenario from within Oxygen XML Editor/Author, follow this procedure:

1. Create a DITA filter file (DITAVAL) and add it in a directory of your choice (for example, named `myFile.ditaval`).
2. Define the property for the elements you want to be flagged. For example, if you want to flag any element that has the `@audience` attribute set to `programmer`, the content of the DITAVAL file should look like this:

   ```xml
   <?xml version="1.0" encoding="UTF-8"?>
   <val>
     <prop att="audience" val="programmer" action="flag"
     img="D:\resource\delta.gif" alt="sample alt text"/>
   </val>
   
   Note:
   For an element to be flagged, at least one attribute-value pair needs to have a property declared in the DITAVAL file.

   3. Edit a DITA Map to WebHelp transformation scenario.
   4. Specify the DITAVAL file in the Filters tab (with the Use DITAVAL File option).
   5. Run the transformation scenario.
How to View MathML Equations in HTML Output

By default, only Firefox can render MathML equations embedded in the HTML code. MathJax is a solution to properly view MathML equations embedded in HTML content in a variety of browsers.

If you have DocBook or DITA content that has embedded MathML equations and you want to properly view the equations in published HTML output types (WebHelp, CHM, EPUB, etc.), you need to add a reference to the MathJax script in the head element of all HTML files that have the equation embedded.

For example:

```html
<script type="text/javascript"
src="https://cdnjs.cloudflare.com/ajax/libs/mathjax/2.7.1/MathJax.js?config=TeX-AMS-MML_HTMLorMM"
>
</script>
```

Alternate Method for DITA

For DITA documents, you can also use the following procedure:

1. Create an XML file that contains a script similar to the one shown in the example above.
2. Edit the DITA Map transformation scenario and open the Parameters tab.
3. Set the following parameter to point to the XML file created in step 1:
   - WebHelp Responsive Systems - Set the `webhelp.fragment.head` parameter to point to your XML file.
   - WebHelp Classic Systems - Set the `webhelp.head.script` parameter to point to your XML file.
   - Any other type of HTML-based publishing - Set the `args.hdf` parameter to point to your XML file.
4. Run the transformation scenario.

Result: The equation should now be properly rendered in other browsers, such as Edge, IE, or Chrome.

How to Disable Caching in WebHelp Responsive Output

In cases where a set of WebHelp Responsive pages need to be updated on a regular basis to deliver the latest version of the documentation, the WebHelp pages should always be requested from the server upon reloading it in a web browser on the client side, (rather than re-using an outdated cached version in the browser).

To disable caching in WebHelp Responsive output, follow this procedure:
1. Create a new well-formed XML file and add the following code snippet:

```xml
<meta http-equiv="Pragma" content="no-cache" />
<meta http-equiv="Expires" content="-1" />
```

**Note:**
The code should look like this:

```xml
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
  <meta http-equiv="Pragma" content="no-cache" />
  <meta http-equiv="Expires" content="-1" />
</head>
</html>
```

2. Edit the DITA Map WebHelp Responsive transformation scenario and open the Parameters tab.
3. Edit the value of the `webhelp.fragment.head` parameter and set it to the absolute path of your XML file.
4. Click OK to save the changes to the transformation scenario.
5. Run the transformation scenario.

**Result:** Your additional content is included at the end of the `<head>` element of your output document.

## How to Configure a Custom Search Engine

It is possible to integrate a custom search engine into your WebHelp Responsive output. This is done by using the following transformation parameters:

### `webhelp.fragment.custom.search.engine.results`

This parameter can be used to replace the search results area with custom XHTML content. The value of the parameter is the path to an XHTML file that contains your custom content.

### `webhelp.fragment.custom.search.engine.script`

This parameter can be used to replace WebHelp's built-in search engine with your own custom search engine. The value of the parameter is the path to an XHTML file that contains the scripts required for your custom search engine to run.

To integrate a custom search engine into your WebHelp Responsive output, follow these steps:

1. If you have not already created a Publishing Template, see How to Create a Publishing Template (on page 1791).
2. Create the following items in the folder that contains your publishing descriptor file (the `.opt` file):
   - A file named `custom-search-results-fragment.xml`.
   - A file named `custom-search-script-fragment.xml`.
   - A folder named `js`. 
3. In the `custom-search-results-fragment.xml` file, define the HTML structure that will be used as the search results area. For example:

```xml
<div id="custom-search-results">...</div>
```

**Note:**

The custom search engine script will need to find an HTML element from the HTML structure that will be used as the search results area and write the search results inside it. In this example, it is the `<div>` element with the id `custom-search-results`.

4. In the `js` folder, create a file named `custom-search.js`.

5. As a starting point, you can copy the following content to the `custom-search.js` file:

```javascript
$(document).ready(function () {
    const params = new URLSearchParams(window.location.search);
    const searchQuery = params.get('searchQuery');
    // Implement your custom search engine
    // Display the search results
});
```

**Important:**

The value entered by the user in the search page will be available in the URL’s query parameters in a parameter named `searchQuery`.

**Attention:**

`URLSearchParams` is not supported on all browsers (it is used as an example). A list with the supported browsers can be found [here](#). A different solution should be used if you need to support other browsers.

6. Implement your custom search engine.

**Note:**

The search results should be pushed into the `<div>` element created earlier with the id `custom-search-results`.

7. In the `custom-search-script-fragment.xml` file, define the scripts that are required for your custom search engine to run. For example:

```xml
<div>
    <script src="${oxygen-webhelp-template-dir}/js/custom-search.js"></script>
</div>
```
8. Copy the js folder to the output folder during the transformation process. For this, open the .opt file and add the following content in the `<resources>` section (see Template Resources (on page 1610) for more details):

```xml
<fileset>
    <include name="js/**"/>
</fileset>
```

9. Set the transformation parameters needed to enable the custom search engine. For this, open the .opt file and add the following content inside the `<webhelp>` element:

```xml
<html-fragments>
    <fragment file="custom-search-script-fragment.xml" placeholder="webhelp.fragment.custom.search.engine.script"/>
    <fragment file="custom-search-results-fragment.xml" placeholder="webhelp.fragment.custom.search.engine.results"/>
</html-fragments>
```

10. Run the transformation with this publishing template selected.

**Note:**
A publishing template containing all the steps we described, except for the actual custom search engine implementation is available to download [here](#). You can use it as a starting point.

### How to Add a Link to PDF Documentation

It is possible to add a component in your WebHelp output that links to an external PDF resource. For example, it could link to the PDF equivalent of the documentation. This is achieved by configuring some transformation parameters and the link component is added in the header/breadcrumb stripe, next to the navigation links.

The transformation parameters used for generating a PDF link component in the WebHelp Responsive output are:

- **webhelp.pdf.link.url**
  Specifies the target URL for the PDF link component.

- **webhelp.pdf.link.text**
  Specifies the text for the PDF link component.

- **webhelp.pdf.link.icon.path**
  Specifies the path or URL of the image icon to be used for the PDF link component. If not specified, a default icon is used.

- **webhelp.show.pdf.link**
  Specifies whether or not the PDF link component is shown in the WebHelp Responsive output. Allowed values are: **yes** (default) and **no**.

- **webhelp.pdf.link.anchor.enabled**
Specifies whether or not the current topic ID should be appended as the name destination at the end of the PDF link. Allowed values are: yes (default) and no.

How to Add a Custom Component for WebHelp Output

This topic explains how to use several customization methods to define and implement a custom component for WebHelp output pages.

Predefined components

The WebHelp output is based on a set of HTML Page Layout Files (on page 1622) that define the default layout of the generated pages. Each layout file is made of a set of various components. Each component is described using an associated XML element that is processed at the generation time resulting in its associated component being included in the output pages.

Here are a few examples of predefined components: Logo, Title, Menu, Search Input, Topics Tiles, Topic Breadcrumb, Topic Content, Publication Table of Contents. A complete list with all the available components is available here: Layout of the Responsive Page Types (on page 1560).

For example, the page component that is used to define the Search Input field in the WebHelp HTML pages is defined as follows:

```xml
<!-- Search form -->
<whc:webhelp_search_input class="navbar-form wh_topic_page_search search" role="form"/>

At publishing time, the above component will be expanded into:

```xml
<div class="wh_search_input navbar-form wh_topic_page_search search">
  <form id="searchForm" method="get" role="search" action="../search.html">
    <div>
      <input type="search" placeholder="Search " class="wh_search_textfield ui-autocomplete-input" id="textToSearch" name="searchQuery" aria-label="Search query" required="required" autocomplete="off"/>
      <button type="submit" class="wh_search_button aria-label="Search">
        <span class="search_input_text">Search</span>
      </button>
    </div>
  </form>
</div>
```

Customization Methods

The most common customization methods for the WebHelp Responsive output include:
• Apply custom CSS styles (on page 1652) to change the default layout and styles.
• Insert additional HTML content (on page 1654) using one of the available HTML Fragment Placeholder parameters (on page 1613).
• Extend the default processing using XSLT Extension Points (on page 1613).
• Configure available Transformation Parameters (on page 1718).

Use Case: Custom Link Component

For the subsequent procedure, suppose you have a DITA project for a User Manual and you also have various video demonstrations available on your website that supplement the documentation. You may want to link a video demonstration for a particular feature it its associated DITA topic in the WebHelp output.

You could simply add a link somewhere in your DITA topic, but this approach would not be very suitable for a printable (PDF) version of your User Manual. Thus, you need to include the link to the associated video demonstration only in the WebHelp output of your User Manual (and not the PDF version).

One way to link a video with its associated topic is to include its URL in the metadata section. For example:

```xml
<prolog>
  <metadata>
    <othermeta name="video-link" content="https://www.youtube.com/watch?v=zNmXfKWXwO8"/>
  </metadata>
</prolog>
```

Next, you need to instruct WebHelp to pick up the URL from the metadata and generate a link in a specific location of the HTML output page. You can achieve this by creating your own WebHelp custom component.

Creating a Custom Component

You can combine several of the available customization methods to define and implement your own WebHelp custom component.

**Figure 503. Custom Component**

<table>
<thead>
<tr>
<th>HTML Fragment</th>
<th>XSLT Extension</th>
<th>CSS</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>&lt;custom-component/&gt;</code></td>
<td><code>&lt;xsl:template match=&quot;custom-component&quot;&gt;</code></td>
<td><code>.custom-component { display: block; }</code></td>
</tr>
</tbody>
</table>

To create a custom component that displays a link to the current topic's associated video tutorial, follow these steps:

1. Define your component. For example, it may have the following form:

   ```xml
   <comp:video-link xmlns:comp="http://example.com/custom-components"/>
   ```

   The component is an XML element that belongs to a custom defined namespace.
2. Insert the component in your topic pages. To do this, you will have to save the associated XML element in an HTML Fragment file (for example, named `video-link-fragment.xml`).

3. Reference the HTML Fragment file in your current Publishing Template’s descriptor file (on page 1607) and associate it with an HTML Fragment placeholder that is available for the topic pages (`webhelp.fragment.before.topic.toolbar` in this case):

   ```xml
   <html-fragments>
     <fragment file="component/html-fragment/video-link-fragment.xml" placeholder="webhelp.fragment.before.topic.toolbar"/>
   </html-fragments>
   
   **Note:**
   The HTML Fragment file is referenced using a path relative to the Publishing Template root directory.

4. Create a custom XSLT file that processes the custom component and picks up the video URL available in the current topic's metadata and generates a link to the page that contains the video:

   ```xml
   <xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
                    xmlns:xs="http://www.w3.org/2001/XMLSchema"
                    xmlns:comp="http://example.com/custom-components"
                    exclude-result-prefixes="xs comp"
                    version="3.0">

   <!-- Custom component implementation -->
   <xsl:template match="comp:video-link" mode="copy_template">
     <xsl:param name="ditaot_topicContent" tunnel="yes"/>
     <!-- Look for a 'video-link' <meta> element in the current topic content -->
     <xsl:variable name="videoLinkMeta" select="$ditaot_topicContent//*:meta[@name='video-link']"/>
     <xsl:if test="exists($videoLinkMeta)">
       <div class="video-link-container">
         <a href="{$videoLinkMeta[1]/@content}" class="video-link" target="_blank" aria-label="Video">
           <span>Video</span>
         </a>
       </div>
     </xsl:if>
   </xsl:template>

   </xsl:stylesheet>
   ```

The HTML content generated for your component will look like this:
5. Reference the above XSL file in your Publishing Template's descriptor file using the XSLT extension point associated with the XSL module that generates an HTML file for each DITA topic:

```xml
<xslt>
  <extension file="component/xsl/video-link-impl.xsl" id="com.oxygenxml.webhelp.xsl.dita2webhelp"/>
</xslt>
```

6. Create a custom CSS file that contains the rules for styling the output for your component:

```css
@import url('https://fonts.googleapis.com/icon?family=Material+Icons');

.video-link-container {
    display: flex;
    align-items: center;
    flex-grow: 10;
    justify-content: flex-end;
}

.video-link {
    display: flex;
    align-items: center;
    color: #fff !important;
}

.video-link:before {
    content: "smart_display";
    font-family: 'Material Icons';
    font-size: 20px;
    display: inline-block;
    word-wrap: normal;
    white-space: nowrap;
}

.video-link span {
    display: none;
}
```
7. Reference the above CSS file in your Publishing Template's descriptor file:

```xml
<resources>
    <!-- .... -->
    <css file="component/css/video-link.css"/>
</resources>
```

**Result:** An icon that is a link to the video appears in the header stripe in the output page.

**Figure 504. Custom Link to Video Component**

![Figure showing a custom link to video component](image)

**Sample Publishing Template**

A sample Publishing Template that contains all the above customizations is available here: [https://github.com/oxygenxml/oxygen-publishing-template-samples/tree/master/templates/video-link-custom-component](https://github.com/oxygenxml/oxygen-publishing-template-samples/tree/master/templates/video-link-custom-component).

**How to Group Related Links by Type**

By default, all links from DITA relationship tables or related link elements within topics are grouped under one "Related information" heading:

<table>
<thead>
<tr>
<th>Related information</th>
</tr>
</thead>
<tbody>
<tr>
<td>Target Topic</td>
</tr>
<tr>
<td>Target Concept</td>
</tr>
<tr>
<td>Target Task</td>
</tr>
</tbody>
</table>

It is possible to group the links by target type (topic type) by setting the `webhelp.rellinks.group.mode=group-by-type` parameter. The output will look like this:

<table>
<thead>
<tr>
<th>Related concepts</th>
</tr>
</thead>
<tbody>
<tr>
<td>Target Concept</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Related tasks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Target Task</td>
</tr>
</tbody>
</table>
Related information

Target Topic

WebHelp Responsive Transformation Parameters

In addition to the common DITA-OT transformation parameters and the HTML-based Output Parameters, there are numerous other supported parameters that are specific to the WebHelp Responsive output.

Publishing Template Parameters

`webhelp.publishing.template`

Specifies the path to the ZIP archive (or root folder) that contains your custom WebHelp Responsive template.

**Note:**
The built-in templates are stored in the `DITA-OT-DIR/plugins/com.oxygenxml.webhelp.responsive/templates` folder.

**Note:**
Relative paths are resolved based on the current working directory.

`webhelp.publishing.template.descriptor`

Specifies the name of the descriptor to be loaded from the WebHelp Responsive template package. If it is not specified, the first encountered descriptor will be automatically loaded.

Custom Resource Parameters

`webhelp.custom.resources`

The file path to a directory that contains resources files. All files from this directory will be copied to the root of the WebHelp output.

`webhelp.favicon`

The file path that points to an image to be used as a favicon in the WebHelp output.

`webhelp.logo.image.target.url`

Specifies a target URL that is set on the logo image. When you click the logo image, you will be redirected to this address.

`webhelp.logo.image`

Specifies a path to an image displayed as a logo in the left side of the output header.

`webhelp.logo.image.alt`
Specifies a value that will be set in the `@alt` attribute of the logo image. If the parameter is not specified, the `@alt` attribute will contain the publication title. Note that this parameter makes sense only in conjunction with the `webhelp.logo.image` parameter.

**Oxygen Feedback Parameter**

`webhelp.fragment.feedback`

You can integrate Oxygen Feedback with your WebHelp Responsive output to provide a comments area at the bottom of each page where readers can offer feedback. When you create an Oxygen Feedback site configuration, an HTML fragment is generated during the final step of the creation process and that fragment should be set as the value for this parameter.

**HTML Fragment Extension Parameters**

`webhelp.enable.html.fragments.cleanup`

Enables or disables the automatic conversion of HTML fragments to well-formed XML. If set to `true` (default), the transformation automatically converts non-well-formed HTML content to a well-formed XML equivalent. If set to `false`, the transformation will fail if at least one HTML fragment is not well-formed.

`webhelp.fragment.after.body`

This parameter can be used to display a given XHTML fragment after the body in all types of pages. The value of the parameter can be either a well-formed XHTML fragment or a path to a file that contains a well-formed XHTML fragment.

`webhelp.fragment.after.body.main.page`

This parameter can be used to display a given XHTML fragment after the body in the main page. The value of the parameter can be either a well-formed XHTML fragment or a path to a file that contains a well-formed XHTML fragment.

`webhelp.fragment.after.body.search.page`

This parameter can be used to display a given XHTML fragment after the body in the search results page. The value of the parameter can be either a well-formed XHTML fragment or a path to a file that contains a well-formed XHTML fragment.

`webhelp.fragment.after.body.terms.page`

This parameter can be used to display a given XHTML fragment after the body in the index terms page. The value of the parameter can be either a well-formed XHTML fragment or a path to a file that contains a well-formed XHTML fragment.

`webhelp.fragment.after.body.topic.page`

This parameter can be used to display a given XHTML fragment after the body in the topic page. The value of the parameter can be either a well-formed XHTML fragment or a path to a file that contains a well-formed XHTML fragment.

`webhelp.fragment.after.feedback`
This parameter can be used to display a given XHTML fragment after the Oxygen Feedback commenting component in the topic page. The value of the parameter can be either a well-formed XHTML fragment or a path to a file that contains a well-formed XHTML fragment.

webhelp.fragment.after.header

This parameter can be used to display a given XHTML fragment after the header section in all types of pages. The value of the parameter can be either a well-formed XHTML fragment or a path to a file that contains a well-formed XHTML fragment.

webhelp.fragment.after.header.main.page

This parameter can be used to display a given XHTML fragment after the header section in the main page. The value of the parameter can be either a well-formed XHTML fragment or a path to a file that contains a well-formed XHTML fragment.

webhelp.fragment.after.header.search.page

This parameter can be used to display a given XHTML fragment after the header section in the search results page. The value of the parameter can be either a well-formed XHTML fragment or a path to a file that contains a well-formed XHTML fragment.

webhelp.fragment.after.header.terms.page

This parameter can be used to display a given XHTML fragment after the header section in the index terms page. The value of the parameter can be either a well-formed XHTML fragment or a path to a file that contains a well-formed XHTML fragment.

webhelp.fragment.after.header.topic.page

This parameter can be used to display a given XHTML fragment after the header section in the topic page. The value of the parameter can be either a well-formed XHTML fragment or a path to a file that contains a well-formed XHTML fragment.

webhelp.fragment.after.logo_and_title

This parameter can be used to display a given XHTML fragment after the logo and title in all types of pages. The value of the parameter can be either a well-formed XHTML fragment or a path to a file that contains a well-formed XHTML fragment.

webhelp.fragment.after.main.content.area

This parameter can be used to display a given XHTML fragment after the main content section in all types of pages. The value of the parameter can be either a well-formed XHTML fragment or a path to a file that contains a well-formed XHTML fragment.

webhelp.fragment.after.main.content.area.main.page

This parameter can be used to display a given XHTML fragment after the main content section in the main page. The value of the parameter can be either a well-formed XHTML fragment or a path to a file that contains a well-formed XHTML fragment.

webhelp.fragment.after.main.content.area.topic.page
This parameter can be used to display a given XHTML fragment after the main content section in the topic page. The value of the parameter can be either a well-formed XHTML fragment or a path to a file that contains a well-formed XHTML fragment.

**webhelp.fragment.after.main.page.search (deprecated)**

This parameter is deprecated. Use [webhelp.fragment.after.search.input.main.page](#) instead.

**webhelp.fragment.after.publication.toc**

This parameter can be used to display a given XHTML fragment before the publication's table of contents component in the topic page. The value of the parameter can be either a well-formed XHTML fragment or a path to a file that contains a well-formed XHTML fragment.

**webhelp.fragment.after.search.input**

This parameter can be used to display a given XHTML fragment after the search field in all types of pages. The value of the parameter can be either a well-formed XHTML fragment or a path to a file that contains a well-formed XHTML fragment.

**webhelp.fragment.after.search.input.main.page**

This parameter can be used to display a given XHTML fragment after the search field in all the main page. The value of the parameter can be either a well-formed XHTML fragment or a path to a file that contains a well-formed XHTML fragment.

**webhelp.fragment.after.search.input.search.page**

This parameter can be used to display a given XHTML fragment after the search field in all the search results page. The value of the parameter can be either a well-formed XHTML fragment or a path to a file that contains a well-formed XHTML fragment.

**webhelp.fragment.after.search.input.terms.page**

This parameter can be used to display a given XHTML fragment after the search field in all the index terms page. The value of the parameter can be either a well-formed XHTML fragment or a path to a file that contains a well-formed XHTML fragment.

**webhelp.fragment.after.search.input.topic.page**

This parameter can be used to display a given XHTML fragment after the search field in all the topic page. The value of the parameter can be either a well-formed XHTML fragment or a path to a file that contains a well-formed XHTML fragment.

**webhelp.fragment.after.toc_or_tiles**

This parameter can be used to display a given XHTML fragment after the table of contents or tiles in the main page. The value of the parameter can be either a well-formed XHTML fragment or a path to a file that contains a well-formed XHTML fragment.

**webhelp.fragment.after.top_menu**

This parameter can be used to display a given XHTML fragment after the top menu in all types of pages. The value of the parameter can be either a well-formed XHTML fragment or a path to a file that contains a well-formed XHTML fragment.
**webhelp.fragment.after.topic.breadcrumb**

This parameter can be used to display a given XHTML fragment after the breadcrumb component in the topic page. The value of the parameter can be either a *well-formed* XHTML fragment or a path to a file that contains a *well-formed* XHTML fragment.

**webhelp.fragment.after.topic.content**

This parameter can be used to display a given XHTML fragment after the topic's content in the topic page. The value of the parameter can be either a *well-formed* XHTML fragment or a path to a file that contains a *well-formed* XHTML fragment.

**webhelp.fragment.after.topic.toc**

This parameter can be used to display a given XHTML fragment after the topic's table of contents component in the topic page. The value of the parameter can be either a *well-formed* XHTML fragment or a path to a file that contains a *well-formed* XHTML fragment.

**webhelp.fragment.after.topic.toolbar**

This parameter can be used to display a given XHTML fragment after the toolbar buttons above the topic content in the topic page. The value of the parameter can be either a *well-formed* XHTML fragment or a path to a file that contains a *well-formed* XHTML fragment.

**webhelp.fragment.before.body**

This parameter can be used to display a given XHTML fragment before the page body in all types of pages. The value of the parameter can be either a *well-formed* XHTML fragment or a path to a file that contains a *well-formed* XHTML fragment.

**webhelp.fragment.before.body.main.page**

This parameter can be used to display a given XHTML fragment before the page body in the main page. The value of the parameter can be either a *well-formed* XHTML fragment or a path to a file that contains a *well-formed* XHTML fragment.

**webhelp.fragment.before.body.search.page**

This parameter can be used to display a given XHTML fragment before the page body in the search results page. The value of the parameter can be either a *well-formed* XHTML fragment or a path to a file that contains a *well-formed* XHTML fragment.

**webhelp.fragment.before.body.terms.page**

This parameter can be used to display a given XHTML fragment before the page body in the index terms page. The value of the parameter can be either a *well-formed* XHTML fragment or a path to a file that contains a *well-formed* XHTML fragment.

**webhelp.fragment.before.body.topic.page**

This parameter can be used to display a given XHTML fragment before the page body in the topic page. The value of the parameter can be either a *well-formed* XHTML fragment or a path to a file that contains a *well-formed* XHTML fragment.

**webhelp.fragment.before.feedback**
This parameter can be used to display a given XHTML fragment before the Oxygen Feedback commenting component in the topic page. The value of the parameter can be either a well-formed XHTML fragment or a path to a file that contains a well-formed XHTML fragment.

**webhelp.fragment.before.logo_and_title**

This parameter can be used to display a given XHTML fragment before the logo and title. The value of the parameter can be either a well-formed XHTML fragment or a path to a file that contains a well-formed XHTML fragment.

**webhelp.fragment.before.main.content.area**

This parameter can be used to display a given XHTML fragment before the main content section in all types of pages. The value of the parameter can be either a well-formed XHTML fragment or a path to a file that contains a well-formed XHTML fragment.

**webhelp.fragment.before.main.content.area.main.page**

This parameter can be used to display a given XHTML fragment before the main content section in the main page. The value of the parameter can be either a well-formed XHTML fragment or a path to a file that contains a well-formed XHTML fragment.

**webhelp.fragment.before.main.content.area.search.page**

This parameter can be used to display a given XHTML fragment before the main content section in the search results page. The value of the parameter can be either a well-formed XHTML fragment or a path to a file that contains a well-formed XHTML fragment.

**webhelp.fragment.before.main.content.area.terms.page**

This parameter can be used to display a given XHTML fragment before the main content section in the index terms page. The value of the parameter can be either a well-formed XHTML fragment or a path to a file that contains a well-formed XHTML fragment.

**webhelp.fragment.before.main.content.area.topic.page**

This parameter can be used to display a given XHTML fragment before the main content section in the topic page. The value of the parameter can be either a well-formed XHTML fragment or a path to a file that contains a well-formed XHTML fragment.

**webhelp.fragment.before.main.page.search (deprecated)**

This parameter is deprecated. Use `webhelp.fragment.before.search.input.main.page` instead.

**webhelp.fragment.before.publication.toc**

This parameter can be used to display a given XHTML fragment before the publication's table of contents component in the topic page. The value of the parameter can be either a well-formed XHTML fragment or a path to a file that contains a well-formed XHTML fragment.

**webhelp.fragment.before.search.input**

This parameter can be used to display a given XHTML fragment before the search field in all types of pages. The value of the parameter can be either a well-formed XHTML fragment or a path to a file that contains a well-formed XHTML fragment.
**webhelp.fragment.before.search.input.main.page**

This parameter can be used to display a given XHTML fragment before the search field in the main page. The value of the parameter can be either a **well-formed** XHTML fragment or a path to a file that contains a **well-formed** XHTML fragment.

**webhelp.fragment.before.search.input.search.page**

This parameter can be used to display a given XHTML fragment before the search field in the search results page. The value of the parameter can be either a **well-formed** XHTML fragment or a path to a file that contains a **well-formed** XHTML fragment.

**webhelp.fragment.before.search.input.terms.page**

This parameter can be used to display a given XHTML fragment before the search field in the index terms page. The value of the parameter can be either a **well-formed** XHTML fragment or a path to a file that contains a **well-formed** XHTML fragment.

**webhelp.fragment.before.search.input.topic.page**

This parameter can be used to display a given XHTML fragment before the search field in the topic page. The value of the parameter can be either a **well-formed** XHTML fragment or a path to a file that contains a **well-formed** XHTML fragment.

**webhelp.fragment.before.toc_or_tiles**

This parameter can be used to display a given XHTML fragment before the table of contents or tiles in the main page. The value of the parameter can be either a **well-formed** XHTML fragment or a path to a file that contains a **well-formed** XHTML fragment.

**webhelp.fragment.before.top_menu**

This parameter can be used to display a given XHTML fragment before the top menu in all types of pages. The value of the parameter can be either a **well-formed** XHTML fragment or a path to a file that contains a **well-formed** XHTML fragment.

**webhelp.fragment.before.topic.breadcrumb**

This parameter can be used to display a given XHTML fragment before the breadcrumb component in the topic page. The value of the parameter can be either a **well-formed** XHTML fragment or a path to a file that contains a **well-formed** XHTML fragment.

**webhelp.fragment.before.topic.content**

This parameter can be used to display a given XHTML fragment before the topic's content in the topic page. The value of the parameter can be either a **well-formed** XHTML fragment or a path to a file that contains a **well-formed** XHTML fragment.

**webhelp.fragment.before.topic.toc**

This parameter can be used to display a given XHTML fragment before the topic's table of contents component in the topic page. The value of the parameter can be either a **well-formed** XHTML fragment or a path to a file that contains a **well-formed** XHTML fragment.

**webhelp.fragment.before.topic.toolbar**
This parameter can be used to display a given XHTML fragment before the toolbar buttons above the topic content in the topic page. The value of the parameter can be either a well-formed XHTML fragment or a path to a file that contains a well-formed XHTML fragment.

**webhelp.fragment.custom.search.engine.results**

This parameter can be used to replace the search results area with custom XHTML content. The value of the parameter is the path to an XHTML file that contains your custom content.

**webhelp.fragment.custom.search.engine.script**

This parameter can be used to replace WebHelp's built-in search engine with your own custom search engine. The value of the parameter is the path to an XHTML file that contains the scripts required for your custom search engine to run.

**webhelp.fragment.footer**

This parameter can be used to display a given XHTML fragment as the page footer in all types of pages. The value of the parameter can be either a well-formed XHTML fragment or a path to a file that contains a well-formed XHTML fragment.

⚠️ **Important:**

This parameter should only be used if you are using a valid, purchased license of Oxygen XML Editor (do not use it with a trial license).

**webhelp.fragment.head**

This parameter can be used to display a given XHTML fragment in the header section in all types of pages. The value of the parameter can be either a well-formed XHTML fragment or a path to a file that contains a well-formed XHTML fragment.

**webhelp.fragment.head.main.page**

This parameter can be used to display a given XHTML fragment in the header section in the main page. The value of the parameter can be either a well-formed XHTML fragment or a path to a file that contains a well-formed XHTML fragment.

**webhelp.fragment.head.search.page**

This parameter can be used to display a given XHTML fragment in the header section in the search results page. The value of the parameter can be either a well-formed XHTML fragment or a path to a file that contains a well-formed XHTML fragment.

**webhelp.fragment.head.terms.page**

This parameter can be used to display a given XHTML fragment in the header section in the index terms page. The value of the parameter can be either a well-formed XHTML fragment or a path to a file that contains a well-formed XHTML fragment.

**webhelp.fragment.head.topic.page**
This parameter can be used to display a given XHTML fragment in the header section in the topic page. The value of the parameter can be either a well-formed XHTML fragment or a path to a file that contains a well-formed XHTML fragment.

**webhelp.fragment.welcome**

This parameter can be used to display a given XHTML fragment as a welcome message (or title). The value of the parameter can be either a well-formed XHTML fragment or a path to a file that contains a well-formed XHTML fragment.

### Output Component Parameters

**webhelp.default.collection.type.sequence**

Specifies if the sequence value will be used by default when the `@collection-type` attribute is not specified. This option is helpful if you want to have *Next* and *Previous* navigational buttons generated for all HTML pages. Allowed values are no (default) and yes.

**webhelp.enable.sticky.header**

Controls whether or not the header section will remain sticky in the output. Possible values are yes (default) or no.

**webhelp.enable.sticky.publication.toc**

Controls whether or not the publication table of contents will remain sticky in the output. Possible values are yes (default) or no.

**webhelp.enable.sticky.topic.toc**

Controls whether or not the topic table of contents will remain sticky in the output. Possible values are yes (default) or no.

**webhelp.figure.title.placement**

Controls the placement of the title for figures (relative to the image). Possible values include top (default) and bottom.

**webhelp.merge.nested.topics.related.links**

Specifies if the related links from nested topics will be merged with the links in the parent topic. Thus the links will be moved from the topic content to the related links component and all of the links from the same group (for example, *Related Tasks, Related References, Related Information*) are merged into a single group. The default value is yes.

**webhelp.publication.toc.hide.chunked.topics**

Specifies if the table of contents will contain links for chunked topics. The default value is yes.

**webhelp.publication.toc.links**

Specifies which links will be included in the table of contents. The possible values are:
• **chapter** (default) - The TOC will include links for the current topic, its children, its siblings, and its direct ancestor (including the direct ancestor’s siblings), and the parent chapter.
• **topic** - The TOC will only include links for the current topic and its direct children.
• **all** - The TOC will include all links.

**webhelp.publication.toc.tooltip.position**

By default, if a topic contains a `<shortdesc>` element, its content is displayed in a tooltip when the user hovers over its link in the table of contents. This parameter controls whether or not this tooltip is displayed and its position relative to the link. The possible values are:

• **left**
• **right** (default)
• **top**
• **bottom**
• **hidden** - The tooltip will not be displayed.

**webhelp.rellinks.group.mode**

Specifies the related links grouping mode. All links can be grouped into a single “Related Information” heading or links can be grouped by their target type (topic, task, or concept).
Allowed values: **single-group** (default) or **group-by-type**.

**webhelp.show.breadcrumb**

Specifies if the breadcrumb component will be presented in the output. The default value is **yes**.

**webhelp.show.changes.and.comments**

When set to **yes**, user comments, replies to comments, and tracked changes are published in the WebHelp output. The default value is **no**.

**webhelp.show.child.links**

Specifies if child links will be generated in the output for all topics that have subtopics. The default value is **no**.

**webhelp.show.full.size.image**

Specifies if responsive images that are displayed with a smaller dimension than their original size can be clicked to see an enlarged version of the image. The default value is **yes**.

**webhelp.show.indexterms.link**

Specifies if an icon that links to the index will be presented in the output. The default value is **yes**.

**webhelp.show.main.page.tiles**

Specifies if the tiles component will be presented in the main page of the output. For a tree style layout, this parameter should be set to **no**.

**webhelp.show.main.page.toc**
Oxygen XML Editor 24.1 | 11 - Publishing | 1728

Specifies if the table of contents will be presented in the main page of the output. The default value is yes.

**webhelp.show.navigation.links**

Specifies if navigation links will be presented in the output. The default value is yes.

**webhelp.show.print.link**

Specifies if a print link or icon will be presented within each topic in the output. The default value is yes.

**webhelp.show.publication.toc**

Specifies if a table of contents will be presented on the left side of each topic in the output. The default value is yes.

**webhelp.show.topic.toc**

Specifies if a topic table of contents will be presented on the right side of each topic in the output. This table of contents contains links to each section within the current topic that contains an gid attribute and the section corresponding to the current scroll position is highlighted. The default value is yes.

**webhelp.show.top.menu**

Specifies if a menu will be presented at the topic of the main page in the output. The default value is yes.

**webhelp.table.title.placement**

Controls the placement of the title for tables. Possible values include top (default) and bottom.

**webhelp.top.menu.activated.on.click**

When this parameter is activated (set to yes), clicking an item in the top menu will expand the submenu (if available). You can then click on a submenu item to open the item (topic). You can click outside the menu or press ESC to hide the menu. When set to no (default), hovering over a menu item displays the menu content.

**webhelp.top.menu.depth**

Specifies the maximum depth level of the topics that will be included in the top menu. The default value is 3. A value of 0 means that the menu has unlimited depth.

**webhelp.topic.collapsible.elements.initial.state**

Specifies the initial state of collapsible elements (tables with titles, nested topics with titles, sections with titles, index term groups). The possible values are collapsed or expanded (default value).

**Search-Related Parameters**

**webhelp.enable.search.autocomplete**

Specifies if the Autocomplete feature is enabled in the WebHelp search text field. The default value is yes.
**webhelp.google.search.results**

A file path that specifies the location of a well-formed XHTML file containing the Google Custom Search Engine element `<gcse:searchresults-only>`. You can use all supported attributes for this element. It is recommended to set the `@linkTarget` attribute to `frm` for frameless (`iframe`) version of WebHelp or to `contentWin` for the frameset version of WebHelp. The default value for this attribute is `_blank` and the search results will be loaded in a new window. If this parameter is not specified, the following code will be used:

```xml
<gcse:searchresults-only linkTarget="frm"></gcse:searchresults-only>
```

**webhelp.google.search.script**

A file path that specifies the location of a well-formed XHTML file containing the Custom Search Engine script from Google.

**webhelp.search.enable.pagination**

Specifies whether or not search results will be displayed on multiple pages. Allowed values are `yes` or `no`.

**webhelp.search.index.elements.to.exclude**

Specifies a list of HTML elements that will not be indexed by the search engine. The value of the `@class` attribute can be used to exclude specific HTML elements from indexing. For example, the `div.not-indexed` value will not index all `<div>` elements that have a `@class` attribute with the value of `not-indexed`. Use a comma separator to specify more than one element.

**webhelp.search.japanese.dictionary**

The file path of the dictionary that will be used by the *Kuromoji* morphological engine for indexing Japanese content in the WebHelp pages. The encoding for the dictionary must be UTF8.

**webhelp.search.page.numberOfItems**

Specifies the number of search results items displayed on each page. This parameter is only used when the `webhelp.search.enable.pagination` parameter is enabled.

**webhelp.search.ranking**

If this parameter is set to `false` then the 5-star rating mechanism is no longer included in the search results that are displayed on the Search tab (default setting is `true`).

**webhelp.search.stop.words.exclude**

Specifies a list of words that will be excluded from the default list of *stop words* that are filtered out before the search processing. Use comma separators to specify more than one word (for example: `if,for,is`).

**webhelp.search.stop.words.include**

Specifies a list of words that will be ignored by the search engine. Use a comma separator to specify more than one word.

**webhelp.sitemap.base.url**
Base URL for all the `<loc>` elements in the generated `sitemap.xml` file. If this parameter is specified, the `<loc>` element will contain the value of this parameter plus the relative path to the page. If this parameter is not specified, the `<loc>` element will only contain the relative path of the page (the relative file path from the `@href` attribute of a `<topicref>` element from the DITA map, appended to this base URL value).

**webhelp.sitemap.change.frequency**

The value of the `<changefreq>` element in the generated `sitemap.xml` file. The `<changefreq>` element is optional in `sitemap.xml`. If you leave this parameter set to its default empty value, then the `<changefreq>` element is not added in `sitemap.xml`. Allowed values: `<empty string>` (default), `always`, `hourly`, `daily`, `weekly`, `monthly`, `yearly`, `never`.

**webhelp.sitemap.priority**

The value of the `<priority>` element in the generated `sitemap.xml` file. It can be set to any fractional number between 0.0 (least important priority) and 1.0 (most important priority). For example, 0.3, 0.5, or 0.8. The `<priority>` element is optional in `sitemap.xml`. If you leave this parameter set to its default empty value, then the `<priority>` element is not added in `sitemap.xml`.

**Publishing Speedup Parameters**

**parallel**

A common parameter with other transformation types. When set to `true` (default value is `false`), the publishing pre-processing stages are run in parallel slightly improving the publishing time.

**store-type**

A common parameter with other transformation types. When set to `memory`, the processing stages use internal memory to store temporarily processed documents, thus decreasing the publishing time but slightly increasing the amount of internal memory used for the process. When publishing on Windows, setting this parameter can decrease the publishing times by about one-third.

**Note:**

The `fix.external.refs.com.oxygenxml` parameter is not supported when running the transformation from a command line. This parameter is normally used to specify whether or not the application tries to fix such references in a temporary files folder before the DITA Open Toolkit is invoked on the fixed references.

**Parameters for Adding a Link to PDF Documentation in WebHelp Responsive Output**

The following transformation parameters can be used to generate a PDF link component in the WebHelp Responsive output (for example, it could link to the PDF equivalent of the documentation):

**webhelp.pdf.link.url**
Specifies the target URL for the PDF link component.

**webhelp.pdf.link.text**

Specifies the text for the PDF link component.

**webhelp.pdf.link.icon.path**

Specifies the path or URL of the image icon to be used for the PDF link component. If not specified, a default icon is used.

**webhelp.pdf.link.anchor.enabled**

Specifies whether or not the current topic ID should be appended as the name destination at the end of the PDF link. Allowed values are: **yes** (default) and **no**.

**webhelp.show.pdf.link**

Specifies whether or not the PDF link component is shown in the WebHelp Responsive output. Allowed values are: **yes** (default) and **no**.

---

### Related information

- Generating WebHelp Responsive Output *(on page 1639)*
- Setting DITA-OT Parameters

---

### WebHelp Responsive XSLT-Import and XSLT-Parameter Extension Points

XSLT extension points can be used from either from an Oxygen Publishing Template or from a DITA-OT extension plug-in.

#### Extension Points from an Oxygen Publishing Template

The publishing template allows you to specify an XSLT extension point. The extension point will only affect the transformations that use the particular template.

⚠️ **Important:**

While the publishing templates only support referencing one extension point at a time, you can use `xslt:include` or `xslt:import` to aggregate multiple modules.

For a specific example of how to use an extension in a publishing template, see: How to Use an XSLT Extension Point from a Publishing Template *(on page 1696)* topic.

**Example:**

```xml
<publishing-template>
...
<webhelp>
...
<xslt>
```
Extension Points from a DITA-OT Extension Plug-in

The DITA-OT plug-in installer adds an XSLT import statement in the default WebHelp XSLT so that the XSLT stylesheets referenced by the extension point becomes part of the normal build. You can use these extension points to override XSLT processing steps.

Example:

```xml
<extension
    id="com.oxygenxml.webhelp.xsl.createMainPage"
    file="xsl/customMainPage.xsl"/>
</xslt>
```

**XSLT-Import Extension Points**

The following extension points are supported:

- **com.oxygenxml.webhelp.xsl.dita2webhelp**
  Extension point to override the XSLT stylesheet (`dita2webhelp.xsl`) that produces an HTML file for each DITA topic. The location of this file is `DITA-OT-DIR\plugins\com.oxygenxml.webhelpresponsive\xsl\dita2webhelp\dita2webhelp.xsl`

- **com.oxygenxml.webhelp.xsl.createMainPage**
  Extension point to override the XSLT stylesheet (`createMainPage.xsl`) that produces the WebHelp Responsive main HTML page (`index.html`). The location of this file is `DITA-OT-DIR\plugins\com.oxygenxml.webhelpresponsive\xsl\mainFiles\createMainPage.xsl`

- **com.oxygenxml.webhelp.xsl.createNavLinks**
  Extension point to override the XSLT stylesheets that are used to generate navigation links in the WebHelp Responsive pages. These stylesheets can be found in the `navLinks` folder: `DITA-OT-DIR\plugins\com.oxygenxml.webhelpresponsive\xsl\navLinks`.

- **com.oxygenxml.webhelp.xsl.createSearchPage**
  Extension point to override the XSLT stylesheet (`createSearchPage.xsl`) that produces the WebHelp Responsive search HTML page (`search.html`). The location of this file is `DITA-OT-DIR\plugins\com.oxygenxml.webhelpresponsive\xsl\mainFiles\createSearchPage.xsl`

- **com.oxygenxml.webhelp.xsl.createIndexTermsPage**
Extension point to override the XSLT stylesheet (`createIndextermsPage.xsl`) that produces the WebHelp Responsive index terms HTML page (`indexterms.html`). The location of this file is `DITA-OT-DIR\plugins\com.oxygenxml.webhelp.responsive\xsl\mainFiles\createIndextermsPage.xsl`.

`com.oxygenxml.webhelp.xsl.createTocXML`  
Extension point to override the XSLT stylesheet (`tocDita.xsl`) that produces the `toc.xml` file. This file contains information extracted from the DITA map (on page 3218) and it is mainly used to construct the WebHelp Table of Contents and navigational links. The path to this stylesheet is: `DITA-OT-DIR\plugins\com.oxygenxml.webhelp.responsive\xsl\navLinks\tocDita.xsl`.

**XSLT-Parameter Extension Points**

If your customization stylesheet declares one or more XSLT parameters and you want to control their values from the transformation scenario, you can use one of the following XSLT parameter extension points:

- `com.oxygenxml.webhelp.xsl.dita2webhelp.param`  
  Use this extension point to pass parameters to the stylesheet specified using the `com.oxygenxml.webhelp.xsl.dita2webhelp` extension point (on page 1732).

- `com.oxygenxml.webhelp.xsl.createMainPage.param`  
  Use this extension point to pass parameters to the stylesheet specified using the `com.oxygenxml.webhelp.xsl.createMainPage` extension point (on page 1732).

- `com.oxygenxml.webhelp.xsl.createNavLinks.param`  
  Use this extension point to pass parameters to the stylesheet specified using the `com.oxygenxml.webhelp.xsl.createNavLinks` extension point (on page 1732).

- `com.oxygenxml.webhelp.xsl.createSearchPage.param`  
  Use this extension point to pass parameters to the stylesheet specified using the `com.oxygenxml.webhelp.xsl.createSearchPage` extension point (on page 1732).

- `com.oxygenxml.webhelp.xsl.createIndexTermsPage.param`  
  Use this extension point to pass parameters to the stylesheet specified using the `com.oxygenxml.webhelp.xsl.createIndexTermsPage` extension point (on page 1732).

- `com.oxygenxml.webhelp.xsl.createTocXML.param`  
  Use this extension point to pass parameters to the stylesheet specified using the `com.oxygenxml.webhelp.xsl.createTocXML` extension point (on page 1733).

Related Information:

[DITA-OT] XSLT-Import Extension Points  
[DITA-OT] XSLT-Parameter Extension Points
WebHelp Classic Output for DocBook

The **WebHelp Classic** variant is designed for desktop systems when feedback from users is not necessary and it is available for DocBook. You can also integrate a feedback system that provides the ability for your users to add comments in the output. This section contains information about configuring a WebHelp Classic system and customizing the output.

This type of WebHelp system can be generated by using the [DocBook WebHelp Classic](on page 1458) transformation scenario.

WebHelp Classic Output Layout and Features

Layout of the WebHelp Classic System Interface

The layout of the **WebHelp Classic** system consists of the following components:

**Left Pane or Frame**

This section on the left side of the help system includes the following tabs:

- **Content**
  
  A typical table of contents style presentation of your content. You can use the **Expand all/Collapse all** buttons to expand or collapse all the topics presented in the Table of Contents.

  ![Note:](on page 1734)

  You can enhance the appearance of items in the *Table of Contents*. See the [Customizing WebHelp Classic Output chapter](on page 1741) for more details.

- **Index**
  
  Presents the index terms for your content. If your content does not contain any `<indexterm>` elements, this tab is not generated.

- **Search Results**
  
  This tab is generated when the **Search** field is used. It presents the search results in the form of links to topics where the search terms are found, along with a rating scheme for each result. For more details, see the [Search Feature section](on page 1736).

**Upper Pane or Frame**

The upper section of the help system includes the following features:

- **Search Field**
Use this feature to perform searches in your content. When you enter search terms in this field, the results are displayed in the Search Results tab in the left section of the help system, along with a rating scheme for each result. For more details, see the Search Feature section (on page 1736).

**Frames Option**
Click on this option to display the output rendered in HTML frames.

**Print Option**
Opens a dialog box with various printing options and a print preview.

**Navigation Links**
You can navigate through the content of your output using the navigation links or arrows in the upper-right part of the page. These arrows allow you to move to the Parent topic, Previous topic, or Next topic. Links to the parent topics of the currently open topic are also presented at the top of the page.

**Tip:**
To hide the Parent, Next, and Previous links, you can edit the transformation scenario and set the value of the args.hide.parent.link parameter to yes.

**Main Pane or Frame**
The content of the help pages are rendered and displayed in this main section.
WebHelp Classic Search Engine

Search Rules

Rules that are applied during a search include:

- You can use quotes to perform an exact search for multiple word phrases (for example, "grow flowers" will only return results if both words are found consecutively and exactly as they are typed in the search field). This type of search is known as a **phrase search**.

- **Boolean Search** is supported using the following operators: **and**, **or**, **not**. When there are two adjacent search terms without an operator, **or** is used as the default search operator (for example, **grow flowers** is the same as **grow or flowers**).

- The space character separates keywords (an expression such as **grow flowers** counts as two separate keywords.

- Words composed by merging two or more words with colon (":"), minus ("-"), underline ("_"), or dot ("." ) characters count as a single word.

- Your search terms should contain two or more characters (note that stop words will be ignored). This rule does not apply to CJK (Chinese, Japanese, Korean) languages.
• When searching for multi-word phrases in CJK (Chinese, Japanese, Korean) languages that often have multiple words appear in strings without a space separator, you need to add a space to separate the words. Otherwise, WebHelp will not find results. For example, Chinese uses a specialized character for space separators, but the current WebHelp implementation cannot detect such specialized characters, so to search for 开始之前 (it translates as "before you begin" or "before start"), you have to enter 开始之前 (notice the space between the second and third symbols) in the search field.

5-Star Rating Mechanism and Sorting

The Search feature is also enhanced with a rating mechanism that computes scores for every result that matches the search criteria. These scores are then translated into a 5-star rating scheme and the stars are displayed to the right of each result. The search results are sorted depending on the following:

• Search entries that satisfy the phrase search criterion are presented first.
• The number of keywords found in a single page (the higher the number, the better).
• The context (for example, a word found in a title, scores better than a word found in unformatted text).

The search ranking order, sorted by relevance is as follows:

◦ The search term is included in a meta keyword.
◦ The search term is in the title of the page.
◦ The search term is in bold text in a paragraph.
◦ The search term is in normal text in a paragraph.

Excluded Terms

To improve performance, the Search feature excludes certain stop words. For example, the English version of the stop words includes: a, an, and, are, as, at, be, but, by, for, if, in, into, is, it, no, not, of, on, or, such, that, the, their, then, there, these, they, this, to, was, will, with.

WebHelp Classic Search Results Tab

When you enter search terms in the Search field at the top of the help system, the results are displayed in the Search Results tab in the left section. When you click on a result in the Search Results tab, that result is displayed in the main pane with the search terms highlighted. If you press Enter with the Search field empty, the highlights are removed.
Missing Terms

If you enter multiple search terms (other than stop words), for any result that the search engine found at least one term but not one or more of the other terms, the Missing terms will be listed below each result.

Tag Element Scoring Values

HTML tag elements are also assigned a scoring value and these values are evaluated for the search results. For information about editing these values, see How to Change Element Scoring in Search Results (on page 1755).

Browser Compatibility

This output format is compatible with the most recent versions of the following common browsers:
Important:
Due to some security restrictions in certain browsers (Google Chrome), WebHelp Classic pages loaded from the local system (through URLs of the file://... format) may not work properly. It is recommended that you load WebHelp Classic pages in Google Chrome only from a web server (with a URL such as http://your.server.com/webhelp/index.html or http://localhost/web_pages/index.html).

Warning:
Due to some restrictions in web browsers regarding JavaScript code, the frameless version (index.html start page) of the WebHelp Classic system should only be loaded from a web server (with a URL such as http://your.server.com/webhelp/index.html or http://localhost/web_pages/index.html). When loading WebHelp Classic pages from the local file system, the frameset version (index_frames.html start page) of the WebHelp Classic system should be used instead (file://...).

Related information
Deploying the Oxygen Feedback Comments Component for DocBook (on page 1740)

Generating WebHelp Classic Output for DocBook

The publishing process can be initiated from a transformation scenario within Oxygen XML Editor/Author or from a command-line tool outside Oxygen XML Editor/Author.

Running from Oxygen XML Editor/Author

To publish DocBook content to WebHelp Classic output from a transformation scenario inside Oxygen XML Editor/Author, use one of the following procedures, depending on whether or not you want a feedback section in your output.

WebHelp Classic Output

To publish a DocBook document as a WebHelp Classic system, follow these steps:

1. Click the Configure Transformation Scenario(s) action from the toolbar (or the Document > Transformation menu).
2. Select the DocBook WebHelp Classic scenario from the DocBook 4 or DocBook 5 section.
3. Click Apply associated.
When the DocBook WebHelp Classic transformation is complete, the output is automatically opened in your default browser.

**Automating the WebHelp Classic Output for DocBook**

DocBook-based WebHelp output can be generated from an automated publishing process using a command-line tool outside of Oxygen XML Editor/Author. However, to do this, you must purchase an additional Oxygen XML WebHelp license.

**Deploying the Oxygen Feedback Comments Component for DocBook**

You can add a comments component in your WebHelp Classic output to provide a simple and efficient way for your community to interact and offer feedback. The comments component is contributed by Oxygen Feedback, a modern comment management system that can be integrated with your WebHelp Classic output to provide a comments area at the bottom of each WebHelp page where readers can add new comments or reply to existing ones.

Oxygen Feedback includes a modern, user-friendly administration interface where you can moderate comments, manage users, view statistics, and configure settings. It is very easy to integrate and there are no requirements for installing additional software.

An add-on is also available that contributes a Feedback Comments Manager view in Oxygen XML Editor/Author where the documentation team can see all the comments added in your WebHelp output. This means they can react to user feedback by making corrections and updating the source content without leaving the application.

**Adding the Feedback System to WebHelp Classic Documentation**

**Prerequisite**

To install and manage Oxygen Feedback, you must obtain a license for the product. This requires that you choose a subscription plan during the installation procedure. To see the subscription plans prior to installing the product, go to: https://www.oxygenxml.com/oxygen_feedback/buy_feedback.html.

**Installation Procedure**

1. Log in to your Feedback account from the administration login page (https://feedback.oxygenxml.com/login). You can click Log in with Google or Log in with Facebook to create an account using your Google or Facebook credentials, or click the Sign Up tab to create an account using your name and email address.
2. Click the Add site button to create a site configuration. If you have not already selected a subscription plan, you will be directed to a page where you can choose from several options.
3. In the Settings page, enter a Name and Description for the site configuration. There are some optional settings that can be adjusted according to your needs. For more details, see the Site Settings topic. Click Continue.
4. In the Initial version page, enter the Base URL for your website (you can add additional URLs by clicking the Add button). You can also specify an Initial version if you want it to be something other than 1.0. If you do not plan to have multiple versions, leave the version as 1.0. For more details, see the Initial Version topic. Click Continue.

5. In the Installation page, choose a site generation option:
   a. If you will generate the documentation using a transformation scenario in Oxygen XML Editor/Author, select the Oxygen XML Editor option and continue with these steps:
      i. Copy the generated HTML fragment and click Finish.
      ii. Create an XML file (for example, feedback-install.xml) with the generated installation fragment.
      iii. In Oxygen XML Editor/Author, open the Configure Transformation Scenario(s) dialog box.
      iv. Select and duplicate the DocBook WebHelp Classic scenario (on page 1458).
      v. Go to the Parameters tab.
      vi. Set the webhelp.footer.file parameter to reference the path of the fragment file created earlier.
   b. If you will generate the documentation using a command-line script, select the Oxygen XML WebHelp option and continue with these steps:
      i. Copy the generated HTML fragment and click Finish.
      ii. Create an XML file (for example, feedback-install.xml) with the generated installation fragment.
      iii. Use the webhelp.footer.file parameter in your command-line script to specify the path to the file you created. For example:

         ```
         docbook.bat -Dwebhelp.footer.file=c:\path\to\feedback-install.xml
         ```

6. [Optional] If you want the Oxygen Feedback comments component to fill the entire page width, contribute a custom CSS file (use the html.stylesheet parameter to reference it) that contains the following style rule:

   ```
   div.footer {
     float: none;
   }
   ```

For more details about Oxygen Feedback, how to configure settings, moderate comments, view statistics, and much more, see the Oxygen Feedback user guide.

### Customizing WebHelp Classic Output

Oxygen XML WebHelp provides support for customizing the WebHelp Classic output to suit your specific needs. The WebHelp Classic type of output is designed for desktop systems and features a familiar tri-pane layout. You can use this system to publish DocBook documents. You can also integrate a feedback system to allow your users to add comments to your output.

To change the overall appearance of the WebHelp Classic output, you can use the visual WebHelp Skin Builder tool (on page 1742), which does not require knowledge of CSS language. If you are familiar with CSS and
coding, you can style your WebHelp output through your own custom stylesheets. You can also customize your output by modifying option and parameters in the transformation scenario.

This section includes topics that explain various ways to customize your WebHelp system output, such as how to improve the appearance of the Table of Contents, add logo images in the title area, integrate with social media, add custom headers and footers, and much more.

**Changing the Layout and Styles**

This section contains some topics that explain how to customize the layout and style of your WebHelp Classic output using custom CSS, inserting custom HTML content, and more.

**WebHelp Skin Builder**

The **WebHelp Skin Builder** is a simple, easy-to-use tool, specially designed to assist users to visually customize the look and feel of the WebHelp output. It is implemented as an online tool hosted on the Oxygen XML website and allows you to experiment with various styles and colors over a documentation sample.

To be able to use the **Skin Builder**, you need:

- An Internet connection and unrestricted access to Oxygen XML website.
- A late version web browser.

To start the **Skin Builder**, use a web browser to go to [https://www.oxygenxml.com/webhelp-skin-builder](https://www.oxygenxml.com/webhelp-skin-builder).

**Skin Builder Layout**

The left side panel of the **Skin Builder** is divided into 3 sections:

- **Actions** - Contains the following two buttons:
  - **Import** - Opens an Import CSS dialog box that allows you to load a CSS stylesheet and apply it over the documentation sample.
  - **Export** - Saves all properties as a CSS file.

- **Settings** - Includes a Highlight selection option that helps you identify the areas affected by a particular element customization.
  - When hovering an item in the customizable elements menu, the affected sample area is highlighted with a dotted blue border.
  - When an item in the customizable elements menu is selected, the affected sample area is highlighted with a solid red border.

- **Customize** - Provides a series of customizable elements organized under four main categories:
  - Header
  - TOC Area
  - Vertical Splitter
  - Content

For each customizable element, you can alter properties such as background color or font face. Any alteration made in the customizable elements menu is applied in real time over the sample area.
Creating a Customization Skin

1. You can start with one of the built-in skins or a CSS stylesheet applied over the sample using the Import button.
2. Use the elements in the Customize section to set properties that modify the look of the skin. By default, all customizable elements display a single property, but you can make more visible by clicking the Add button and choosing from the available properties.

Note: If you want to revert a particular property to its initial value, click the Reset button.

3. When you are happy with the skin customizations you have made, click the Export button. All settings will be saved in a CSS file.

Apply a Customization Skin to a DocBook to WebHelp Classic Transformation Scenario

1. Start Oxygen XML Editor.
2. Load the DocBook file you want to produce as a WebHelp output.
3. In the Parameters tab, set the webhelp.skin.css parameter to point to the previously exported CSS.
4. To customize the logo, use the following parameters: webhelp.logo.image and webhelp.logo.image.target.url.
5. Run the transformation to obtain the WebHelp output.

For more information about using the WebHelp Skin Builder, watch our video demonstration:

https://www.youtube.com/embed/32PGX--PQx0

How to Use CSS Styling to Customize WebHelp Output

The most common way to customize the look and style of your WebHelp output is to use custom CSS styling. This method can be used to make small, simple styling changes or more advanced, precise changes. To implement the styling in your WebHelp output, you simply need to create the custom CSS file and reference it in your transformation scenario or script. This custom file will be the final CSS to be applied so its content will override the styles in the other pre-existing CSS files.

Using the CSS Inspector to Identify Content for Custom CSS File

You can use your browser's CSS inspector to identify the pertinent code in the current CSS files and you can even make changes directly in the CSS inspector to test the results so that you know exactly what content to use in your custom CSS file.

In most popular browsers (such as Chrome, Firefox, and Edge), you can access the CSS inspector by using F12 or by selecting Inspect Element (or simply Inspect) from the contextual menu.
Tip:
When using Safari on macOS, you must first enable the Develop menu by going to the Advanced settings and selecting Show Develop menu in menu bar. Then you can select Show Web Inspector from the Develop menu or click Command + Option + I.

Referencing the Custom CSS Using Oxygen XML Editor/Author

To use a custom CSS to style WebHelp output and use a transformation scenario from within Oxygen XML Editor/Author, follow this procedure:

1. Create your custom CSS file.
2. Edit the WebHelp transformation scenario and open the Parameters tab. Set the html.stylesheet parameter to the path of your custom CSS file.
3. Run the WebHelp transformation scenario to generate the output.

Referencing the Custom CSS Using a Script Outside of Oxygen XML Editor/Author

Important:
Running WebHelp transformations from a script outside of Oxygen XML Editor/Author requires an additional license and some additional setup:

- You must have a valid license for the Oxygen XML WebHelp Plugin (https://www.oxygenxml.com/buy_webhelp.html).
- The Oxygen XML WebHelp Plugin must be installed and integrated.

To use a custom CSS to style WebHelp output and use a script outside of Oxygen XML Editor/Author (on page 1739), follow this procedure:

1. Create your custom CSS file.
2. Reference your custom CSS file. Use the html.stylesheet parameter in your transformation script and set its value to the path of your custom CSS file.
3. Execute the transformation script.

How to Add Custom HTML Content in WebHelp Classic Output

You can add custom HTML content in the WebHelp Classic output by inserting it in a well-formed XML file that will be referenced in the transformation. This content may include references to additional JavaScript, CSS, and other types of resources, or such resources can be inserted inline within the HTML content that is inserted in the XML file.
Using Oxygen XML Editor/Author

To include custom HTML content in the WebHelp Classic output using a transformation scenario from within Oxygen XML Editor/Author, follow this procedure:

1. Insert the HTML content in a well-formed XML file considering the following notes:

   ◦ **Well-Formedness** - If the content of the file is not *XML Well-formed (on page 773)*, (or fragments are not well-formed), the transformation will fail.

   A common use case is if you want to include several `<script>` or `<link>` elements. In this case, the XML fragment has multiple root elements and to make it well-formed, you can wrap it in an `<html>` element. This element tag will be filtered out and only its children will be copied to the output documents. Similarly, you can wrap your content in `<head>`, `<body>`, `<html/head>`, or `<html/body>` elements.

   ◦ **Referencing Resources in the XML File** - You can include references to local resources (such as JavaScript or CSS files) by using the built-in `${oxygen-webhelp-output-dir}` macro to specify their paths relative to the output directory:

   ```xml
   <html>
   <script type="text/javascript" src="${oxygen-webhelp-output-dir}/js/test.js"/>
   <link rel="stylesheet" type="text/css"
         href="${oxygen-webhelp-output-dir}/css/test.css" />
   </html>
   ```

   To copy the referenced resources to the output directory, follow the procedure in: *How to Copy Additional Resources to Output Directory (on page 1753).*

   ◦ **Inline JavaScript or CSS Content:**

   **JavaScript:**

   ```javascript
   /* Include JavaScript code here. */

   function myFunction() {
     return true;
   }
   ```

   **CSS:**

   ```css
   /* Include CSS style rules here. */

   * { color:red
   ```
2. Edit the WebHelp Classic transformation scenario.
3. Go to the Parameters tab.
4. Edit the value of the `webhelp.head.script` parameter and set it to reference the URL of the XML file created in step 1. Your additional content will be included at the end of the `head` element of your output document.

![Note:](image)

If you have special characters (for example, `&`, `<`) that break the well-formedness of the XML fragment, it is important to place the content inside an XML comment.

**[Important]** XML comment tags (both the start and end tags) must be on lines by themselves. If they are on the same line as any of the script's content, it will likely result in a JavaScript error.

```javascript
<script type="text/javascript">
  <!--
  /* Include JavaScript code here. */

  function myFunction() {
    return true;
  }
  //-->
</script>
```

5. Click **OK** to save the changes and run the transformation scenario.

**Using a Script Outside of Oxygen XML Editor/Author**

![Important:](image)

Running WebHelp transformations from a script outside of Oxygen XML Editor/Author requires an additional license and some additional setup:
You must have a valid license for the Oxygen XML WebHelp Plugin (https://www.oxygenxml.com/buy_webhelp.html).

The Oxygen XML WebHelp Plugin must be installed and integrated.

To include custom HTML content in the WebHelp Classic output using a script outside of Oxygen XML Editor/Author (on page 1739), follow this procedure:

1. Insert the HTML content in a well-formed XML file considering the following notes:
   - **Well-Formedness** - If the content of the file is not XML Well-formed (on page 773), (or fragments are not well-formed), the transformation will fail.

     A common use case is if you want to include several `<script>` or `<link>` elements. In this case, the XML fragment has multiple root elements and to make it well-formed, you can wrap it in an `<html>` element. This element tag will be filtered out and only its children will be copied to the output documents. Similarly, you can wrap your content in `<head>`, `<body>`, `<html/head>`, or `<html/body>` elements.

   - **Referencing Resources in the XML File** - You can include references to local resources (such as JavaScript or CSS files) by using the built-in `${oxygen-webhelp-output-dir}` macro to specify their paths relative to the output directory:

     ```html
     <html>
     <script type="text/javascript" src="${oxygen-webhelp-output-dir}/js/test.js" />
     <link rel="stylesheet" type="text/css" href="${oxygen-webhelp-output-dir}/css/test.css" />
     </html>
     ```

     To copy the referenced resources to the output directory, follow the procedure in: How to Copy Additional Resources to Output Directory (on page 1753).

   - **Inline JavaScript or CSS Content**:

     **JavaScript**:

     ```javascript
     /* Include JavaScript code here. */
     
     function myFunction() {
         return true;
     }
     ```

     **CSS**: 
Note:

If you have special characters (for example, &, <) that break the well-formedness of the XML fragment, it is important to place the content inside an XML comment.

[Important] XML comment tags (both the start and end tags) must be on lines by themselves. If they are on the same line as any of the script's content, it will likely result in a JavaScript error.

```html
<script type="text/javascript">
<!--
/* Include JavaScript code here. */

function myFunction() {
    return true;
}
-->
</script>
```

2. Use the `webhelp.head.script` parameter in your transformation script and set its value to reference the URL of the XML file created in step 1. Your additional content will be included at the end of the `head` element of your output document.

Note:

If you want to include the content in the `body` element, use the `webhelp.body.script` parameter instead.

3. Execute the transformation script.

Related Information:

How to Copy Additional Resources to Output Directory (on page 1753)
How to Change Number Styles for Ordered Lists

Ordered lists (\texttt{\textless{ol}\textgreater}) are usually numbered in XHTML output using numerals. If you want to change the numbering to alphabetical, follow these steps:

1. Define a custom \texttt{outputclass} value and set it as an attribute of the ordered list, as in the following example:

\begin{verbatim}
<ol outputclass="number-alpha">
    <li>A</li>
    <li>B</li>
    <li>C</li>
</ol>
\end{verbatim}

2. Add the following code snippet in a custom CSS file:

\begin{verbatim}
ol.number-alpha{
    list-style-type: lower-alpha;
}
\end{verbatim}

3. Edit the WebHelp transformation scenario and open the Parameters tab. Set the \texttt{html.stylesheet} parameter to the path of your custom CSS file.

4. Run the transformation scenario.

How to Change the Icons in a WebHelp Classic Table of Contents

You can change the icons that appear in a WebHelp Classic table of contents by assigning new image files in a custom CSS file. By default, these icons are defined with the following CSS codes (the first example is the icon that appears for a collapsed menu and the second for an expanded menu):

\begin{verbatim}
.hasSubMenuClosed{
    background: url('../img/book_closed16.png') no-repeat;
    padding-left: 16px;
    cursor: pointer;
}
\end{verbatim}

\begin{verbatim}
.hasSubMenuOpened{
    background: url('../img/book_opened16.png') no-repeat;
    padding-left: 16px;
    cursor: pointer;
}
\end{verbatim}

Using Oxygen XML Editor/Author

To assign other icons and use a transformation scenario from within Oxygen XML Editor/Author, follow this procedure:
1. Create a custom CSS file that assigns your desired icons to the `.hasSubMenuClosed` and `.hasSubMenuOpened` properties.

```css
.hasSubMenuClosed{
    background: url('TOC-my-closed-button.png') no-repeat;
}

.hasSubMenuOpened{
    background: url('TOC-my-opened-button.png') no-repeat;
}
```

2. It is recommended that you store the image files in the same directory as the default icons (`[OXYGEN_INSTALL_DIR]\frameworks\docbook\xsl\com.oxygenxml.webhelp.classic\oxygen-webhelp\resources\img\`).

3. Edit the WebHelp transformation scenario and open the Parameters tab. Set the `html.stylesheet` parameter to the path of your custom CSS file.

4. Run the WebHelp transformation scenario to generate the output.

**Using a Script Outside of Oxygen XML Editor/Author**

⚠️ **Important:**
Running WebHelp transformations from a script outside of Oxygen XML Editor/Author requires an additional license and some additional setup:

- You must have a valid license for the Oxygen XML WebHelp Plugin (https://www.oxygenxml.com/buy_webhelp.html).
- The Oxygen XML WebHelp Plugin must be installed and integrated.

To assign other icons and use a script outside of Oxygen XML Editor/Author (on page 1739), follow this procedure:

1. Create a custom CSS file that assigns your desired icons to the `.hasSubMenuClosed` and `.hasSubMenuOpened` properties.

```css
.hasSubMenuClosed{
    background: url('TOC-my-closed-button.png') no-repeat;
}

.hasSubMenuOpened{
    background: url('TOC-my-opened-button.png') no-repeat;
}
```

2. It is recommended that you store the image files in the same directory as the default icons (`[DocBook XSL directory]\com.oxygenxml.webhelp.classic\oxygen-webhelp\resources\img\`).
3. Reference your custom CSS file. Use the `html.stylesheet` parameter in your transformation script and set its value to the path of your custom CSS file.

4. Execute the transformation script.

**How to Customize the Appearance of Selected Items in the Table of Contents**

The appearance of selected items in the table of contents of WebHelp Classic output can be enhanced.

For example, to highlight the background of the selected item, follow these steps:

1. Locate the `toc.css` file in the following directory: `[DocBook XSL directory]\com.oxygenxml.webhelp.classic\oxygen-webhelp\resources\css`.
2. Edit that CSS file, find the `menuItemSelected` class, and change the value of the `background` property.
3. Run the transformation.

---

**Note:**

You can also overwrite the same value from your own custom CSS and then specify the path to your CSS in the transformation scenario by using the `html.stylesheet` parameter and set its value to the path of your custom CSS file.

**Adding Graphics and Media Resources**

This section contains topics that explain how to add media resources to the published WebHelp Class output or to the output directory.

**How to Add a Favicon in WebHelp Systems**

You can add a custom favicon to your WebHelp output by simply using a parameter in the transformation scenario to point to your favicon image. This is available for DocBook WebHelp output using the WebHelp Classic transformation.

**Using Oxygen XML Editor/Author**

To add a favicon to your WebHelp output using a transformation scenario from within Oxygen XML Editor/Author, follow this procedure:

1. Edit the WebHelp transformation scenario and open the Parameters tab.
2. Locate the `webhelp.favicon` parameter and enter the file path that points to the image that will be use as the favicon.
3. Run the transformation scenario.

**Result:** Browsers that provide favicon support will display the favicon (typically in the browser's address bar, in the list of bookmarks, and in the history).
Using a Script Outside of Oxygen XML Editor/Author

**Important:**
Running WebHelp transformations from a script outside of Oxygen XML Editor/Author requires an additional license and some additional setup:

- You must have a valid license for the Oxygen XML WebHelp Plugin (https://www.oxygenxml.com/buy_webhelp.html).
- The Oxygen XML WebHelp Plugin must be installed and integrated.

To add a `favicon` to your WebHelp output using a script outside of Oxygen XML Editor/Author, follow this procedure:

1. Specify the file path that points to the image that will be use as the `favicon` using the `webhelp.favicon` parameter.
2. Execute the transformation script.

**Result:** Browsers that provide `favicon` support will display the `favicon` (typically in the browser’s address bar, in the list of bookmarks, and in the history).

How to Add a Logo Image in the Title Area

You can customize WebHelp Classic output to include a logo in the title area. It will be displayed before the publication title. You can also specify a URL that can be used to send users to a specific website when they click the logo image.

This customization can be done using a transformation scenario from within Oxygen XML Editor/Author or using a command-line script outside of Oxygen XML Editor/Author.

Using Oxygen XML Editor/Author

To add a logo in the title area of your WebHelp output using a transformation scenario from within Oxygen XML Editor/Author, follow this procedure:

1. Edit a WebHelp Classic transformation scenario, then open the Parameters tab.
2. Specify the path to your logo in the `webhelp.logo.image` parameter.
3. If you also want to add a link to your website when you click the logo image, set the URL in the `webhelp.logo.image.target.url` parameter.
4. Run the transformation scenario.
Using a Script Outside of Oxygen XML Editor/Author

Important:
Running WebHelp transformations from a script outside of Oxygen XML Editor/Author requires an additional license and some additional setup:

- You must have a valid license for the Oxygen XML WebHelp Plugin (https://www.oxygenxml.com/buy_webhelp.html).
- The Oxygen XML WebHelp Plugin must be installed and integrated.

To add a logo in the title area of your WebHelp output using a script outside of Oxygen XML Editor/Author, follow this procedure:

1. Specify the path to your logo using the `webhelp.logo.image` parameter.
2. If you also want to add a link to your website when you click the logo image, set the URL using the `webhelp.logo.image.target.url` parameter.
3. Execute the transformation script.

How to Add Videos in DocBook WebHelp Classic Output

You can insert references to videos in your DocBook topics and then publish them to WebHelp Classic output. The videos can be played directly in all HTML5-based outputs, including WebHelp systems.

To add videos in the WebHelp Classic output generated from DocBook documents, follow these steps:

1. Edit the DocBook document and reference the video using a `<mediaobject>` element, as in the following example:

```xml
<mediaobject>
  <videobject>
    <videodata fileref="http://www.youtube.com/watch?v/VideoName"/>
  </videobject>
</mediaobject>
```

2. Apply a WebHelp transformation scenario to obtain the output.

How to Copy Additional Resources to Output Directory

You can copy additional resources (such as JavaScript, CSS or other resources) to the output directory of a WebHelp system by using the `webhelp.custom.resources` parameter.

Using Oxygen XML Editor/Author

To copy additional resources to the output directory using a transformation scenario from within Oxygen XML Editor/Author, follow this procedure:
1. Place all your resources in the same directory.
2. Edit the WebHelp transformation scenario, then open the **Parameters** tab.
3. Edit the value for the `webhelp.custom.resources` parameter and set it to the absolute path of the directory in step 1.
4. Click **OK** to save the changes to the transformation scenario.
5. Run the transformation scenario.

**Result:** All files from the new directory will be copied to the root of the WebHelp output directory.

### Using a Script Outside of Oxygen XML Editor/Author

**Important:**

Running WebHelp transformations from a script outside of Oxygen XML Editor/Author requires an additional license and some additional setup:

- You must have a valid license for the Oxygen XML WebHelp Plugin (https://www.oxygenxml.com/buy_webhelp.html).
- The Oxygen XML WebHelp Plugin must be installed and integrated.

To copy additional resources to the output directory using a script outside of Oxygen XML Editor/Author (on page 1739), follow this procedure:

1. Place all your resources in the same directory.
2. Specify the absolute path to that directory using the `webhelp.custom.resources` parameter.
3. Execute the transformation script.

**Result:** All files from the new directory will be copied to the root of the WebHelp output directory.

### How to Add MathML Equations in WebHelp Output

Currently, only Firefox can render MathML equations embedded in the HTML code. MathJax is a solution to properly view MathML equations embedded in HTML content in a variety of browsers.

If you have DocBook content that has embedded MathML equations and you want to properly view the equations in published HTML output types (such as WebHelp), you need to add a reference to the MathJax script in the **head** element of all HTML files that have the equation embedded.

For example:

```html
<script type="text/javascript" src="https://cdnjs.cloudflare.com/ajax/libs/mathjax/2.7.1/MathJax.js?config=TeX-AMS-MML_HTMLorMM L"></script>
```
**Result:** The equation should now be properly rendered in the WebHelp output for other browsers.

**Searching the Output**

This section contains topics that explain how to customize some of the search features in WebHelp Classic output.

**How to Change Element Scoring in Search Results**

The WebHelp Search feature is enhanced with a rating mechanism that computes scores for every page that matches the search criteria. HTML tag elements are assigned a scoring value and these values are evaluated for the search results. The WebHelp directory includes a properties file that defines the scoring values for tag elements and this file can be edited to customize the values according to your needs.

To edit the scoring values of HTML tag element for enhancing WebHelp search results, follow these steps:

1. Edit the scoring properties file for DocBook WebHelp systems ([DocBook XSL directory]\com.oxygenxml.webhelp.classic\indexer\scoring.properties). The properties file includes instructions and examples to help you with your customization.

   The values that can be edited in the `scoring.properties` file:

   ```
   h1 = 10
   h2 = 9
   h3 = 8
   h4 = 7
   h5 = 6
   h6 = 5
   b = 5
   strong = 5
   em = 3
   i=3
   u=3
   div.toc=-10
   title=20
   div.ignore=ignored
   meta_keywords = 20
   meta_indexterms = 20
   meta_description = 25
   shortdesc=25
   ```

2. Save your changes to the file.

3. Re-run your WebHelp transformation.
How to Index Japanese Content in WebHelp Classic

To optimize the indexing of Japanese content in WebHelp pages, the *Lucene Kuromoji Japanese analyzer* can be used. This analyzer is included in the *Oxygen XML Editor/Author* installation kit.

**Using Oxygen XML Editor/Author**

To activate the Japanese indexing in your WebHelp output using a transformation scenario from within *Oxygen XML Editor/Author*, follow this procedure:

1. Set the language for your content to Japanese. Edit a *DocBook to WebHelp* transformation scenario and in the **Parameters** tab, set the value of the `l10n.gentext.default.language` parameter to `ja`.
2. Run the WebHelp transformation scenario to generate the output.

**Using a Script Outside of Oxygen XML Editor/Author**

**Important:**

Running WebHelp transformations from a script outside of *Oxygen XML Editor/Author* requires an additional license and some additional setup:

- You must have a valid license for the *Oxygen XML WebHelp Plugin* ([https://www.oxygenxml.com/buy_webhelp.html](https://www.oxygenxml.com/buy_webhelp.html)).
- The *Oxygen XML WebHelp Plugin* must be installed and integrated.

To activate the Japanese indexing in your WebHelp output using a script outside of *Oxygen XML Editor/Author*, follow this procedure:

1. Set the language for your content to Japanese. Use the `l10n.gentext.default.language` parameter in your transformation script and set its value to `ja`.
2. Execute the transformation script.

**Related information**

*How to Localize the Interface of DocBook to WebHelp Classic Output (on page 1756)*

**Localization in WebHelp Classic Output**

This section contains topics that explain the localization support for DocBook WebHelp Classic transformations.

**How to Localize the Interface of DocBook to WebHelp Classic Output**

Static labels that are used in the WebHelp output are kept in translation files in the *[DocBook XSL directory]/com.oxygenxml.webhelp.classic/oxygen-webhelp/resources/localization*
folder. Translation files have the `strings-lang1-lang2.xml` name format, where `lang1` and `lang2` are ISO language codes. For example, the US English text is kept in the `strings-en-us.xml` file.

To localize the interface of the WebHelp output for DocBook transformations, follow these steps:

1. Look for the `strings-[lang1]-[lang2].xml` file in `[DocBook XSL directory]/com.oxygenxml.webhelp.classic/oxygen-webhelp/resources/localization` directory (for example, the Canadian French file would be: `strings-fr-ca.xml`). If it does not exist, create one starting from the `strings-en-us.xml` file.

2. Translate all the labels from the above language file. Labels are stored in XML elements that have the following format: `<str name="Label name">Caption</str>`.

3. Make sure that the new XML file that you created in the previous two steps is listed in the file `[DocBook XSL directory]/com.oxygenxml.webhelp.classic/oxygen-webhelp/resources/localization/strings.xml`. For example, a Canadian French file would be listed as: `<lang xml:lang="fr-ca" filename="strings-fr-ca.xml">`.

4. Edit any of the DocBook to WebHelp transformation scenarios (with or without feedback) and set the `l10n.gentext.default.language` parameter to the code of the language you want to localize (for example, `fr-ca` for Canadian French).

5. Run the transformation scenario to produce the WebHelp output.

Related Information:

- How to Index Japanese Content in WebHelp Classic ([on page 1756])
- How to Activate Support for Right-to-Left (RTL) Languages
- Integrating Social Media and Google Tools in the WebHelp Classic Output
- How to Add a Facebook Like Button in WebHelp Classic Output

How to Activate Support for Right-to-Left (RTL) Languages

To activate support for RTL (right-to-left) languages in WebHelp output, set the `@xml:lang` attribute with the corresponding attribute value:

- `ar-eg` - Arabic
- `he-il` - Hebrew
- `ur-pk` - Urdu

Integrating Social Media and Google Tools in the WebHelp Classic Output

Oxygen XML Editor includes support for integrating some of the most popular social media sites in WebHelp output.

How to Add a Facebook Like Button in WebHelp Classic Output

It is possible to integrate Facebook™ into your WebHelp Classic output and the widget will appear in the footer sections of your WebHelp page.
Using Oxygen XML Editor/Author

To add a Facebook™ Like widget to your WebHelp output using a transformation scenario from within Oxygen XML Editor/Author, follow this procedure:

1. Go to the Facebook Developers website.
2. Fill-in the displayed form, then click the Get Code button.
3. Copy the two code snippets and paste them into a `<div>` element inside an XML file called `facebook-widget.xml`. Make sure you follow these rules:
   ◦ The file must be well-formed.
   ◦ The code for each `<script>` element must be included in an XML comment.
   ◦ The start and end tags for the XML comment must be on a separate line.

   The content of the XML file should look like this:

   ```xml
   <div id="facebook">
     <div id="fb-root"/>
     <script>
     <!--
     (function(d, s, id) {
       var js, fjs = d.getElementsByTagName(s)[0];
       if (d.getElementById(id)) return;
       js = d.createElement(s); js.id = id;
       js.src = "//connect.facebook.net/en_US/sdk.js#xfbml=1&version=v2.0";
       fjs.parentNode.insertBefore(js, fjs);
     }(document, 'script', 'facebook-jssdk'));
     -->
     </script>
     <div class="fb-like" data-layout="standard" data-action="like"
      data-show-faces="true" data-share="true"/>
   </div>
   
   ```

4. In Oxygen XML Editor/Author, click the Configure Transformation Scenario(s) action from the toolbar (or the Document > Transformation menu).
5. Select an existing WebHelp Classic transformation scenario (depending on your needs, it may be with or without feedback) and click the Duplicate button to open the Edit Scenario dialog box.
6. Switch to the Parameters tab and edit the `webhelp.footer.file` parameter to reference the `facebook-widget.xml` file that you created earlier.
7. Click Ok and run the transformation scenario.
Using a Script Outside of Oxygen XML Editor/Author

Important:
Running WebHelp transformations from a script outside of Oxygen XML Editor/Author requires an additional license and some additional setup:

- You must have a valid license for the Oxygen XML WebHelp Plugin (https://www.oxygenxml.com/buy_webhelp.html).
- The Oxygen XML WebHelp Plugin must be installed and integrated.

To add a Facebook™ Like widget to your WebHelp output using a script outside of Oxygen XML Editor/Author (on page 1739), follow this procedure:

1. Go to the Facebook Developers website.
2. Fill-in the displayed form, then click the Get Code button.
3. Copy the two code snippets and paste them into a `<div>` element inside an XML file called `facebook-widget.xml`. Make sure you follow these rules:
   - The file must be well-formed.
   - The code for each `<script>` element must be included in an XML comment.
   - The start and end tags for the XML comment must be on a separate line.

The content of the XML file should look like this:

```xml
<div id="facebook">
  <div id="fb-root"/>
  <script>
    <!--
    (function(d, s, id) {
      var js, fjs = d.getElementsByTagName(s)[0];
      if (d.getElementById(id)) return;
      js = d.createElement(s); js.id = id;
      js.src = "//connect.facebook.net/en_US/sdk.js#xfbml=1&version=v2.0";
      fjs.parentNode.insertBefore(js, fjs);
    }(document, 'script', 'facebook-jssdk'));
    -->
  </script>
  <div class="fb-like" data-layout="standard" data-action="like"
    data-show-faces="true" data-share="true"/>
</div>
```
4. Use the `webhelp.footer.file` parameter in your transformation script and set its value to reference the `facebook-widget.xml` file that you created earlier.

5. Execute the transformation script.

How to Add Tweet Button in WebHelp Classic Output

It is possible to integrate Twitter into your WebHelp Classic output and the widget will appear in the footer section of your WebHelp page.

Using Oxygen XML Editor/Author

To add a Twitter™ Tweet widget to your WebHelp Classic output using a transformation scenario from within Oxygen XML Editor/Author, follow this procedure:

1. Go to the Tweet button generator page.
2. Fill in the displayed form. The Preview and code area displays the code that you will need.
3. Copy the code snippet displayed in the Preview and code area and paste it into a `<div>` element inside an XML file called `tweet-button.xml`. Make sure you follow these rules:
   - The file must be well-formed.
   - The code for each `<script>` element must be included in an XML comment.
   - The start and end tags for the XML comment must be on a separate line.

   The content of the XML file should look like this:

```xml
<div id="twitter">
  <a href="https://twitter.com/share" class="twitter-share-button">Tweet</a>
  <script>
    !function (d, s, id) {
      var js, fjs = d.getElementsByTagName(s)[0], p = /^http:/.test(d.location) ? 'http': 'https';
      if (! d.getElementById(id)) {
        js = d.createElement(s);
        js.id = id;
        js.src = p + '//platform.twitter.com/widgets.js';
        fjs.parentNode.insertBefore(js, fjs);
      }
      {document, 'script', 'twitter-wjs'};
    -->
  </script>
</div>
```
4. In Oxygen XML Editor/Author, click the Configure Transformation Scenario(s) action from the toolbar (or the Document > Transformation menu).

5. Select an existing WebHelp transformation scenario (depending on your needs, it may be with or without feedback) and click the Duplicate button to open the Edit Scenario dialog box.

6. Switch to the Parameters tab and edit the webhelp.footer.file parameter to reference the tweet-button.xml file that you created earlier.

7. Click Ok and run the transformation scenario.

Using a Script Outside of Oxygen XML Editor/Author

Important:
Running WebHelp transformations from a script outside of Oxygen XML Editor/Author requires an additional license and some additional setup:

- You must have a valid license for the Oxygen XML WebHelp Plugin (https://www.oxygenxml.com/buy_webhelp.html).
- The Oxygen XML WebHelp Plugin must be installed and integrated.

To add a Twitter™ Tweet widget to your WebHelp Classic output using a script outside of Oxygen XML Editor/Author (on page 1739), follow this procedure:

1. Go to the Tweet button generator page.
2. Fill in the displayed form. The Preview and code area displays the code that you will need.
3. Copy the code snippet displayed in the Preview and code area and paste it into a `<div>` element inside an XML file called tweet-button.xml. Make sure you follow these rules:
   - The file must be well-formed.
   - The code for each `<script>` element must be included in an XML comment.
   - The start and end tags for the XML comment must be on a separate line.

The content of the XML file should look like this:

```xml
<div id="twitter">
  <a href="https://twitter.com/share" class="twitter-share-button">Tweet</a>
  <script>
    !function(d, s, id) {
      var js, fjs = d.getElementsByTagName(s)[0], p = /^http:.test(d.location) ? 'http': 'https';
      if (! d.getElementById(id)) {
        js = d.createElement(s);
        js.id = id;
        js.src = p + '://platform.twitter.com/widgets.js';
```
4. Use the `webhelp.footer.file` parameter in your transformation script and set its value to reference the `tweet-button.xml` file that you created earlier.

5. Execute the transformation script.

How to Integrate Google Analytics in WebHelp Classic Output

You can use Google Analytics to track and report site data for your WebHelp Classic output.

Using Oxygen XML Editor/Author

To integrate Google Analytics into your WebHelp Classic output using a transformation scenario from within Oxygen XML Editor/Author, follow this procedure:

1. Create a new Google Analytics account (if you do not already have one) and log on.
2. Choose the Analytics solution that best fits the needs of your website.
3. Follow the on-screen instructions to obtain a Tracking Code that contains your Tracking ID. A Tracking Code looks like this:

   ```
   <script>
   (function(i,s,o,g,r,a,m){i['GoogleAnalyticsObject']=r;i[r]=i[r]||function(){
   (i[r].q=i[r].q||[]).push(arguments)},i[r].l=1*new Date();a=s.createElement(o),
m=s.getElementsByTagName(o)[0];a.async=1;a.src=g;m.parentNode.insertBefore(a,m)
 })(window,document,'script','//www.google-analytics.com/analytics.js','ga');

   ga('create', 'UA-XXXXXXXX-X', 'auto');
   ga('send', 'pageview');

   </script>
   ```

4. Save the Tracking Code (obtained in the previous step) in a new XML file called `googleAnalytics.xml`. Note that the file should only contain the tracking code.

5. In Oxygen XML Editor/Author, click the Configure Transformation Scenario(s) action from the toolbar (or the Document > Transformation menu).

6. Select an existing WebHelp transformation scenario (depending on your needs, it may be with or without feedback) and click the Duplicate button to open the Edit Scenario dialog box.
7. Switch to the Parameters tab and edit the `webhelp.footer.file` parameter to reference the `googleAnalytics.html` file that you created earlier.

8. Click Ok and run the transformation scenario.

**Using a Script Outside of Oxygen XML Editor/Author**

!!! Important:
Running WebHelp transformations from a script outside of Oxygen XML Editor/Author requires an additional license and some additional setup:

- You must have a valid license for the Oxygen XML WebHelp Plugin (https://www.oxygenxml.com/buy_webhelp.html).
- The Oxygen XML WebHelp Plugin must be installed and integrated.

To integrate Google Analytics into your WebHelp Classic output using a script outside of Oxygen XML Editor/Author (on page 1739), follow this procedure:

1. Create a new Google Analytics account (if you do not already have one) and log on.
2. Choose the Analytics solution that best fits the needs of your website.
3. Follow the on-screen instructions to obtain a Tracking Code that contains your Tracking ID. A Tracking Code looks like this:

```
<script>
  (function(i,s,o,g,r,a,m){i['GoogleAnalyticsObject']=r;i[r]=i[r]||function(){
  (i[r].q=i[r].q||[]).push(arguments)},i[r].l=1*new Date();a=s.createElement(o),
m=s.getElementsByTagName(o)[0];a.async=1;a.src=g;m.parentNode.insertBefore(a,m)
 })(window,document,'script','//www.google-analytics.com/analytics.js','ga');

  ga('create', 'UA-XXXXXXXX-X', 'auto');
  ga('send', 'pageview');
</script>
```

4. Save the Tracking Code (obtained in the previous step) in a new XML file called `googleAnalytics.xml`. Note that the file should only contain the tracking code.
5. Use the `webhelp.footer.file` parameter in your transformation script and set its value to reference the `googleAnalytics.html` file that you created earlier.

6. Execute the transformation script.

**How to Integrate Google Search in WebHelp Classic Output**

It is possible to integrate the Google Search Engine into your WebHelp Classic output and you can specify where you want the results to appear in your WebHelp page.
Using Oxygen XML Editor/Author

To integrate the Google Search Engine into your WebHelp output using a transformation scenario from within Oxygen XML Editor/Author, follow this procedure:

1. **Go to the Google Custom Search Engine page** using your Google account.
2. **Select the Create a custom search engine button.**
3. **Follow the on-screen instructions to create a search engine for your site.** At the end of this process you should obtain a code snippet that looks like this:

```html
<script>
(function() {
  var cx = '000882108897588983:8mn4x_mf-yg';
  var gcse = document.createElement('script');
  gcse.type = 'text/javascript';
  gcse.async = true;
  var s = document.getElementsByTagName('script')[0];
  s.parentNode.insertBefore(gcse, s);
}())();
</script>
```

4. **Save the script into a well-formed HTML file called googlecse.html.**
5. In **Oxygen XML Editor/Author**, click the Configure Transformation Scenario(s) action from the toolbar (or the Document > Transformation menu).
6. **Select an existing WebHelp Responsive transformation scenario** (depending on your needs, it may be with or without feedback) and click the Duplicate button to open the Edit Scenario dialog box.
7. **Switch to the Parameters tab** and edit the `webhelp.google.search.script` parameter to reference the `googlecse.html` file that you created earlier.
8. You can also use the `webhelp.google.search.results` parameter to choose where to display the search results.
   a. Create an HTML file with the following content: `<div class="gcse-searchresults-only" data-autoSearchOnLoad="true" data-queryParameterName="searchQuery"></div>` (you must use the HTML5 version for the GCSE). For more information about other supported attributes, see Google Custom Search: Supported Attributes.
   b. Set the value of the `webhelp.google.search.results` parameter to the file path of the file you just created. If this parameter is not specified, the following code is used: `<div class="gcse-searchresults-only" data-autoSearchOnLoad="true" data-queryParameterName="searchQuery"></div>`.
9. Click **Ok** and run the transformation scenario.
Using a Script Outside of Oxygen XML Editor/Author

⚠️ **Important:**

Running WebHelp transformations from a script outside of Oxygen XML Editor/Author requires an additional license and some additional setup:

- You must have a valid license for the Oxygen XML WebHelp Plugin (https://www.oxygenxml.com/buy_webhelp.html).
- The Oxygen XML WebHelp Plugin must be installed and integrated.

To integrate the Google Search Engine into your WebHelp Classic output using a script outside of Oxygen XML Editor/Author (on page 1739), follow this procedure:

1. Go to the Google Custom Search Engine page using your Google account.
2. Select the Create a custom search engine button.
3. Follow the on-screen instructions to create a search engine for your site. At the end of this process you should obtain a code snippet that looks like this:

```html
<script>
(function() {
  var cx = '000888210889775888983:8mn4w_mf-yg';
  var gcse = document.createElement('script');
  gcse.type = 'text/javascript';
  gcse.async = true;
  var s = document.getElementsByTagName('script')[0];
  s.parentNode.insertBefore(gcse, s);
}());
</script>
```

4. Save the script into a well-formed HTML file called `googlecse.html`.
5. Use the `webhelp.google.search.script` parameter in your transformation script and set its value to reference the `googlecse.html` file that you created earlier.
6. You can also use the `webhelp.google.search.results` parameter to choose where to display the search results.
   a. Create an HTML file with the following content: `<div class="gcse-searchresults-only" data-autoSearchOnLoad="true" data-queryParameterName="searchQuery"></div>` (you must use the HTML5 version for the GCSE). For more information about other supported attributes, see Google Custom Search: Supported Attributes.
b. Set the value of the `webhelp.google.search.results` parameter to the file path of the file you just created. If this parameter is not specified, the following code is used:

```
<div class="gcse-searchresults-only" data-autoSearchOnLoad="true" data-queryParameterName="searchQuery"></div>
```

7. Execute the transformation script.

**Miscellaneous Customization Topics**

This section contains miscellaneous topics about how to customize the WebHelp Classic output.

**How to Disable Caching in WebHelp Classic Output**

In cases where a set of WebHelp Classic pages need to be updated on a regular basis to deliver the latest version of the documentation, the WebHelp pages should always be requested from the server upon re-loading it in a Web browser on the client side, rather than re-using an outdated `cached` version in the browser.

To disable caching in WebHelp Classic output, follow this procedure:

1. Edit the following file: `\OXYGEN_INSTALL_DIR\frameworks\docbook\xsl\com.oxygenxml.webhelp.classic\xsl\createMainFiles.xsl`

2. Locate the following template in the XSL file: `<xsl:template name="create-toc-common-file">` and add the following code snippet:

```xml
<meta http-equiv="Pragma" content="no-cache" />
<meta http-equiv="Expires" content="-1" />
```

**Note:**

The code should look like this:

```xml
<html>
  <head>
    <xsl:if test="$withFrames">
      <base target="contentwin"/>
    </xsl:if>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
    <!-- Disable caching of WebHelp pages in web browser. -->
    <meta http-equiv="Pragma" content="no-cache" />
    <meta http-equiv="Expires" content="-1" />
  </head>
  ....
</html>
```

3. Save your changes to the file.

4. Re-run your WebHelp transformation scenario.
How to Publish WebHelp Classic Output on a SharePoint Site

Since WebHelp output must be published locally, on the same machine where the WebHelp process is running, to publish your files directly to a SharePoint library you need to map a network drive to connect to SharePoint and change your file extensions to .aspx, as described in the steps below.

Using Oxygen XML Editor/Author

To publish WebHelp Classic output on a SharePoint site and use a transformation scenario from within Oxygen XML Editor/Author, follow this procedure:

1. Map a network drive to connect to your SharePoint library. For more information, see: https://support.microsoft.com/en-us/kb/2616712.
2. Edit the WebHelp transformation scenario and open the Parameters tab. Set the html.ext parameter to .aspx.
3. Run the WebHelp transformation scenario to generate the output.

Using a Script Outside of Oxygen XML Editor/Author

Important:
Running WebHelp transformations from a script outside of Oxygen XML Editor/Author requires an additional license and some additional setup:

- You must have a valid license for the Oxygen XML WebHelp Plugin (https://www.oxygenxml.com/buy_webhelp.html).
- The Oxygen XML WebHelp Plugin must be installed and integrated.

To publish WebHelp Classic output on a SharePoint site and use a script outside of Oxygen XML Editor/Author (on page 1739), follow this procedure:

1. Map a network drive to connect to your SharePoint library. For more information, see: https://support.microsoft.com/en-us/kb/2616712.
2. Use the html.ext parameter in your transformation script and set its value to .aspx.
3. Execute the transformation script.

DITA to PDF Output Customization

Oxygen XML Editor provides support for generating PDF output using transformation scenarios for certain types of documents (for example, DITA, DocBook, TEI, and JATS) and Oxygen XML Editor supports several different types of processors. There are numerous ways to customize the published output to fit your specific needs.
CSS-based DITA to PDF Customization

Oxygen XML Editor comes bundled with a **DITA-OT CSS-based PDF Publishing Plugin** for transforming DITA maps or single topics to PDF, while styling the resulting output using CSS. It is the base of two types of transformation scenarios:

**DITA Map Transformation Type (DITA Map PDF - based on HTML5 & CSS)**

This transformation type converts DITA maps to PDF using a CSS-based processing engine and HTML5 as an intermediate format. For this transformation, the `pdf-css-html5` transtype is used. Because the structure of the HTML5 intermediate format resembles the one used in WebHelp output, it is possible to reuse parts of your CSS file you developed for a WebHelp customization.

**Single Topic Transformation Type (DITA PDF - based on HTML5 & CSS)**

This transformation type converts a single DITA topic to PDF using a CSS-based processing engine and HTML5 as an intermediate format. For this transformation, the `pdf-css-html5-single-topic` transtype is used. This transformation is derived from the DITA Map PDF - based on HTML5 & CSS transformation type but applies on a single topic.

Related Information:

- DITA Map PDF - based on HTML5 & CSS Transformation *(on page 1449)*
- DITA PDF - based on HTML5 & CSS Transformation *(on page 3103)*

Overview

This section contains topics that provide a basic overview of the **DITA-OT CSS-based PDF Publishing Plugin**, technical details, and some additional resources to help you with your customizations.

**Tip:**

For more information and some tips in regard to publishing DITA documents to PDF using CSS, watch our Webinars:

- Transforming DITA documents to PDF using CSS, Part 1 – Page Definitions, Cover Page and PDF Metadata.
- Transforming XML and HTML documents to PDF using CSS, Part 2 – Lists, Tables and Images.
- Transforming XML and HTML documents to PDF using CSS, Part 3 – Global Page Layout.
- Transforming XML and HTML documents to PDF using CSS, Part 4 – Advanced Functionalities.
Resources

Customizing the PDF output requires knowledge of CSS, Paged Media, and DITA. The following list provides some resources to help you:


- **CSS Paged Media** - This is a part of the CSS specification that shows how to organize your publication in pages, how to use headers/footers, page breaks, and other page-related issues. The specification is available here: [https://www.w3.org/TR/CSS2/page.html](https://www.w3.org/TR/CSS2/page.html). Also, there is a set of hands-on examples in the Oxygen PDF Chemistry user guide: [https://www.oxygenxml.com/doc/ug-chemistry/](https://www.oxygenxml.com/doc/ug-chemistry/).

- **DITA** - You will need a basic understanding of DITA elements, attributes, and structure. A good resource is *The DITA Style Guide - Best Practices for Authors* by Tony Self. It is available at: [www.ditastyle.com](http://www.ditastyle.com) and: [https://www.oxygenxml.com/dita/styleguide/webhelp-feedback/Artefact/c_DITA_Authoring_Concepts.html](https://www.oxygenxml.com/dita/styleguide/webhelp-feedback/Artefact/c_DITA_Authoring_Concepts.html). You can find all the details for every DITA element on OASIS website: [http://docs.oasis-open.org/dita/v1.2/os/spec/DITA1.2-spec.html](http://docs.oasis-open.org/dita/v1.2/os/spec/DITA1.2-spec.html).


**Related Information:**

DITA-OT DAY 2017: Using CSS to Style PDF Output

**Supported Processors**

The **DITA-OT CSS-based PDF Publishing Plugin** supports the following CSS processors:

- **Oxygen PDF Chemistry** - This is recommended processor because the built-in CSS files were fine-tuned for this processor. For example, metadata extraction *(on page 1863)* only functions with this processor. If the plugin is started from an Oxygen XML Editor/Author distribution, a Chemistry installation is not needed.


**Technical Details**

It has the following transformation types:

- **pdf-css-html5** (*DITA Map PDF - based on HTML5 & CSS transformation*) - CSS styling applied over a merged HTML5 document (the merged DITA map converted to HTML5).
- **pdf-css-html5-single-topic** (*DITA PDF - based on HTML5 & CSS transformation*) - CSS styling applied over a merged HTML5 document (the merged DITA topic converted to HTML5).

This is how it works:

1. It expands all the topic references into a temporary clone of the map, resolving keys and reused content. For the single topic transformation the result is a file with the keys and content resolved.
2. It generates a structure for the table of contents and index. The result is a merged map with all the references resolved. When transforming a single topic, the TOC and Index are not added to the merged file, this includes only the contents of the topic.
3. Then it post-processes the merged map. It fixes some of the structure in the TOC and index, moves the *frontmatter* and *backmatter* to the correct places, transforms any change tracking and review processing instructions to elements that can be styled later, etc. The result is another merged map.
Note:
In the single topic transformation type (**DITA PDF - based on HTML5 & CSS**), these steps are simplified.

4. The merged DITA map or topic is transformed to a single HTML5 file.

The generated HTML elements copy the class attributes from the original DITA elements. In this way you can either use selectors that were designed for DITA structure, or ones for the HTML structure - see: **Reusing the Styling for WebHelp and PDF Output** (on page 1946). This step can apply customization **XSLT extension points** (on page 1794) that change the default processing.

5. In the last phase, it uses a collection of CSS files to style the merged map. References to the CSS files are collected from the **publishing template** (on page 1785).
Increasing Memory Allocation for Java

If you are working with a large project with extensive metadata or key references, you may need to increase the amount of memory that is allocated to the Java process that performs the publishing.

**When the Transformation is Started from Oxygen**

To alter the memory allocation setting from the transformation scenario, follow these steps:

1. Open the **Configure Transformation Scenario(s)** dialog box.
2. Select your transformation scenario, then click **Edit**.
3. Go to the **Advanced** tab.
4. Locate the **JVM Arguments** and increase the default value. For instance, to set 2 gigabytes as the maximum amount of memory, you can use: `-Xmx2g`. If you do not specify the `-Xmx` value in this field, by default, the application will use a maximum of 512 megabytes when used with a 32-bit Java Virtual Machine and one gigabyte with a 64-bit Java Virtual Machine.

**Note:**

This memory setting is used by both the DITA-OT process and the Chemistry CSS processor.

**When the Transformation is Started from the Command Line**

There can be two situations where an out of memory error can be triggered:

- From the DITA-OT basic processing (the preparation of the merged XML document).
- From the Chemistry PDF CSS processor (the transformation of the merged XML document to PDF).

To solve both of them, you can change the value of the **ANT_OPTS** environment variable from a command line for a specific session.

**Example:** To increase the JVM memory allocation to 1024 MB for a specific session, issue the following command from a command prompt (depending on your operating system):

- **Windows**
  
  ```
  set ANT_OPTS=%ANT_OPTS% -Xmx1024M
  ```

- **Linux/macOS**
  
  ```
  export ANT_OPTS="$ANT_OPTS -Xmx1024M"
  ```

**Tip:**

To persistently change the memory allocation, change the value allocated to the **ANT_OPTS** environment variable on your system.
If the Chemistry PDF CSS processor fails with an Out Of Memory Error, try adding the baseJVMArgLine parameter to the DITA-OT command line. For example:

-DbaseJVMArgLine=-Xmx984m

**Transformation Parameters**

This list includes the most common customization parameters that are available in the DITA Map PDF - based on HTML5 & CSS transformation scenario. Other standard DITA-OT parameters were omitted for clarity, but they are supported.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>args.allow.external.coderefs</td>
<td>Enables the inclusion of code files that are located outside the DITA map folder hierarchy, referenced using the DITA <code>&lt;coderef&gt;</code> element. Allowed values are <strong>yes</strong> or <strong>no</strong> (default).</td>
</tr>
<tr>
<td>args.chapter.layout</td>
<td>Specifies whether chapter-level TOCs are generated for bookmaps. When set to <strong>MINITOC</strong>, a small section with links is added at the beginning of each chapter. The default is <strong>BASIC</strong>. For details, see: Table of Contents on a Page (Mini TOC) (on page 1897). Allowed values:</td>
</tr>
<tr>
<td></td>
<td>• <strong>BASIC</strong> - No chapter TOC is created.</td>
</tr>
<tr>
<td></td>
<td>• <strong>MINITOC</strong> - A chapter-level TOC will be generated.</td>
</tr>
<tr>
<td></td>
<td>• <strong>MINITOC-BOTTOM-LINKS</strong> - A chapter-level TOC will be generated, with the links under the chapter description.</td>
</tr>
<tr>
<td>args.css</td>
<td>You can use this to specify a list of CSS URLs to be used in addition to those specified in the <code>dita.css.list</code> parameter or publishing template. The files must have URL syntax and be separated using semicolons.</td>
</tr>
<tr>
<td>args.css.param.*</td>
<td>You can use this parameter pattern to set attributes on the root of the merged map. This means you can activate specific CSS rules from your custom CSS using custom attributes. For examples, see: Styling Through Custom Parameters (on page 1975).</td>
</tr>
<tr>
<td>args.css.param.numbering</td>
<td>You can use this parameter to change the numbering of the first-level topics (chapters) and nested topics. Allowed values:</td>
</tr>
<tr>
<td></td>
<td>• <strong>shallow</strong> - Only the topics from the first level will be numbered (chapters). This is the default.</td>
</tr>
<tr>
<td></td>
<td>• <strong>deep</strong> - All the topics from the map will be numbered (nested topics up to level 3).</td>
</tr>
</tbody>
</table>
| | • **deep-chapter-scope** - Similar to deep, but in addition, the page numbers, figures, and table numbers are reset at the start of each first-level topic (chapter). The table and figure titles (and the links to them) are prefixed with the chapter numbers. The generic cross
reference links contain both the first-level topic (chapter) numbers and the page numbers to avoid ambiguity. This parameter value is only available for the DITA Map PDF - based on HTML5 & CSS transformation scenario.

- **deep-chapter-scope-no-page-reset** - Similar to deep-chapter-scope, but the page numbers do not reset at the start of each first-level topic (chapter). The generic cross reference links contain only the page number. This parameter value is only available for the DITA Map PDF - based on HTML5 & CSS transformation scenario.

For more details, see Numbering Types (on page 1883).

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>args.css.param.numbering-sections</strong></td>
<td>Controls whether or not the sections are included in the table of contents. When set to yes (sections are included), they are numbered according the numbering scheme set by the args.css.param.numbering parameter.</td>
</tr>
<tr>
<td><strong>args.css.param.show-onpage-lbl</strong></td>
<td>Controls whether or not the links will have an on page NN label after them. This parameter has different defaults, depending on the transformation type. For map transformations (pdf-css-html5 trans type), the default is yes. For topic transformations (pdf-css-html5-single-topic trans type), the default is no.</td>
</tr>
<tr>
<td><strong>args.css.param.title.layout</strong></td>
<td>Changes the structure of the title element. In the output, the title area consists of two parts: one is the number of the chapter (and optionally, the sections number), and one is the title text. This parameter allows a switch between normal text flow (in-line flow) and a table layout where the number is placed in one cell and the text in the other (to avoid wrapping text under the chapter number).</td>
</tr>
<tr>
<td><strong>• normal</strong></td>
<td></td>
</tr>
<tr>
<td><strong>• table</strong></td>
<td>(avoid wrapping text under counter)</td>
</tr>
<tr>
<td><strong>args.draft</strong></td>
<td>Specifies whether or not the content of &lt;draft-comment&gt; and &lt;required-cleanup&gt; elements is included in the output.</td>
</tr>
<tr>
<td>Allowed values:</td>
<td></td>
</tr>
<tr>
<td><strong>• no</strong></td>
<td>(default) - No draft information is shown in the output.</td>
</tr>
<tr>
<td><strong>• yes</strong></td>
<td>- The draft information is shown in the output.</td>
</tr>
<tr>
<td><strong>args.figurelink.style</strong></td>
<td>Specifies how cross references to figures are styled in output. Allowed values:</td>
</tr>
<tr>
<td>Args</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>args.gen.task.lbl</td>
<td>Specifies whether or not to generate headings for sections within task topics. Allowed values: YES or NO (default). When set to YES, headings such as &quot;About this task&quot;, &quot;Before you begin&quot;, &quot;Procedure&quot;, or &quot;What to do next&quot;, are shown in the task contents.</td>
</tr>
<tr>
<td>args.hyph.dir</td>
<td>Specifies the directory that contains custom hyphenation dictionaries. For more details see: [Hyphenation](on page 1924).</td>
</tr>
<tr>
<td>args.input</td>
<td>Specifies the main DITA map file for your documentation project.</td>
</tr>
<tr>
<td>args.keep.output.debug.files</td>
<td>Specifies whether or not the debug files generated during the transformation should be kept in the output folder. Allowed values: YES (default) or NO.</td>
</tr>
<tr>
<td>args.output.base</td>
<td>Specifies the name of the output file without a file extension. By default, the name of the PDF file is derived from the name of the DITA map file. This parameter allows you to override it.</td>
</tr>
<tr>
<td>args.rellinks.group.mode</td>
<td>Specifies the related links grouping mode. All links can be grouped into a single &quot;Related Information&quot; group or links grouped by their target type (topic, task, or concept). Allowed values: single-group (default) or group-by-type. For more details see: [How to Group Related Links by Type](on page 1954).</td>
</tr>
<tr>
<td>args.tablelink.style</td>
<td>Specifies how cross references to tables are styled in output. Allowed values:</td>
</tr>
<tr>
<td></td>
<td>• NUMBER - Only the number of the tables will be shown in links.</td>
</tr>
<tr>
<td></td>
<td>• TITLE - Only the title of the tables will be shown in links.</td>
</tr>
<tr>
<td></td>
<td>• NUMTITLE (default) - Both the title and number of the tables will be shown in links.</td>
</tr>
<tr>
<td>clean.temp</td>
<td>Specifies whether or not the DITA-OT deletes the files in the temporary directory after it finishes a build. Allowed values: yes (default) or no.</td>
</tr>
<tr>
<td>chemistry.security.policy</td>
<td>Specifies a Java policy file that applies to the Chemistry process. A template can be found here: plugins/com.oxygenxml.pdf.css/lib/oxygen-pdf-chemistry/config/chemistry.policy.</td>
</tr>
<tr>
<td>chemistry.security.workspace</td>
<td>Specifies a directory where the temporary files and font cache created by the Chemistry process need to be stored. This becomes required when the chemistry.security.policy is specified.</td>
</tr>
<tr>
<td>Parameter Name</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>chemistry.security.resources.dir</td>
<td>Path to an additional folder that Chemistry will use to read its resources (CSS, images). The process already has read access to the input map folder, the publishing templates folder, and the OPE install folder. This optional parameter should only be used when the <code>chemistry.security.policy</code> parameter is set.</td>
</tr>
<tr>
<td>chemistry.security.resources.host</td>
<td>The host, specified as <code>name:port</code>, that Chemistry will use to get resources (e.g. CSS files, images, fonts). This optional parameter should only be used when the <code>chemistry.security.policy</code> parameter is set.</td>
</tr>
<tr>
<td>css.processor.path.antenna-house</td>
<td>Path to the Antenna House executable file that needs to be run to generate the PDF (for example, <code>C:\path\to\AHFCmd.exe</code> on Windows).</td>
</tr>
<tr>
<td>css.processor.path.chemistry</td>
<td>Path to the Oxygen PDF Chemistry executable file that needs to be run to generate the PDF (for example, <code>C:\path\to\chemistry.bat</code> on Windows). If this parameter is not set, the plugin will use the system's PATH environment variable to locate and start Oxygen PDF Chemistry.</td>
</tr>
<tr>
<td>css.processor.path.prince</td>
<td>Path to the Prince executable file that needs to be run to generate the PDF (for example, <code>C:\path\to\prince.exe</code> on Windows).</td>
</tr>
<tr>
<td>css.processor.type</td>
<td>Specifies the processor to use for the transformation. Allowed values: chemistry (default), antenna-house, or prince.</td>
</tr>
<tr>
<td>default.language</td>
<td>Specifies the default language for source documents. Examples: fr, de, zh, etc. Depending on the transformation type, the actual number of supported languages can vary, see: Localization (on page 1979).</td>
</tr>
<tr>
<td>drop.block.margins.at.page-boundary</td>
<td>Specifies that the top and bottom margins associated with a block element should be discarded when the block is at the top or bottom of the page. Allowed values: YES (default) or NO.</td>
</tr>
<tr>
<td>editlink.ditamap.edit.url</td>
<td>Use this parameter to add an Edit link next to the topic title in the WebHelp output. When a user clicks the link, the topic is opened in Oxygen XML Web Author or Content Fusion where they can make changes that can be saved to a file server. The value should be set as the edit URL of the main DITA map used for publishing your output. The easiest way to obtain the URL is to open the map in Web Author or Content Fusion and copy the URL from the browser's address bar.</td>
</tr>
<tr>
<td>editlink.additional.query.parameters</td>
<td>You can use this optional parameter to add additional parameters to be appended to each generated edit link. Each parameter must start with &amp; (for example: &amp;tags-mode=no-tags).</td>
</tr>
<tr>
<td>editlink.remote.ditamap.url (deprecated)</td>
<td>Use this parameter in conjunction with <code>editlink.web.author.url</code> to add an Edit link next to the topic title in the PDF output. When a user clicks the link, the topic is opened in Oxygen XML Web Author where they can make changes that can be saved to a file server. The value</td>
</tr>
</tbody>
</table>
should be set as the custom URL of the main DITA map. For example, a GitHub custom URL might look like this: https://getFileContent/oxyengxml/userguide/master/UserGuide.ditamap.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>editlink.web.author.url (deprecated)</td>
<td>This parameter needs to be used in conjunction with editlink.remote.ditamap.url to add an Edit link next to the topic title in the PDF output. When a user clicks the link, the topic is opened in Oxygen XML Web Author where they can make changes that can be saved to a file server. The value should be set as the URL of the Web Author installation. For example: <a href="https://www.oxygenxml.com/oxygen-xml-web-author/">https://www.oxygenxml.com/oxygen-xml-web-author/</a>.</td>
</tr>
<tr>
<td>enable.latin.glyph.substitutions</td>
<td>When set to yes (default), glyph substitution is enabled (if the particular font supports it). This applies to Latin-based scripts only (the substitutions are always enabled in other types of scripts). If you encounter problems rendering or copying accented glyphs (i.e. umlauts or other diacritics), it might be helpful to set this parameter to no to disable the font glyph substitutions. Another example of a case when you might need to disable the substitutions is a situation where an accented character cannot be mapped to a compound glyph, resulting in the glyph not being rendered in the PDF output.</td>
</tr>
<tr>
<td>Warnings:</td>
<td></td>
</tr>
<tr>
<td>• Disabling substitutions also disables Latin ligatures.</td>
<td></td>
</tr>
<tr>
<td>• Disabling substitutions is not recommended unless absolutely necessary. It is better practice to use another font if you can find one that does not have the rendering issues.</td>
<td></td>
</tr>
<tr>
<td>figure.title.placement</td>
<td>Controls the title placement of the figures, relative to the image. Possible values include top (default) and bottom.</td>
</tr>
<tr>
<td>fix.external.refs.com.oxygenxml</td>
<td>The DITA Open Toolkit usually has problems processing references that point to locations outside of the processed DITA map directory. This parameter is used to specify whether or not the application should try to fix such references in a temporary files folder before the DITA Open Toolkit is invoked on the fixed references. The fix has no impact on your edited DITA content. Allowed values: true or false (default).</td>
</tr>
<tr>
<td>hide.frontpage.toc.index.glossary</td>
<td>When set to yes, the generated structures (table of contents, index list, front page, etc.) are removed from the output. The default is no.</td>
</tr>
<tr>
<td>image.resolution</td>
<td>You can use this parameter to set the default resolution used by images. It works mainly on vector images since raster images have their reso-</td>
</tr>
</tbody>
</table>
Oxygen XML Editor 24.1

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>pdf.accessibility</code></td>
<td>When set to <strong>yes</strong>, the PDF output will be generated in compliance with the PDF/Universal Accessibility standard (also known as ISO 14289). The default is <strong>no</strong>.</td>
</tr>
<tr>
<td><code>pdf.version</code></td>
<td>Use this parameter to specify the version of the produced PDF. It has no impact on the set of PDF features used by the engine, but may be used to signal a compatibility level to the PDF readers. The default is <strong>1.5</strong>.</td>
</tr>
<tr>
<td><code>pdf.securityrestrict.printhq</code></td>
<td>Restricts high quality printing. Used for protecting the PDF Document. The restriction is off by default. Accepted values: <strong>yes</strong> or <strong>no</strong>.</td>
</tr>
<tr>
<td><code>pdf.securityrestrict.assembledoc</code></td>
<td>Restricts assembling document (e.g. adding pages). Used for protecting the PDF Document. The restriction is off by default. Accepted values: <strong>yes</strong> or <strong>no</strong>.</td>
</tr>
<tr>
<td><code>pdf.securityrestrict.accesscontent</code></td>
<td>Restricts extracting text and graphics. Used for protecting the PDF Document. The restriction is off by default. Accepted values: <strong>yes</strong> or <strong>no</strong>.</td>
</tr>
<tr>
<td><code>pdf.securityrestrict.fillinforms</code></td>
<td>Restricts filling in existing interactive forms. Used for protecting the PDF Document. The restriction is off by default. Accepted values: <strong>yes</strong> or <strong>no</strong>.</td>
</tr>
<tr>
<td><code>pdf.securityrestrict.annotations</code></td>
<td>Restricts filling in existing interactive forms. Used for protecting the PDF Document. The restriction is off by default. Accepted values: <strong>yes</strong> or <strong>no</strong>.</td>
</tr>
<tr>
<td><code>pdf.securityrestrict.print</code></td>
<td>Restricts printing. Used for protecting the PDF Document. The restriction is off by default. Accepted values: <strong>yes</strong> or <strong>no</strong>.</td>
</tr>
<tr>
<td><code>pdf.securityrestrict.copy</code></td>
<td>Restricts copying content. Used for protecting the PDF Document. The restriction is off by default. Accepted values: <strong>yes</strong> or <strong>no</strong>.</td>
</tr>
<tr>
<td><code>pdf.securityrestrict.edit</code></td>
<td>Restricts copying content. Used for protecting the PDF Document. The restriction is off by default. Accepted values: <strong>yes</strong> or <strong>no</strong>.</td>
</tr>
<tr>
<td><code>pdf.security.user.password</code></td>
<td>User password. The document can be opened using this password. When the owner password parameter is not specified, the user password gives full rights to the people using it. When the owner password parameter is specified, the people can open the document using the user password but restrictions will apply. Missing by default.</td>
</tr>
<tr>
<td><code>pdf.security.owner.password</code></td>
<td>Owner password. There are no restrictions for people using this password.</td>
</tr>
<tr>
<td><code>pdf.security.encrypt.metadata</code></td>
<td>Encrypts the metadata. By default active when other security parameters are set. Accepted values: <strong>yes</strong> or <strong>no</strong>.</td>
</tr>
<tr>
<td><code>show.changes.and.comments</code></td>
<td>When set to <strong>yes</strong>, the user comments, colored highlights and tracked changes are shown in the output.</td>
</tr>
<tr>
<td>Parameter</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>show.changes.and.comments.as.changebars</td>
<td>When set to <strong>yes</strong> (default) and the <code>show.changes.and.comments</code> parameter is also set to <strong>yes</strong>, the user comments and tracked changes are shown as change bars in the PDF output. This parameter can be used in conjunction with the <code>show.changes.and.comments.as.pdf.sticky.notes</code> parameter to choose whether the change bars are displayed in footnotes or sticky notes. You can override this from your customization CSS <em>(on page 1814)</em>.</td>
</tr>
<tr>
<td>show.changes.and.comments.as.pdf.sticky.notes</td>
<td>When set to <strong>yes</strong> (default) and the <code>show.changes.and.comments</code> parameter is also set to <strong>yes</strong>, the user comments and tracked changes are shown in the PDF output as sticky note annotations. When set to <strong>no</strong>, the comments and tracked changes are left in the document model and are styled by the default CSS rules as footnotes. You can override this from your customization CSS <em>(on page 1814)</em>.</td>
</tr>
<tr>
<td>show.changed.text.in.pdf.sticky.notes.content</td>
<td>When set to <strong>yes</strong> (default) and both the <code>show.changes.and.comments</code> and <code>show.changes.and.comments.as.pdf.sticky.notes</code> parameters are also set to <strong>yes</strong>, the inserted and deleted text is shown in the sticky note annotations. When set to <strong>no</strong>, only the inserted and deleted labels are shown in the annotations (this is useful for search scope).</td>
</tr>
<tr>
<td>show.image.map.area.numbers</td>
<td>When set to <strong>yes</strong>, a counter for each area from the image map will be displayed over the image, near the defined shape. The default is <strong>no</strong>.</td>
</tr>
<tr>
<td>show.image.map.area.shapes</td>
<td>When set to <strong>yes</strong>, each of the image map area shapes will be displayed with a translucent fill over the image. You can use this to debug your image maps. The default is <strong>no</strong>.</td>
</tr>
<tr>
<td>show.media.as.link</td>
<td>When set to <strong>yes</strong>, media objects will not appear and an external link will be generated for each one instead.</td>
</tr>
<tr>
<td>sort.and.group.glossentries</td>
<td>When set to <strong>no</strong> (default), elements in the glossary will be sorted based upon the document order. If set to <strong>yes</strong>, elements in the glossary will be sorted alphabetically and grouped by their first letter.</td>
</tr>
<tr>
<td>store-type</td>
<td>Setting this parameter to <strong>memory</strong> will increase the processing speed and thus, could help decrease the publishing time.</td>
</tr>
<tr>
<td>table.title.placement</td>
<td>Controls the placement of the title for tables. Possible values include <strong>top</strong> (default) and <strong>bottom</strong>.</td>
</tr>
<tr>
<td>table.title.repeat</td>
<td>Specifies whether or not a table caption should repeat on other pages when the table spans onto multiple pages. The caption is not repeated for tables nested in lists or other tables. Allowed values are <strong>yes</strong> (default) or <strong>no</strong>.</td>
</tr>
</tbody>
</table>
### Oxygen XML Editor 24.1

**use.css.for.embedded.svg**

When set to **yes** (default), the CSS files specified in the publishing template or by the `args.css` parameter are also applied on embedded SVG elements. Allowed values are **yes** and **no**.

**use.navtitles.in.all.links**

Specifies whether a `<navtitle>` defined in a topic or a topic reference should be used as the display name for all links or only in the table of contents. Allowed values are **yes** and **no** (default).

**parallel**

Specifies whether or not certain pre-processing tasks should be run in parallel. Setting this parameter to **true** may add a small increase to the publishing speed. Allowed values are: **true** and **false** (default).

The following parameters can be used to specify a publishing template:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>pdf.publishing.template</code></td>
<td>Specifies the path to the folder containing the custom PDF template</td>
</tr>
<tr>
<td><code>pdf.publishing.template.descriptor</code></td>
<td>Specifies the name of the descriptor file to be loaded from the PDF template folder or package. If it is not specified, the first encountered descriptor file will be loaded.</td>
</tr>
</tbody>
</table>

The following parameter is available on all DITA transformations when using the **Oxygen Publishing Engine**:

**args.disable.security.checks**

Specifies whether or not to load external entities that are not solved through catalogs. For security reasons, the default is **no**.

Allowed values:

- **yes**
- **no** (default)

The following parameters are only available for the **DITA PDF - based on HTML5 & CSS** single DITA topic transformation scenario (`pdf-css-html5-single-topic` trans type):

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>args.root.map</code></td>
<td>Specifies the path of the root map file used to expand the key references in the published topic.</td>
</tr>
<tr>
<td><code>args.enable.root.map.key.processing</code></td>
<td>Indicates whether or not the keys should be processed using the root map parameter. Allowed values:</td>
</tr>
<tr>
<td></td>
<td>• <strong>auto</strong> (default)</td>
</tr>
<tr>
<td></td>
<td>• <strong>yes</strong></td>
</tr>
<tr>
<td></td>
<td>• <strong>no</strong></td>
</tr>
</tbody>
</table>
**Console Logging**

To activate the logging of the last processing stage, involving the usage of the Chemistry processor to generate the PDF from the merged HTML, use the `--verbose` (or `-v`) DITA-OT parameter from the command line.

**Note:**
When the transformation is started from an Oxygen application, this parameter is automatically set.

**License Key**

**Chemistry License**

If you have an Oxygen PDF Chemistry license key, you will be able to generate PDF output that is not stamped with the Chemistry logo image.

To install your Chemistry license key:

- If you are using the version of Chemistry that comes bundled in Oxygen XML Editor/Author, save the license key text in a file with the name `licensekey.txt` and place it in the `DITA-OT-DIR/plugins/com.oxygenxml.pdf.css/lib/oxygen-pdf-chemistry` folder.
- If you are using another Chemistry installation, make sure you place the `licensekey.txt` file in that folder.

**Oxygen Publishing Engine License**

If you have purchased a license for the Oxygen Publishing Engine, you will be able to produce both PDF and WebHelp output without any restrictions.

To install your Oxygen Publishing Engine license key, save the license key text in a file with the name `licensekey.txt` and place it in the `DITA-OT-DIR` folder.

**Generating PDF Output**

The publishing process can be initiated from a transformation scenario within Oxygen XML Editor/Author, from a command line outside Oxygen XML Editor/Author, or from an integration server.

**Generating PDF from a Command Line**

To publish the PDF output from a command line outside of Oxygen XML Editor/Author, you can use the `dita` startup script that comes bundled with the DITA Open Toolkit distribution.

The command line supports all the parameters specific to the PDF transformation (on page 1773). Here is an example of how to write the commands:
Generating PDF from an Integration Server

PDF output can be automatically generated from a Continuous Integration/Continuous Delivery system, such as Jenkins.

To integrate PDF output with the Jenkins CI tool, follow these steps:

1. Create a Maven project to incorporate the DITA-OT that already integrates Oxygen XML Editor.
2. Go to the root of your Maven project and edit the `pom.xml` file to include the following fragment:

```xml
<project>
  <modelVersion>4.0.0</modelVersion>

  <groupId>com.oxygenxml</groupId>
  <artifactId>oxygen-oxygen-pdf-css-generator</artifactId>
  <version>1.0</version>

  <properties>
    <!-- The path to Oxygen Publishing Engine -->
    <dita-ot-dir>/path/to/oxygen-publishing-engine-3.x</dita-ot-dir>
    <!-- The path to the DITA map that you want to process. -->
    <input-file>/path/to/map.ditamap</input-file>
    <!-- The path of the output directory. -->
    <output-dir>/path/to/output/folder</output-dir>
    <!-- The path to the PDF publishing template folder (containing the .opt file). -->
    <publishing-template>/path/to/template/folder</publishing-template>
  </properties>

  <build>
    <plugins>
```

Note:
You can use the long form of the command-line options (e.g. `--format` or `--input`). For more information about the build parameters, see DITA-OT: Building Output Using the `dita` Command.
3. Go to the Jenkins top page and create a new Jenkins job. Configure this job to suit your particular requirements, such as the build frequency and location of the Maven project.

Related information

Webinar: Introducing the Oxygen Publishing Engine for DITA

Publishing Templates

An Oxygen Publishing Template defines all aspects of the layout and styles for output obtained from the following transformation scenarios:

- WebHelp Responsive
- DITA Map PDF - based on HTML5 & CSS
It is a self-contained customization package stored as a ZIP archive or folder that can easily be shared with others. It provides the primary method for customizing the output.

Some possible customization methods include:

- Add additional template resources to customize the output (such as logos, Favicons, or CSS files).
- Extend the default processing by specifying one or more XSLT extension points.
- Specify one or more transformation parameters to customize the output.
- Customize various aspects of the output through simple CSS styling.
- For WebHelp Responsive output, change the layout of the main page or topic pages by customizing which components will be displayed and where they will be positioned in the page.

The following graphics are possible sample structures for Oxygen Publishing Template packages:

**Figure 507. Oxygen Publishing Template Package (WebHelp Responsive)**

- **publishing template**
  - CSS
    - oxygen-skin.css
  - JS
  - fonts
  - HTML-Fragments
    - webhelp.fragment.footer.html
    - webhelp.fragment.before.logo_and_title.html
  - XSLT-Extensions
    - topic_page_extension.xsl
  - page-templates
    - wt_index.html
    - wt_search.html
    - wt_topic.html
    - wt_terms.html
  - template_descriptor.opt

**Figure 508. Oxygen Publishing Template Package (PDF)**

- **publishing template**
  - CSS
    - oxygen-skin.css
  - XSLT-Extensions
    - topic_page_extension.xsl
  - template_descriptor.opt

For information about creating and customizing publishing templates, and how to adjust the WebHelp and PDF output through CSS styling and other customization methods, watch our Webinar: Creating Custom
Publishing Templates for WebHelp and PDF Output. The Webinar slides and sample project are also available from that webpage.

Related Information:
How to Create a Publishing Template (on page 1642)
How to Edit a Packed Publishing Template (on page 1644)
How to Add a Publishing Template to the Publishing Templates Gallery (on page 1644)
How to Share a Publishing Template (on page 1794)

Publishing Template Package Contents for PDF Customizations

An Oxygen Publishing Template for PDF output must contain a template descriptor file and at least one CSS file, and may contain other resources (such as graphics, XSLT files, etc.). All the template resources can be stored in either a ZIP archive or in a folder. It is recommended to use a ZIP archive because it is easier to share with others.

Template Descriptor File

Each publishing template includes a descriptor file that defines the meta-data associated with template. It is an XML file with certain elements that defines all the resources included in a template (such as CSS files, images, and transformation parameters).

The template descriptor file must have the .opt file extension and must be located in the templates' root folder.

A PDF template descriptor might look like this:

```
<publishing-template>
  <name>Flowers</name>

  <pdf>
    <tags>
      <tag>purple</tag>
      <tag>light</tag>
    </tags>
    <preview-image file="flowers-preview.png"/>

    <resources>
      <css file="flowers.css"/>
    </resources>

    <parameters>
      <parameter name="figure.title.placement" value="top"/>
    </parameters>
  </pdf>
</publishing-template>
```
Tip:
It is recommended to edit the template descriptor in Oxygen XML Editor/Author because it provides content completion and validation support.

Template Name and Description

Each template descriptor file requires a `<name>` element. This information is displayed as the name of the template in the transformation scenario dialog box.

Optionally, you can include a `<description>` and it displayed when the user hovers over the template in the transformation scenario dialog box.

```xml
<publishing-template>
  <name>Flowers</name>
  <description>Flowers themed light colored template</description>
  ...
</publishing-template>
```

Template Author

Optionally, you can include author information in the descriptor file and it displayed when the user hovers over the template in the transformation scenario dialog box. This information might be useful if users run into an issue or have questions about a certain template.

If you include the `<author>` element, a `<name>` is required and optionally you can include `<email>`, `<organization>`, and `<organizationUrl>`.

```xml
<publishing-template>
  ...
  <author>
    <name>John Doe</name>
    <email>jdoe@example.com</email>
    <organization>ACME</organization>
    <organizationUrl>http://www.example.com/jdoe</organizationUrl>
  </author>
  ...
</publishing-template>
```

PDF Element

The `<pdf>` element contains various details about the template and its resources that define the PDF output. It is a required element if you intend on using a DITA Map to PDF transformation scenario. The elements that are allowed in this `<pdf>` section specify the template tags (on page 1787), template preview image (on page...
Template Tags

The `<tags>` section provides meta information about the template (such as color theme). Each tag is displayed at the top of the Templates tab window in the transformation scenario dialog box and they help the user filter and find particular templates.

Template Preview Image

The `<preview-image>` element is used to specify an image that will be displayed in the transformation scenario dialog box. It provides a visual representation of the template to help the user select the right template. The image dimensions should be 200 x 115 pixels and the supported image formats are: JPEG, PNG, or GIF.

You can also include an `<online-preview-url>` element to specify the URL of a published sample of your template. This will display an Online preview icon in the bottom-right corner of the image in the transformation scenario dialog box and if the user clicks that icon, it will open the specified URL in their default browser.
Template Resources

The `<resources>` section of the descriptor file specifies a set of resources (CSS files) that are used to customize various components in the generated output. These resources will be copied to the output folder during the transformation process. At least one CSS file must be included (using the `<css>` element).

```xml
<publishing-template>
  ...
  <pdf>
    ...
    <resources>
      <css file="css/custom_styles.css"/>
      <css file="css/custom_fonts.css"/>
    </resources>
  </pdf>
</publishing-template>
```

**Note:**
All relative paths specified in the descriptor file are relative to the template root folder.

Transformation Parameters

You can also set one or more transformation parameters in the descriptor file.

```xml
<publishing-template>
  ...
  <pdf>
    ...
    <parameters>
      ...
      <parameter name="show.changes.and.comments" value="yes"/>
    </parameters>
  </pdf>
</publishing-template>
```

The following information can be specified in the `<parameters>` element:

**Parameter name**

The name of the parameter. It may be one of the transformation parameters listed in the Parameters tab of the DITA Map PDF - based on HTML5 & CSS transformation scenario or a DITA-OT PDF-based output parameter.
Note:
It is not recommended to specify an input/output parameter in the descriptor file (such as the input Map, DITAVAL file, or temporary directory).

Attention:
JVM arguments like -Xmx cannot be specified as a transformation parameter.

Parameter Value
The value of the parameter. It should be a relative path to the template root folder for file paths parameters.

Parameter Type
The type of the parameter: string or filepath. The string value is default.

After creating a publishing template (on page 1791) and adding it to the templates gallery (on page 1793), when you select the template in the transformation scenario dialog box, the Parameters tab will automatically be updated to include the parameters defined in the descriptor file. These parameters are displayed in italics.

XSLT Extension Points
The publishing templates support one or more XSLT extension points. They can be specified using the <xslt> element in the descriptor file using the following structure:

```xml
<publishing-template>
  ...
  <pdf>
    ...
    <xslt>
      <extension
        id="com.oxygenxml.pdf.css.xsl.merged2html5"
        file="xslt/merged2html5Extension.xsl"/>
      <extension
        id="com.oxygenxml.pdf.css.xsl.merged2merged"
        file="xslt/merged2mergedExtension.xsl"/>
    </xslt>
  </pdf>
  ...
</publishing-template>
```

For more information about the available extension points, see: XSLT Extensions for PDF Transformations (on page 1794).

Combining PDF and WebHelp Responsive Customizations in a Template Package
An Oxygen Publishing Template package can contain both a PDF and WebHelp Responsive customization in the same template package and you can use that same template in both types of transformations. The
A template descriptor file can define the customization for both types by including both a `<webhelp>` and `<pdf>` element and some of the resources can be reused. Resources referenced in elements in the `<webhelp>` element will only be used for WebHelp transformations, and resources referenced in the elements in the `<pdf>` element will only be used in PDF transformations.

```xml
<publishing-template>
  <name>Flowers</name>
  <description>Flowers themed light-colored template</description>

  <webhelp>
    <tags>
      <tag>purple</tag>
      <tag>light</tag>
    </tags>
    <preview-image file="flowers-preview.png"/>
    <resources>
      <css file="flowers-wh.css"/>
      <css file="flowers-page-styling.css"/>
    </resources>
    <parameters>
      <parameter name="webhelp.show.main.page.tiles" value="no"/>
      <parameter name="webhelp.show.main.page.toc" value="yes"/>
    </parameters>
  </webhelp>

  <pdf>
    <tags>
      <tag>purple</tag>
      <tag>light</tag>
    </tags>
    <preview-image file="flowers-preview.png"/>
    <resources>
      <css file="flowers-pdf.css"/>
      <css file="flowers-page-styling.css"/>
    </resources>
    <parameters>
      <parameter name="show.changes.and.comments" value="yes"/>
    </parameters>
  </pdf>
</publishing-template>
```

Related Information:

Publishing Template Package Contents for WebHelp Responsive Customizations (on page 1606)
How to Create a Publishing Template

To create a customization, you can start from scratch or from an existing template, and then adapt it according to your needs.

Creating a Publishing Template Starting from Scratch

To create a new Oxygen Publishing Template, follow these steps:

1. Create a folder that will contain all the template files.

2. In Oxygen XML Editor/Author, open the new document wizard (use File > New or the New toolbar button), then choose the Publishing Template Descriptor template.

   ![Figure 509. Choosing the Publishing Template Descriptor Document Template](image)

3. Save the .opt file into your customization directory.

4. Open the .opt file in the editor and customize it to suit your needs.

Creating a Publishing Template Starting from an Existing Template

If you are using a WebHelp Responsive or DITA Map PDF - based on HTML5 & CSS transformation, the easiest way to create a new Oxygen Publishing Template (on page 3221) is to select an existing template in the transformation scenario dialog box and use the Save template as button to save that template into a new template package that can be used as a starting point.

To create a new Oxygen Publishing Template, follow these steps:
1. Open the transformation scenario dialog box and select the publishing template you want to export and use as a starting point.

2. **Optional:** You can set one or more transformation parameters from the **Parameters** tab and the edited parameters will be exported along with the selected template. You will see which parameters will be exported in the dialog box that is displayed after the next step.

3. Click the **Save template as** button.

   **Step Result:** This opens a template package configuration dialog box that contains some options and displays the parameters that will be exported to your template package.

4. Specify a name for the new template.

5. **Optional:** Specify a template description.

6. **Optional:** The same publishing template package can contain both a WebHelp Responsive and PDF customization and you can use the same template in both types of transformations (DITA Map WebHelp Responsive or DITA Map to PDF - based on HTML5 & CSS). You can use the Include WebHelp customization and Include PDF customization options to specify whether your custom template will include both types of customizations.

7. **Optional:** For WebHelp Responsive customizations, you can select the Include HTML Page Layout Files option if you want to copy the default HTML Page Layout Files (on page 1622) in your template package. They are helpful if you want to change the structure of the generated HTML pages.

8. In the **Save as** field, specify the name and path of the ZIP file where the template will be saved.

   **Step Result:** A new ZIP archive will be created on disk in the specified location with the specified name.

9. Open the .opt file in the editor and customize it to suit your needs.

For more information about creating and customizing publishing templates, watch our video demonstration:

https://www.youtube.com/embed/zNmXfKWXw08

**Related Information:**

- Publishing Template Package Contents for PDF Customizations (on page 1785)
- Publishing Template Package Contents for WebHelp Responsive Customizations (on page 1606)

**How to Edit a Packed Publishing Template**

To edit an existing **Oxygen Publishing Template (on page 3221)** package, follow these steps:

1. Unzip the ZIP archive associated with the **Oxygen Publishing Template** in a separate folder.
2. Link the folder associated with the template in the **Project** view.
3. Using the **Project** view, you can modify the resources (CSS, JS, fonts) within the **Oxygen Publishing Template** folder to fit your needs.
4. Open the publishing template descriptor file (.opt extension) in the editor and modify it to suit your needs.

5. **Optional:** Once you finish your customization, you can archive the folder as a ZIP file.

---

**Related Information:**

- Publishing Template Package Contents for PDF Customizations *(on page 1785)*
- Publishing Template Package Contents for WebHelp Responsive Customizations *(on page 1606)*

---

**How to Use a Publishing Template in a PDF Transformation**

**From Oxygen XML Editor/Author**

A publishing template can be used for PDF output from the DITA Map PDF - based on HTML5 & CSS transformation scenario (or from the DITA PDF - based on HTML5 & CSS transformation scenario).

The Templates tab in the transformation scenario dialog box displays all the templates that are available in your template gallery. To use a particular template in the transformation scenario, simply select it from this tab and then continue configuring the transformation using the other tabs to suit your needs.

To add the publishing template to your templates gallery, follow these steps:

1. Open the transformation scenario dialog box by editing a DITA Map PDF - based on HTML5 & CSS transformation (or a DITA PDF - based on HTML5 & CSS transformation scenario).

2. In the Templates tab, click the Configure Publishing Templates Gallery link to.

   **Step Result:** This will open the preferences page.

3. Click the Add button and specify the location of your template directory.

   **Step Result:** Your template directory is now added to the Additional Publishing Templates Galleries list.

4. Click OK to return to the transformation scenario dialog box.

**Result:** All the templates contained in your template directory will be displayed in the preview pane along with all the built-in templates.

**From a Command Line**

You can use the `pdf.publishing.template` parameter to point to the *.opt (publishing template) file:

```bash
dita.bat
 --input=map\test.ditamap
 "-Dpdf.publishing.template=full_path_to_template_dir/my_template.opt"
 --format=pdf-css-html5
 ...
```
Or use the two parameters to indicate the folder containing the publishing templates and the name of the
publishing template file relative to that folder:

dita.bat
   --input=map\test.ditamap*
   "-Dpdf.publishing.template=full_path_to_template_dir"
   "-Dpdf.publishing.template.descriptor=my_template.opt"
   --format=pdf-css-html5
   ...

Tip:
You can also start the dita process by passing it a DITA OT Project File. Inside the project file you
can specify as parameters for the webhelp-responsive transformation type the WebHelp-related
parameters.

Related Information:
Transformation Parameters (on page 1773)

How to Share a Publishing Template

To share a publishing template with others, following these steps:

1. Copy your template in a new folder in your project.
2. Go to Options > Preferences > DITA > Publishing and add that new folder to the list.
3. Switch the option as the bottom of that preferences page to Project Options.
4. Share your project file (.xpr).

XSLT Extensions for PDF Transformations

Since PDF output is primarily obtained by running XSLT transformations over the DITA input files,
one customization method would be to override the default XSLT templates that are used by the PDF
transformation.

The pdf-css-html5 transformation type uses two stages to transform the merged DITA map (the one that
aggregates all the topics) to HTML5:

1. Stage 1: Makes some changes on the merged map (on page 1816) and the result is a modified merged
   map. This stage can be altered by implementing the com.oxygenxml.pdf.css.xsl.merged2merged
   XSLT extension point. This extension overrides the stylesheets found in the following folder: DITA-OT-
   DIR\plugins\com.oxygenxml.pdf.css\xsl\merged2merged.

   Note:
   Use this when you need to filter DITA content.
2. **Stage 2**: Transforms the merged map (on page 1816) to HTML5 and the result is a single HTML document. This stage can be altered by implementing the `com.oxygenxml.pdf.css.xsl.merged2html5` XSLT extension point. This extension overrides the stylesheets found in the following folder: `DITA-OT-DIR\plugins\com.oxygenxml.pdf.css\xsl\merged2html5`.

   **Note:**

   Use this when you need to change the HTML structures generated for a specific DITA element.

These extension points can be used either from a *Publishing Template* or a DITA-OT extension plugin.

**How to Use XSLT Extension Points for PDF Output from a Publishing Template**

The examples in this section demonstrate how to use XSLT extension points from an *Oxygen Publishing Template*.

**How to Style Codeblocks with a Zebra Effect**

Suppose you want your *codeblocks* to have a particular background color for one line, and another color for the next line. One advantage of this coloring technique is that you can clearly see when text from the *codeblock* is wrapped.

This effect can be done by altering the HTML5 output, creating a `div` for each line from the code block, then styling them.

To add this functionality using an *Oxygen Publishing Template*, follow these steps:

1. If you have not already created a Publishing Template, you need to create one. For details, see How to Create a Publishing Template (on page 1791).
2. Link the folder associated with the publishing template to your current project in the Project view.
3. Using the Project view, create an `xslt` folder inside the project root folder.
4. In this folder, create an XSL file (for example, named `merged2html5Extension.xsl`) with the following content:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<xs1:stylesheet xmlns:xs1="http://www.w3.org/1999/XSL/Transform"
    xmlns:xs="http://www.w3.org/2001/XMLSchema"
    exclude-result-prefixes="xs"
    version="2.0">

    <xs1:template match="*[contains(@class, ' pr-d/codeblock ')]">
        <div class="zebra">
            <xs1:analyze-string regex="\n" select="."/>
            <xs1:matching-substring/>
            <xs1:non-matching-substring>
        </div>
    </xs1:template>
</xs1:stylesheet>
```
5. Open the *template descriptor file (on page 1785)* associated with your *publishing template* (the .opt file) and set the XSLT stylesheet created in the previous step with the `com.oxygenxml.pdf.css.xsl.merged2html5` XSLT extension point:

```xml
<publishing-template>
  ...
  <pdf>
    ...
    <xslt>
      <extension
        id="com.oxygenxml.pdf.css.xsl.merged2html5"
        file="/xslt/merged2html5Extension.xsl"/>
    </xslt>
  </pdf>
</publishing-template>
```

6. Create a *css* folder in the publishing template directory. In this directory, save a custom CSS file with rules that style the *codeblock* structure. For example:

```css
div.zebra {
  font-family: courier, fixed, monospace;
  white-space: pre-wrap;
}

div.zebra > *:nth-of-type(odd) {
  background-color: silver;
}
```

7. Open the *template descriptor file (on page 1785)* associated with your *publishing template* (the .opt file) and reference your custom CSS file in the *resources* element:

```xml
<publishing-template>
  ...
  <pdf>
    ...
    <resources>
      <css file="css/custom.css"/>
    </resources>
  </pdf>
</publishing-template>
```

8. Edit the DITA Map PDF - based on HTML5 & CSS transformation scenario.
9. In the **Templates** tab, click the **Choose Custom Publishing Template** link and select your template.
10. Click **OK** to save the changes and run the transformation scenario.

### How to Remove the Related Links Section

Suppose that you want the *related links* sections to be removed from the PDF output.

To add this functionality using an *Oxygen Publishing Template*, follow these steps:

1. If you have not already created a Publishing Template, you need to create one. For details, see *How to Create a Publishing Template (on page 1791)*.
2. Link the folder associated with the publishing template to your current project in the **Project** view.
3. Using the **Project** view, create an **xslt** folder inside the project root folder.
4. In this folder, create an XSL file (for example, named `merged2mergedExtension.xsl`) with the following content:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    xmlns:xs="http://www.w3.org/2001/XMLSchema"
    exclude-result-prefixes="xs"
    version="2.0">

    <xsl:template match="*[contains(@class, ' topic/related-links ')]">
        <!-- Remove. -->
    </xsl:template>

</xsl:stylesheet>
```

5. Open the **template descriptor file (on page 1785)** associated with your publishing **template** (the `.opt` file) and set the XSLT stylesheet created in the previous step with the `com.oxygenxml.pdf.css.xsl.merged2merged` XSLT extension point:

```xml
<publishing-template>
    ...
    <pdf>
        ...
        ...
        <xslt>
            <extension
                id="com.oxygenxml.pdf.css.xsl.merged2merged"
                file="xslt/merged2mergedExtension.xsl"/>
        </xslt>
    </pdf>
</publishing-template>
```

6. Edit the **DITA Map PDF - based on HTML5 & CSS** transformation scenario.
7. In the **Templates** tab, click the **Choose Custom Publishing Template** link and select your template.
8. Click **OK** to save the changes and run the transformation scenario.
How to Wrap Words in Markup

Suppose you want compound words that contain hyphens (or any other criteria) to be wrapped with inline elements (such as the HTML `<code>` element).

To add this functionality using an *Oxygen Publishing Template*, follow these steps:

1. If you have not already created a Publishing Template, you need to create one. For details, see *How to Create a Publishing Template*(on page 1791).

2. Link the folder associated with the publishing template to your current project in the Project view.

3. Using the Project view, create an `xslt` folder inside the project root folder.

4. In this folder, create an XSL file (for example, named `merged2htmlExtension.xsl`) with the following content:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    xmlns:xs="http://www.w3.org/2001/XMLSchema"
    exclude-result-prefixes="xs"
    version="2.0">

    <xsl:template match="text()">

        <xsl:variable name="txt">
            <xsl:next-match/>
        </xsl:variable>

        <xsl:analyze-string regex="(\w|\-)+" select="$txt">
            <xsl:matching-substring>
                <!-- A word -->
                <xsl:choose>
                    <xsl:when test="contains(., '-')">
                        <!-- A compound word -->
                        <code class="compound-word">
                            <xsl:value-of select="."/>
                        </code>
                    </xsl:when>
                    <xsl:otherwise>
                        <!-- A simple word -->
                        <xsl:value-of select="."/>
                    </xsl:otherwise>
                </xsl:choose>
            </xsl:matching-substring>

            <xsl:non-matching-substring>
                <!-- Not a word -->
            </xsl:non-matching-substring>
        </xsl:analyze-string>
    </xsl:template>
</xsl:stylesheet>
```
5. Open the template descriptor file (on page 1785) associated with your publishing template (the .opt file) and set the XSLT stylesheet created in the previous step with the com.oxygenxml.pdf.css.xsl.merged2merged XSLT extension point:

```xml
<publishing-template>
  ...
  <pdf>
    ...
    <xslt>
      <extension
        id="com.oxygenxml.pdf.css.xsl.merged2merged"
        file="xslt/merged2mergedExtension.xsl"/>
    </xslt>
  </pdf>
</publishing-template>
```

6. Edit the DITA Map PDF - based on HTML5 & CSS transformation scenario.

7. In the Templates tab, click the Choose Custom Publishing Template link and select your template.

8. Click OK to save the changes and run the transformation scenario.

**How to Convert Definition Lists into Tables**

Suppose you want your definitions lists (<dl>) to be displayed as tables in your PDF output.

To add this functionality using an Oxygen Publishing Template, follow these steps:

1. If you have not already created a Publishing Template, you need to create one. For details, see How to Create a Publishing Template (on page 1791).

2. Link the folder associated with the publishing template to your current project in the Project view.

3. Using the Project view, create an xslt folder inside the project root folder.

4. In this folder, create an XSL file (for example, named merged2html15Extension.xsl) with the following content:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  exclude-result-prefixes="xs"
  version="2.0">

  <xsl:template match="*[contains(@class, ' topic/dl ')]">
    <xsl:call-template name="setaname"/>
  </xsl:template>
</xsl:stylesheet>
```
<xsl:template match="*[contains(@class, ' topic/dlentry ')]">
  <!-- Wrap in a table row -->
  <tr>
    <xsl:call-template name="commonattributes"/>
    <xsl:call-template name="setidaname"/>
    <xsl:apply-templates/>
  </tr>
</xsl:template>

<xsl:template match="*[contains(@class, ' topic/dd ')] | *[contains(@class, ' topic/dt ')]">
  <!-- Wrap in a cell -->
  <td>
    <xsl:call-template name="commonattributes"/>
    <xsl:call-template name="setidaname"/>
    <xsl:apply-templates select="..
      *[contains(@class, ' ditaot-d/ditaval-startprop ')]" mode="out-of-line"/>
    <xsl:apply-templates/>
    <xsl:apply-templates select="..
      *[contains(@class, ' ditaot-d/ditaval-endprop ')]" mode="out-of-line"/>
  </td>
</xsl:template>

</xsl:stylesheet>
5. Open the template descriptor file (on page 1785) associated with your publishing template (the .opt file) and set the XSLT stylesheet created in the previous step with the com.oxygenxml.pdf.css.xsl.merged2html5 XSLT extension point:

```
<publishing-template>
  ...
  <pdf>
  ...
  <xslt>
    <extension
      id="com.oxygenxml.pdf.css.xsl.merged2html5"
      file="xslt/merged2html5Extension.xsl"/>
  </xslt>
</publishing-template>
```

6. Edit the DITA Map PDF - based on HTML5 & CSS transformation scenario.

7. In the Templates tab, click the Choose Custom Publishing Template link and select your template.

8. Click OK to save the changes and run the transformation scenario.

### How to Display Footnotes Below Tables

In your PDF output, you may want to group all the footnotes contained in a table just below it instead of having them displayed at the bottom of the page.

To add this functionality, use an Oxygen Publishing Template and follow these steps:

1. If you have not already created a Publishing Template, you need to create one. For details, see How to Create a Publishing Template (on page 1791).
2. Link the folder associated with the publishing template to your current project in the Project view.
3. Using the Project view, create an xslt folder inside the project root folder.
4. In the newly created folder, create an XSL file (for example, named merged2mergedExtension.xsl) with the following content:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
                 xmlns:xs="http://www.w3.org/2001/XMLSchema"
                 exclude-result-prefixes="xs"
                 version="2.0">

    <!-- Match only top level tables (i.e tables that are not nested in other tables), that have footnotes. -->
    <xsl:template match="
        *[contains(@class, 'topic/table')] 
        [not(ancestor::*[contains(@class, 'topic/table')])]
        //*[contains(@class, 'topic/fn')]]"
    <xsl:next-match>
```
5. Open the template descriptor file (on page 1785) associated with your publishing template (the .opt file) and set the XSLT stylesheet created in the previous step with the com.oxygenxml.pdf.css.xsl.merged2merged XSLT extension point:

   <publishing-template>
     ...
     <pdf>
       ...
       <xslt>
         <extension
           id="com.oxygenxml.pdf.css.xsl.merged2merged"/>
           file="xslt/merged2mergedExtension.xsl"
         </extension>
       </xslt>
     </pdf>
     ...
   </publishing-template>

6. Create a css folder in the publishing template directory. In this directory, save a custom CSS file with rules that style the glossary structure. For example:
/* Customize footnote calls, inside the table. */
*[outputclass ~='table-fn-call'] {
  color: black;
  text-decoration: none;
  font-size: smaller;
  vertical-align: super;
  line-height: none;
}

/* Hide the "on page .." text that follows xrefs. */
*[outputclass ~='table-fn-call']:after {
  display: none;
}

/* Customize the list containing all the table footnotes. */
*[outputclass ~='table-fn-container'] {
  border-top: 1pt solid black;
}

/* Customize footnotes display, below the table. */
*[outputclass ~='table-fn'] {
  font-size: 1em;
}
*[outputclass ~='table-fn']::marker {
  font-size: 1em;
}

7. Open the template descriptor file (on page 1785) associated with your publishing template (the .opt file) and reference your custom CSS file in the resources element:

```xml
<publishing-template>
  ...
  <pdf>
    ...
    <resources>
      <css file="css/custom.css"/>
    </resources>
  </pdf>
</publishing-template>
```

8. Edit the DITA Map PDF - based on HTML5 & CSS transformation scenario.
9. In the Templates tab, click the Choose Custom Publishing Template link and select your template.
10. Click OK to save the changes and run the transformation scenario.
How to Wrap Scientific Numbers in Tables Cells

In your PDF output, you may need to wrap scientific numbers on two lines when they are included in table cells.

To add this functionality, use an *Oxygen Publishing Template* and follow these steps:

1. If you have not already created a Publishing Template, you need to create one. For details, see How to Create a Publishing Template (on page 1791).
2. Link the folder associated with the publishing template to your current project in the **Project** view.
3. Using the **Project** view, create an xslt folder inside the project root folder.
4. In the newly created folder, create an XSL file (for example, named `merged2html5Extension.xsl`) with the following content:

   ```xml
   <?xml version="1.0" encoding="UTF-8"?>
   <xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    xmlns:xs="http://www.w3.org/2001/XMLSchema"
    exclude-result-prefixes="xs"
    version="2.0">

   <!-- Matches text from table cells. -->
   <xsl:template match="*[contains(@class, ' topic/entry ') ]/text()">
     <xsl:analyze-string select="." regex="[0-9]\.[0-9]{2}e-[0-9]{2}"
       select="concat(substring-before(., 'e'), 'e', substring-after(., 'e'))"/>
     <xsl:non-matching-substring />
   </xsl:template>

   </xsl:stylesheet>
   
   <!-- Matches text from table cells. -->
   <xsl:template match="*[contains(@class, ' topic/entry ') ]/text()">
     <xsl:analyze-string select="." regex="[0-9]\.[0-9]{2}e-[0-9]{2}"
       select="concat(substring-before(., 'e'), 'e', substring-after(., 'e'))"/>
     <xsl:non-matching-substring />
   </xsl:template>

5. Open the **template descriptor file** (on page 1785) associated with your publishing template (the .opt file) and set the XSLT stylesheet created in the previous step with the `com.oxygenxml.pdf.css.xsl.merged2html5` XSLT extension point:

   ```xml
   <publishing-template>
     ...
   </pdf>
   ```
6. Edit the DITA Map PDF - based on HTML5 & CSS transformation scenario.
7. In the Templates tab, click the Choose Custom Publishing Template link and select your template.
8. Click OK to save the changes and run the transformation scenario.

How to Use a Bullet for Tasks that Contain a Single Step

If a DITA Task document only contains one list item (a single `<step>` element), you probably want it to be rendered the same as an unordered list (displayed with a bullet instead of a number), as in the following example:

```xml
...  
<steps>
  <step>
    <cmd>My single step</cmd>
  </step>
</steps>
...  
```

Normally, the output will be rendered as:

1. The step

instead of:

- The step

To change the default rendering so that a single step will be rendered with a bullet instead of a number, use an Oxygen Publishing Template and follow these steps:

1. If you have not already created a Publishing Template, you need to create one. For details, see How to Create a Publishing Template (on page 1791).
2. Link the folder associated with the publishing template to your current project in the Project view.
3. Using the Project view, create an xslt folder inside the project root folder.
4. In the newly created folder, create an XSL file (for example, named merged2html5Extension.xsl) with the following content:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
                xmlns:xs="http://www.w3.org/2001/XMLSchema"
                exclude-result-prefixes="xs"
                version="2.0">
5. Open the template descriptor file (on page 1785) associated with your publishing template (the .opt file) and set the XSLT stylesheet created in the previous step with the com.oxygenxml.pdf.css.xsl.merged2html5 XSLT extension point:

```xml
<publishing-template>
  ...
  <pdf>
    ...
    <xslt>
      <extension
        id="com.oxygenxml.pdf.css.xsl.merged2html5"/>
        file="xslt/merged2html5Extension.xsl"
    </xslt>
  </pdf>
</publishing-template>
```

6. Create a css folder in the publishing template directory. In this directory, save a custom CSS file with rules that style the glossary structure. For example:

```css
*{outputclass =~ "single"} {  
  list-style-type:circle !important;  
  margin-left:2em;  
}
```

7. Open the template descriptor file (on page 1785) associated with your publishing template (the .opt file) and reference your custom CSS file in the resources element:

```xml
<publishing-template>
  ...
  <pdf>
    ...
    <resources>
      <css file="css/custom.css"/>
      </resources>
  </pdf>
</publishing-template>
```

8. Edit the DITA Map PDF - based on HTML5 & CSS transformation scenario.

9. In the Templates tab, click the Choose Custom Publishing Template link and select your template.

10. Click OK to save the changes and run the transformation scenario.
How to Change the Critical Dates Format

By default, the dates are entered in a **YYYY-MM-DD** format (where **YYYY** is the year, **MM** is the number of the month, and **DD** is the number of the day). You can change the format (for example, to something like *January 1, 2020*) using an XSLT extension.

To add this functionality, use an *Oxygen Publishing Template* and follow these steps:

1. If you have not already created a Publishing Template, you need to create one. For details, see **How to Create a Publishing Template (on page 1791)**.
2. Link the folder associated with the publishing template to your current project in the **Project** view.
3. Using the **Project** view, create an **xslt** folder inside the project root folder.
4. In the newly created folder, create an XSL file (for example, named **merged2mergedExtension.xsl**) with the following content:

   ```xml
   <?xml version="1.0" encoding="UTF-8"?>
   <xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
                   xmlns:xs="http://www.w3.org/2001/XMLSchema"
                   exclude-result-prefixes="xs"
                   version="2.0">

   <xsl:template match="*[contains(@class, 'topic/created')]/@date | *[contains(@class, 'topic/revised')]/@modified" mode="copy-meta">
     <xsl:attribute name="{name()}">
       <xsl:value-of select="format-date(., '[MNn] [D01], [Y0001]')"/>
     </xsl:attribute>
   </xsl:template>

   </xsl:stylesheet>
   ``

5. Open the **template descriptor file (on page 1785)** associated with your publishing template (the *.opt* file) and set the XSLT stylesheet created in the previous step with the **com.oxygenxml.pdf.css.xsl.merged2merged** XSLT extension point:

   ```xml
   <publishing-template>
     ...
     <pdf>
       ...
       <xslt>
         <extension id="com.oxygenxml.pdf.css.xsl.merged2merged"/>
         file="xslt/merged2mergedExtension.xsl"
       </xslt>
   </publishing-template>
   
   6. Edit the **DITA Map PDF - based on HTML5 & CSS** transformation scenario.
7. In the Templates tab, click the Choose Custom Publishing Template link and select your template.
8. Click OK to save the changes and run the transformation scenario.

Related information

Formatting Dates and Times in XSLT

**How to Remove Links from Terms**

Your topics might contain multiple references to the same `<term>`. These terms can further be explained in the glossary. In this case, you may want to only keep the first occurrence of this term to be a link to the glossary and display the other terms as text.

To add this functionality, use an Oxygen Publishing Template and follow these steps:

1. If you have not already created a Publishing Template, you need to create one. For details, see How to Create a Publishing Template *(on page 1791).*
2. Link the folder associated with the publishing template to your current project in the Project view.
3. Using the Project view, create an `xslt` folder inside the project root folder.
4. In the newly created folder, create an XSL file (for example, named `merged2html5Extension.xsl`) with the following content:

```xml
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    xmlns:xs="http://www.w3.org/2001/XMLSchema"
    exclude-result-prefixes="xs"
    version="2.0">

    <xsl:template match="*[contains(@class, ' topic/term ')]" name="topic.term">
      <!-- Save the current @href value -->
      <xsl:variable name="current-href" select="@href"/>
      <!-- Get the closest parent topic -->
      <xsl:variable name="closest-parent"
        select="ancestor::*[contains(@class, ' topic/topic ')][1]"/>
      <!-- Get the first <term> having the same href -->
      <xsl:variable name="first-term-with-same-href"
        select="($closest-parent//*[contains(@class, ' topic/term ')][@href="$current-href"])[1]"/>
      <!-- Call the HTML5 default template -->
      <xsl:variable name="result">
        <xsl:next-match/>
      </xsl:variable>
      <!-- Call the copy template that will remove the links -->
      <xsl:apply-templates select="$result" mode="remove-extra-links"/>

    </xsl:template>

</xsl:stylesheet>
```
5. Open the template descriptor file (on page 1785) associated with your publishing template (the .opt file) and set the XSLT stylesheet created in the previous step with the

   <publishing-template>
     ...
     <pdf>
       ...
     </pdf>
     <xslt>
       <extension
id="com.oxygenxml.pdf.css.xsl.merged2html5" />
       file="xslt/merged2html5Extension.xsl"
     </xslt>
   </publishing-template>

6. Edit the DITA Map PDF - based on HTML5 & CSS transformation scenario.

7. In the Templates tab, click the Choose Custom Publishing Template link and select your template.

8. Click OK to save the changes and run the transformation scenario.
How to Use XSLT Extension Points for PDF Output from a DITA-OT Plugin

The examples in this section demonstrate how to use XSLT extension points from a DITA-OT plugin.

Instead of directly adding plugins inside the embedded DITA-OT, it is highly recommended to use an external Oxygen Publishing Engine so that you will not lose any of your customizations anytime you upgrade the product in the future.

You just need to follow these steps before starting your custom DITA-OT plugins:

1. Download the Oxygen Publishing Engine and unzip it inside a folder where you have full write access.
2. Create your custom plugin(s) inside the DITA-OT-DIR\plugins\ folder.
3. Go to Options > Preferences > DITA, set the DITA Open Toolkit option to Custom, and specify the path to the unzipped folder.

Warning:
The path must end with: oxygen-publishing-engine-3.x.

How to Style Codeblocks with a Zebra Effect

Suppose you want your codeblocks to have a particular background color for one line, and another color for the next line. One advantage of this coloring technique is that you can clearly see when text from the codeblock is wrapped.

This effect can be done by altering the HTML5 output, creating a <div> for each line from the code block, then styling them.

To add this functionality using a DITA-OT plugin, follow these steps:

1. In the DITA-OT-DIR\plugins\ folder, create a folder for this plugin (for example, com.oxygenxml.pdf.custom.codeblocks).
2. Create a plugin.xml file (in the folder you created in step 1) that specifies the extension point and your customization stylesheet. For example:

```xml
<plugin id="com.oxygenxml.pdf.custom.codeblocks">
  <feature extension="com.oxygenxml.pdf.css.xsl.merged2html5" file="custom_codeblocks.xsl"/>
</plugin>
```
3. Create your customization stylesheet (for example, custom_codeblocks.xsl) with the following content:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  exclude-result-prefixes="xs"
  version="2.0">
```
4. Use the Run DITA-OT Integrator transformation scenario (on page 1455) found in the DITA Map section in the Configure Transformation Scenario(s) dialog box.

5. Create a custom CSS file with rules that style the codeblock structure. For example:

```css
div.zebra {
    font-family: courier, fixed, monospace;
    white-space: pre-wrap;
}
div.zebra > *:nth-of-type(odd){
    background-color: silver;
}
```

6. Edit a DITA Map PDF - based on HTML5 & CSS transformation scenario and reference your custom CSS file (using the args.css parameter).

7. Run the transformation scenario.

How to Remove the Related Links Section

Suppose you want the related links sections to be removed from the PDF output.

To add this functionality using a DITA-OT plugin, follow these steps:

1. In the DITA-OT-DIR\plugins\ folder, create a folder for this plugin (for example, com.oxygenxml.pdf.custom.codeblocks).
2. Create a plugin.xml file (in the folder you created in step 1) that specifies the extension point and your customization stylesheet. For example:

```xml
<plugin id="com.oxygenxml.pdf.custom.related.links">
    <feature extension="com.oxygenxml.pdf.css.xsl.merged2merged" file="custom_related_links.xsl"/>
</plugin>
```
3. Create your customization stylesheet (for example, custom_related_links.xsl) with the following content:
4. Use the Run DITA-OT Integrator transformation scenario (on page 1455) found in the DITA Map section in the Configure Transformation Scenario(s) dialog box.

5. Run the DITA Map PDF - based on HTML5 & CSS transformation scenario.

How to Use Custom Parameters in XSLT Stylesheets

Suppose you want to add an attribute with a custom value inside a `<div>` element.

To add this functionality using a DITA-OT plugin, follow these steps:

1. In the DITA-OT-DIR\plugins\ folder, create a folder for this plugin (for example, com.oxygenxml.pdf.css.param).

2. Create a plugin.xml file (in the folder you created in step 1) that specifies the extension points, your parameter file, and your customization stylesheet. For example:

```xml
<plugin id="com.oxygenxml.pdf.css.param">
  <feature extension="com.oxygenxml.pdf.css.xsl.merged2html5.parameters" file="params.xml"/>
  <feature extension="com.oxygenxml.pdf.css.xsl.merged2html5" file="custom.xsl"/>
</plugin>
```

**Note:**

The com.oxygenxml.pdf.css.xsl.merged2html5 extension point can also be called from a Publishing Template.

3. Create a params.xml file that specifies the name of the custom attribute with the following content:

```xml
<dummy xmlns:if="ant:if">
  <param name="custom-param" expression="${custom.param}" if:set="custom.param"/>
</dummy>
```

4. Create your customization stylesheet (for example, custom.xsl) with the following content:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  xmlns:xs="http://www.w3.org/2001/XMLSchema" version="2.0">
  <xsl:template match="*[contains(@class, ' topic/related-links ')]">
    <!-- Remove. -->
  </xsl:template>
</xsl:stylesheet>
```
5. Use the Run DITA-OT Integrator transformation scenario (on page 1455) found in the DITA Map section in the Configure Transformation Scenario(s) dialog box.

6. Duplicate the DITA Map PDF - based on HTML5 & CSS transformation scenario, then in the Parameters tab click New to create a new parameter (e.g. named custom.param with the value of customValue).

7. Run the transformation scenario.

Related information
Adding parameters to existing XSLT steps

DITA-OT Extension Points

The DITA-OT CSS-based PDF Publishing Plugin supports DITA-OT extension points that can be used to expand the functionality of the transformation. The extension points are defined in the plugin.xml file. For more information, see DITA Open Toolkit Extension Points.

Related Information:
XSLT Extensions for PDF Transformations (on page 1794)

How to Contribute a Custom CSS to the Transformation from a DITA-OT Plugin

This topic is intended for publishing architects/developers that need to deploy a customized DITA-OT.

Usually, the CSS styles can be passed to the transformation by referencing the CSS files using the args.css parameter. However, there are cases where you want to add some sort of "built-in" CSS that is applied in conjunction with the publishing template or CSS files referenced in the transformation.
For this, you need to use the `com.oxygenxml.pdf.css.init` extension point and set the value of the `extension.css` ANT property to the path of the custom CSS file:

1. In your `plugin.xml` file, add:

   ```xml
   <feature extension="com.oxygenxml.pdf.css.init" file="init.xml"/>
   ```

2. Create a file named `init.xml` with the following ANT content:

   ```xml
   <root>
   <property name="extension.css" value="${dita.plugin.[com.my.plugin.id].dir}/css/my-custom.css"/>
   <!-- add here more init stuff if needed -->
   </root>
   ```

   **Note:**
   The name of the root element does not matter. The content of this element will be copied in an initialization template.

   **Important:**
   Make sure all file references begin with the ANT variable that is expanded to the base directory of your plugin.

---

**Customizing PDF Output Using CSS**

The publishing process is driven by a *customization CSS*.

**Tip:**
If you use the default Chemistry processor in **Oxygen XML Editor/Author**, you can use LESS instead of CSS. In this case, the customization files should have the `.less` extension.

To change the styling of the output for the **DITA Map PDF - based on HTML5 & CSS** and the **DITA PDF - based on HTML5 & CSS** transformation scenarios:

1. Create the CSS file that will contain all of your customizations. It is recommended to create this file in your project directory so you can edit it easily.
2. Add your custom CSS rules. As a good starting point you can:
   - Check the various topics in this section for assistance with specific types of customizations.
   - Use the **Oxygen Styles Basket** (*on page 1815*) to generate basic selectors for common content.
3. For linking the CSS file, you have two options:
Create a publishing template, create the customization CSS file inside the template folder, and link it to the publishing template descriptor. For assistance, see Publishing Templates (on page 1603).

Choose an existing publishing template, then edit the scenario and set the full path to the custom CSS file as the value of the `args.css` parameter. The rules from custom CSS will override the rules from the template CSS files.

4. Run the transformation scenario.

**Tip:**
For more information and some tips in regard to publishing DITA documents to PDF using CSS, watch our Webinars:

- Transforming DITA documents to PDF using CSS, Part 1 – Page Definitions, Cover Page and PDF Metadata

**Oxygen Styles Basket**

The Oxygen Styles Basket is a handy free-to-use web-based visual tool that helps you to fine-tune the CSS file that is used to customize your PDF or WebHelp output. You simply pick and mix aspects from galleries to add and adjust the rules in your custom CSS file.

It is based on galleries that you can use to pick and mix styling aspects to create a custom look and feel. Various different types of aspects can be selected to be integrated in the CSS stylesheet (such as fonts, tables, lists, spacing, code). It is also possible to re-upload a previously generated CSS for further customization.

You can see the results of your changes in the Preview pane or you can click the See Results toolbar button to see the results in a generated preview version of either PDF or WebHelp output. When you are finished with your customization, you can Download the results as a CSS file or a Publishing Template Package that you can use in your transformation process.
Figure 510. Oxygen Styles Basket Interface

Resources

For more information about the Oxygen Styles Basket, see the following resources:

- Video: Introducing the New Oxygen Styles Basket
- Webinar: Using Oxygen Styles Basket to Create CSS Customization from Scratch

Debugging the CSS

If you notice that some of the CSS properties were not applied as expected, some of the tips offered in this topic might help you with the debugging process.

Merged Map File

Depending on the type of transformation, one or more merged map files are created at some point during the transformation stages. These files could be used to help debug unexpected results.

1. The first thing you should try is to check the file structure of the merged map file. This can be found in the out/pdf-css directory and it has the .merged.html file extension (you will also find a .merged.xml file that aggregates the entire ditamap structure). You can open the HTML files in Oxygen XML Editor/Author to examine the structure. Optionally, you can use the pretty print feature (Format and Indent) to make the structure easier to read.

2. Check that the CSS selectors are written correctly against the document structure.

3. If you still cannot identify the problem, then inspect how the styles are applied (you can try any of the methods listed below).

Inspecting the Applied Styles Using the Chrome Browser

To inspect the applied CSS styles using Chrome:
1. Open the file ending in `.merged.html`.
2. Click on the element you want to inspect.
3. Activate the **Chrome Developer Tools** by using `More Tools > Developer Tools`, or press `CTRL + SHIFT + I`.
4. Activate the **Rendering pane** by using `More Tools > Rendering`.
5. In the **Rendering pane**, select **print** from the Emulate CSS media section. This will activate the CSS selectors enclosed in `@media print {...}`.

![Chrome Developer Tools](image)

![Rendering pane](image)

![Emulate CSS media](image)
Note:
This allows you to debug the styling of elements, table of contents, and index, but not the styles of the page margin boxes (headers, footers) or page breaks.

Tip:
In the left pane of the Developer Tools interface, you can inspect elements and their styles in the Elements tab. You can click on any of the links to display the applied CSS files in the Styles tab in the right pane. Editing the styles in that pane results in a live preview of how the change will affect the output.

CAUTION:
Do not modify the built-in rules directly in the CSS files from the Oxygen XML Editor/Author installation. Instead, copy the rules to your own customization CSS.

Inspecting the Applied Styles Using Oxygen XML Editor/Author

To inspect styles:

1. In Oxygen XML Editor/Author, open the file ending in .merged.html.
2. [Optional] From the Styles toolbar, choose the + Print Ready entry. This will activate certain CSS selectors enclosed in @media print {..}.
3. Click on the element you want to style. Use the Inspect Styles action from the Contextual Menu. A specialized CSS Inspector view will show the built-in CSS rules.

Tip:
With this file open in Author mode, it might be helpful to switch the Tags Display Mode to Full Tags with Attributes. You might be able to identify the selector you need to style without using the CSS Inspector view.

Note:
This allows you to debug styling of elements, but not of the page margin boxes (headers, footers) or page breaks.

CAUTION:
Do not modify the built-in rules directly in the CSS files from the Oxygen XML Editor/Author installation. Instead, copy the rules to your own customization CSS.

Other techniques

These are some other techniques you may find useful:
• Add background and borders properties to the specific CSS rule. If they do not appear in the output then there is a problem with the rule selector.
• Try to use the `!important` notation to the property that is not applied, or make the selector more specific (you can add more parent selectors).
• To figure out how the elements are mapped to PDF, you can use this fragment in the customization CSS:

```css
* {
    border: 1pt solid blue !important;
}

*:before(1000) {
    content: oxy_name() !important;
    color: orange;
}

*:before(999) {
    content: "[ class= "attr(class) "]" !important;
    color: orange;
}
```

This will show the element name, its class attribute, and will paint a blue border around each of the elements in the output. It will not show the page margin boxes or some content elements that were hidden.

### How to Speed up CSS Development and Debugging

You can speed up your CSS development considerably by not invoking the entire pipeline of transforming your DITA maps to PDF. Instead, you can use the merged map (*on page 1816*) and transform it directly to PDF.

1. Transform your DITA Map to PDF using the DITA Map PDF - based on HTML5 & CSS transformation scenario.
2. Open the merged file (*on page 1816*) (*.merged.html) that is located in the output directory in the editor.
3. Configure an XML to PDF transformation with CSS scenario. Do not set CSS files here since the merged file already contains pointers to the stylesheets. This scenario uses the Chemistry CSS processor.
4. Optional: Enable the output of the CSS processor using the following preferences page: Options > Preferences > XML > PDF Processors > CSS Processor.

Now you can make incremental changes to the CSS stylesheet and quickly see the results by transforming the merged file directly.
Fastpath:
If your changes involve only element styling, with no specific paged media CSS rules and properties, you can simply open the merged file in a browser (such as Chrome or Firefox) and refresh at each CSS change, as shown in: Debugging the CSS (on page 1816).

How to Write XPath Expressions

This topic contains some guidelines for writing XPath expressions. They are used to extract the content from the merged DITA map document.

This is an example where the product name meta-information is placed before the front page title:

```xml
*[class="front-page/front-page-title"]:before {
  text-align: left;
  content: oxy_xpath("//*[contains(@class, 'topic/prodname')]/text()")
  display: block
}
```

Notes:

- Do not use the DITA element names directly. You must use the DITA @class attribute instead, as these attributes are propagated to the merged elements (including HTML `<div>` elements) while the element names can be lost. By using the class selectors, you also cover DITA specializations.
- Use the `[1]` XPath predicate to select the first value from the document. For example, `oxy_xpath("//*[contains(@class, 'topic/prodname')]/text()")`. The meta-information might be copied multiple times in the output, inherited by the `<topicref>` elements, so you can get many more values than expected.
- Do not use strings as values for the pseudo-elements content, as the string values are not supported for pseudo-elements. Instead, use the XPath directly.
- Use the Oxygen XPath Builder view to test the XPath expressions.

Related Information:
How to Debug XPath Expressions (on page 1820)

How to Debug XPath Expressions

You can use the content extracted from the document using the `oxy_xpath` function in your pseudo-elements (`:before`, `:after`) or in `string-set` variables.

For example, the following XPath finds the publication author, set in the DITA map:
To debug an XPath expression:

1. Read the XPath Expressions Guidelines (on page 1820).
2. Begin by transforming your document using your customization CSS.
3. In the output folder, you will find a [MAP_NAME].merged.html file (or if you are using the DITA Map PDF - based on HTML5 & CSS transformation, a [MAP_NAME].merged.html file).
4. Open the merged file in the Oxygen XML Editor/Author.
5. Activate the XPath Builder view (Window > Show View > XPath/XQuery Builder).
6. Paste your XPath expression and click the Execute XPath button. Check if it returns the expected results.

The XPath builder has a function that allows it to display the document path of the current element from the editor (⚙️. Settings drop-down menu > ⬇️ Update on cursor move). Alternatively, you can right-click the element in the merged document and select the Copy XPath action, then paste it in the XPath builder.

Related Information:
XPath Builder Documentation
XPath Examples (w3schools.com)

Default Page Definitions
All page definitions are found in: [PLUGIN_DIR]css/print/p-pages-and-headers.css.

Note:
This is listed solely for illustration purposes, as the plugin might use something different.

There are page definitions for the default page, chapter page, table of contents page, front matter page, back matter page, index page, large tables page, and blank page.
Default Page

The default page imposes a header that contains the publication title, chapter, and section title. They alternate on the left or right side of the page:

```xml
@page :left {
  @top-left {
    content: string(maptitle) string(parttitle) string(chaptertitle) string(sectiontitle) " | " counter(page);
  }
}

@page :right {
  @top-right {
    content: string(maptitle) string(parttitle) string(chaptertitle) string(sectiontitle) " | " counter(page);
  }
}
```

Tip:

To override the default rules defined for named pages (such as chapter or table of contents), you need to use more specific page rules that contain the page name:

```xml
@page :left, table-of-contents: left, chapter: left {
  @top-left {
    content: "...";
  }
}

@page :right, table-of-contents: right, chapter: right {
  @top-right {
    content: "...";
  }
}
```

Chapter Page

This is inherited from the default page. The chapter page is associated with the topics marked as chapters, usually direct children of the map. It clears the header from the first page of each chapter. If you need to add other information to the chapter headers, make sure you override these rules in your CSS:

```xml
@page chapter{
  /* Currently inherit from the default page. */
}
```
/* No headers on the chapter first page. */
@page chapter:first:
  @top-left {
    content: none;
  }

@page chapter:first:
  @top-right {
    content: none;
  }

/* Do not put a header on the chapter first page. */
@page table-of-contents:
  @top-left {
    content: string(toc-header) " | " counter(page, lower-roman);
  }

/* Do not put a header on the first page of the TOC */
@page table-of-contents:first:
  @top-left {
    content: none;
  }

  @top-right {
    content: string(toc-header) " | " counter(page, lower-roman);
  }

Table of Contents Page

The table of content page. It clears the headers and uses a lower roman page number in the header.
Front Matter and Back Matter Page

The bookmap front matter and back matter page. It clears the headers.

```xml
@page matter-page {
  @top-left-corner    { content: none; }
  @top-center         { content: none; }
  @top-right-corner   { content: none; }
  @bottom-left-corner { content: none; }
  @bottom-left        { content: none; }
  @bottom-center      { content: none; }
  @bottom-right       { content: none; }
  @bottom-right-corner{ content: none; }
}
@page matter-page: left {
  @top-left           { content: counter(page, lower-roman); }
}
@page matter-page: right {
  @top-right          { content: counter(page, lower-roman); }
}
```

Index Page

The page that contains the index terms (appears only if there are such items in your topics). It uses a lower roman page number in the footer:

```xml
@page index {
  @top-left-corner    { content: none; }
  @top-left           { content: none; }
  @top-right          { content: none; }
  @top-right-corner   { content: none; }
  @top-center         { content: none; }
  @bottom-left-corner { content: none; }
  @bottom-left        { content: none; }
  @bottom-right       { content: none; }
  @bottom-right-corner{ content: none; }
  @bottom-center      { content: counter(page, lower-alpha); font-size: 11pt; }
```
When transformed, the page layout is spread on two columns.

**Large Tables Page**

The big tables are placed on a rotated page, with orientation landscape:
Blank Page

The following example clears the header for the blank pages that may be created by a page-break-before, page-break-after, or by using double side pagination (on page 1904):

```css
@page :blank{
  @top-left {
    content: none;
  }
  @top-right {
    content: none;
  }
}
```

Page Size

This is where you can find information on how the page sizes are defined.

Page Size - Built-in CSS rules

The [PLUGIN_DIR]/css/print/p-page-size.css file contains the default page rules. It uses the US-LETTER size (8.5 X 11 inches). The content of this file is:

```css
@page {
  padding-top:0.2em;
  padding-bottom:0.2em;
  size: us-letter;
  margin: 1in;
}
```
How to Change the Page Size

Suppose you want to publish using the standard A4 page size, with a margin of 2cm.

In your customization CSS (on page 1814), use:

```css
@page {
    size: A4;
    margin: 2cm;
}
```

If you need different margins depending on the page side:

```css
@page {
    size: A4;
    margin: 2cm;
}
@page :left{
    margin-right:4cm;
}
@page :right{
    margin-left:4cm;
}
```

This would only increase the gutter margins or the inside margins needed for binding of the final book. The other margins would remain 2cm.

How to Change the Page Orientation

Suppose you want to publish on a landscape page orientation. The default is portrait, so you need to change it by using the size property. This will contain both the physical measurements and the orientation. In your customization CSS (on page 1814), use:

```css
@page {
    size: us-letter landscape;
}
```

How to Change the Page Settings for a Specific Element

Suppose your publication mainly uses a portrait page orientation, but there are some topics that have wide images. To avoid having the images bleed outside of the page, you could use a wider page setting (landscape).
1. Mark the topic with an `@outputclass` attribute and give it a distinct value (for example, `wide`), you can set the attribute on the root element of the topic or on the `<topicref>` element from the map.

   **Note:**
   The `@outputclass` values from the `<topicref>` automatically propagate to the root of the topic from the merged map (on page 1816).

2. In your customization CSS (on page 1814), match the output class and associate it with a named page. In the following example, the page has a landscape orientation and small margins. This technique works for any element (e.g. a table or list) not just for a topic.

   ```css
   @page wide-page {
       size: letter landscape;
       margin: 0.5in;
   }

   *[outputclass = 'wide'] {
       page: wide-page !important;
   }
   
   **Note:**
   The `!important` rule is necessary to override the default page settings.

**Page Headers and Footers**

The page headers and footers use the string sets defined for publication, chapter, and section titles. These string-sets are defined in the numbering CSS (on page 1879):

- **parttitle**
  Set to the title of the current part (only for DITA bookmaps that use parts).

- **chaptertitle**
  Set to the title of the current chapter (Shallow and Deep numbering).

- **sectiontitle**
  Set to the title of each section (Deep numbering only).

To see where the default page rules are defined, see: Default Page Definitions (on page 1821).

Although you may define string sets in your customization CSS, you need to take into account the fact that the string-set CSS property is not additive, and matching the same elements will end up breaking the current definitions. A very common use-case is to match the title element that is also used in the default CSS. The best approach, in this case, is to take a look at the rules from the numbering CSS (on page 1879), copy the ones dealing with string sets to your customization, then alter the property definition by adding your definition to the existing ones (and not removing the existing ones).
Page Headers and Footers - Built-in CSS

The headers and footers are part of the page definitions. To see how the default page layouts are defined, see: Default Page Definitions (on page 1821).

How to Position Text in the Headers and Footers

By default, the name of the publication and chapter titles are placed in the top-left or top-right page margin boxes:

```
@page :left {
    @top-left {
        content: string(maptitle) string(parttitle) string(chaptertitle) string(sectiontitle) "|
        counter(page);
    }
}

@page :right{
    @top-right {
        content: string(maptitle) string(parttitle) string(chaptertitle) string(sectiontitle) "|
        counter(page);
    }
}
```

If you want to change this, you should use the content CSS properties of other page margin boxes, and inhibit the ones in the above content. For example, to set the chapter title in the page top left corner, you can use:

```
@page :left {
    @top-left {
        content:none;
    }
    @top-left-corner {
        content: string(maptitle) string(parttitle) string(chaptertitle) string(sectiontitle) "|
        counter(page);
        white-space: nowrap;
        text-align: left;
    }
}

@page :right{
    @top-right {
```
Note:
The corner page margin boxes are fixed and limited as the available space. Above, the text-align and white-space properties are used to make the text bleed out of these boxes towards the center of the page. If you plan to add an image or artwork background, you should consider using the technique described in: How to Decorate the Header by Using a Background Image on the Entire Page (on page 1839).

How to Display Chapter's Headers on First Page

By default, the header is not displayed on the first page of each chapter:

```xml
/* No headers on the chapter first page. */
@page chapter:first:left{
    @top-left {
        content: none;
    }
}

@page chapter:first:right{
    @top-right {
        content: none;
    }
}
```

If you want to display them on the first page, you just need to override the above default rules with the following default content:

```xml
@page chapter:first:left{
    @top-left {
        content: string(maptitle) string(parttitle) string(chaptertitle) string(sectiontitle) " | " counter(page);
        white-space: nowrap;
        text-align:right;
    }
}

@page chapter:first:right{
    @top-right {
    }
}
```
How to Change the Size of Headers and Footers

This is directly related to the page margins and size.

The headers and footers are placed in the so-called page margin boxes, a series of rectangular areas residing in the page margins.

To affect the margins of all page definitions, you may use the following rule:

```css
@page {
  margin-top: 3cm !important;
  margin-bottom: 3cm !important;
  margin-left: 2cm !important;
  margin-right: 2cm !important;
}
```

If you want to affect only a specific page, like the first page from chapters for instance, you must use more specific page selectors. See the Default Page Definitions (on page 1821) for details.

Note that the page margin boxes fill the entire page margin. This means the `margin-top`, for example, dictates the height of the `@top-left-corner`, `@top-left`, `@top-center`, `@top-right`, `@top-right-corner` margin boxes. These cannot have margins on themselves, so to change the position of the content inside them, you must use `padding` properties:

```css
@page {
  @top-left {
    content: "...
    padding: 1cm;
  }
  ...
}
```

How to Change the Font of the Headers and Footers

To change the font for all the headers and footers, in your customization CSS (on page 1814), add a CSS rule similar to this:

```css
@page {
  font-size: 12pt;
  font-family: "Arial";
}
```
**Important:**
These settings apply to all page margin boxes, but not to the text inside the page.

If you want to change the settings only for a specific page type (for example, the table of contents), use the name of the page:

```css
@page table-of-contents {
    font-size: 12pt;
    font-family: "Arial";
}
```

**Related Information:**
How to Change the Header of the Table of Contents *(on page 1895)*

### How to Change Header Text for Each Topic

It is possible to dynamically change the header depending on the content in a topic. The following example assumes that the data to be presented in the header is located in the metadata section of each topic. One way is to specify it in the DITA map by using the `<topicmeta>` element for the `<topicref>` topic reference:

```xml
...  
<topicref href="topics/installing.dita">
    <topicmeta>
        <data name="header-data" value="ID778-3211"/>
    </topicmeta>
...  
```

In the above example, there is a set of key value pairs with the name `header-data`. This information is automatically copied into the content in the *merged map file (on page 1816)*, like this:

```xml
<topic ... >
    <title class="- topic/title ">Installing</title>
    <shortdesc class="- topic/shortdesc ">You install components to make them available for your solution.</shortdesc>
    <prolog class="- topic/prolog ">
        ...
    </prolog>
    ...
    <data class="- topic/data " name="header-data" value="ID778-3211"/>
...  
```

This information can be extracted from the CSS:

```css
/* Define the string set variable that contains the text extracted from the data element */
*[class = "topic/topic"] *[class = "topic/data"] [name="header-data"] {
    string-set: hdrstr attr(value);
}
```
/* Using the value='none' stops applying the image. */
*[class = "topic/topic"] *[class = "topic/data"]*[name="header-data"][value="none"] {
  string-set: hdrstr "";
}

/* Use the string set variable in one of the page margin boxes. */
@page chapter {
  @top-left-corner {
    content: string(hdrstr);
  }
}

Notes:
The string set is applied to all pages that follow the data element, until another data element changes it:

...<topicref href="topics/installing.dita">
  <topicmeta>
    <data name="header-data" value="ID778-3211"/>
  </topicmeta>
</topicref>

  <!-- Uses the same value -->
</topicref>

  <!-- The string set is changed now -->
</topicref>

  <!-- The string set is void now -->
...</nary set is void now -->
How to Change Header Images for Each Chapter

It is possible to dynamically change an image in the header depending on the chapter. For this, you need to define an image reference in the metadata section of each chapter. One way is to specify it in the DITA map by using the `<topicmeta>` element for the `<chapter>` topic reference:

```xml
...  
<chapter href="topics/installing.dita">  
  <topicmeta>  
    <data name="header-image" value="img/installing.png"/>  
  </topicmeta>  
...  
```

In the above example, there is set of key value pairs with the name `header-image`. The `img/installing.png` is an image reference relative to the DITA map URI. This information is automatically copied into the content in the merged map file (on page 1816), like this:

```xml
<footer is-chapter="true" ... >  
  <title class="- topic/title ">Installing</title>  
  <shortdesc class="- topic/shortdesc ">You install components to make them available for your solution.</shortdesc>  
  <prolog class="- topic/prolog ">...  
  <data class="- topic/data " name="header-image" value="img/installing.png"/>  
  ...  
```

This information can be picked up from CSS:

```css
/* Define the string set variable that contains an URL */
* [class =~ "topic/topic"] * [class =~ "topic/data"] [name="header-image"] {  
  string-set: imgst oxy_url(oxy_xpath('/@xtrf'), attr(value));  
}

/* Using the value='none' stops applying the image. */
* [class =~ "topic/topic"] * [class =~ "topic/data"] [name="header-image"] [value="none"] {  
  string-set: imgst "";  
}

/* Use the string set variable in one of the page margin boxes. */
@page chapter {  
  @top-left-corner {  
    content: string(imgst);  
    font-size:0; /* remove the font ascent and descent */  
  }  
}
```
Details: The @value attribute is used to build a URL relative to the URI of the DITA map. To determine the base URI of the DITA map, the @xtrf attribute was used from the root element of the merged map document, extracted using theoxy_xpath function.

Notes:

- The image is always aligned vertically to the middle of available space from the page margin box.
- Make sure you use an image of the correct size. For example, if you want to place the image in the top-left corner of the page, assuming the top and left page margins are 1 in, then make sure the image is a square having a size of 1 in.
- The image is applied to all pages that follow the data element, until another data element changes it:

```
...<chapter href="topics/installing.dita">
  <topicmeta>
    <data name="header-image" value="img/installing.png"/>
  </topicmeta>
</chapter>
<chapter href="..."><!-- Uses the same installing.png image -->
<chapter href="..."><!-- Uses the same installing.png image -->
<chapter href="..."><!-- Uses the same installing.png image -->
<chapter href="topics/change.dita">
  <topicmeta>
    <data name="header-image" value="img/change.png"/>
  </topicmeta>
</chapter>
<chapter href="..."><!-- Uses the same change.png image -->
<chapter href="..."><!-- Uses the same change.png image -->
<chapter href="..."><!-- Uses the same change.png image -->
To clear the image, use the none value:

```
...<data name="header-image" value="none"/>
...```

How to Add a Multi-line Copyright Notice to the Footer

Suppose you want to add a footer with the following two lines of text at the end of each page that is shown on the right side:
For this, you need to specify a rule that matches all the right pages and adds that content in the `bottom-center`.

In your customization CSS (on page 1814), add the following CSS rule:

```css
@page :right {
  @bottom-center {
    content: "© 2017 - My Company Ltd \A All rights reserved";
    font-size: 0.5em;
    color: silver;
  }
}
```

**Note:**

Other page rules (such as the `table-of-contents`) override the contents of the `@bottom-center` because they are more specific. If you need to also print the copyright in the TOC pages, then use this as the selector:

```css
@page :right, table-of-contents:right {
  ... 
}
```

**Note:**

To use new lines (\n characters) in your headers or footers, use the \A notation, as in the example above.

### How to Add a Group of Topics to the Footer

To create a footer that contains the content of several topic files, but only on the last page, there are two possible approaches:

**Method 1: Using the position:fixed CSS Property**

1. Group all the footer topics under a single parent topic, under the last topic from your DITA map. For example, you can have the following map structure:

```xml
... 
End topic 
  Footer container topic 
    Footer content topic 1 
    Footer content topic 2 
```

2. Add an `@outputclass=footer` on the `<topic>` root element of the footer container topic, or on its `<topicref>` in the map.
3. Use the CSS `position: fixed` property to position this topic to the bottom of the page:

```css
*[outputclass ~= "footer"] {  
    position: fixed;  
    bottom: 0.5in;  
    left: 0.5in;  
    width:5in;  
    height:200pt;  
}
```

Note:
Make sure the width and height are enough for the content of the footer to fit. Be careful because the content might bleed out of the page. Use bottom and left values to position the block in the page.

**Method 2: Using the float:footnote CSS Property**

The second approach would be to declare the footer block as a footnote. Assuming the same DITA Map structure as above, you can use the following CSS fragment:

```css
*[outputclass ~= "footer"] {  
    float:footnote;  
}
*[outputclass ~= "footer"]:footnote-call{  
    color:transparent;  
    font-size:0;  
}
*[outputclass ~= "footer"]:footnote-marker{  
    color:transparent;  
    font-size:0;  
}
```

Note:
Use transparent colors and/or zero size font to avoid the display of the footnote counters.

**How to Add a Background Image to the Header**

A common use-case is to add a background image to one of the page corners.

```css
@page :left {  
    @bottom-left-corner{  
```
Important:

Always specify a `content` property. If not, the page margin box will not be generated.

Another use-case is to use the @top-left or @top-right page margin boxes. These boxes have an automatic layout and they can be very small if they have no content. If there is no text to be placed over the image, use a series of non-breaking spaces (\A0) to increase the box width as in the following example (alternatively, you can use the technique described in How to Decorate the Header by Using a Background Image on the Entire Page (on page 1839)):

```css
@page :left {
  @top-left{
    content: '\A0\A0\A0\A0\A0\A0\A0\A0\A0\A0\A0\A0';
    background-image: url('https://www.oxygenxml.com/resellers/resources/OxygenXMLEditor_icon.svg');
    background-repeat:no-repeat;
    background-position:50% 50%;
  }
}
```

Note:

You can use raster image formats (such as PNG or JPEG), but it is best to use vector images (such as SVG or PDF). They scale very well and produce better results when printed. In addition, the text from these images is searchable and can be selected (if the glyphs have not been converted to shapes) in the PDF viewer.

Related Information:

Images and Figures (on page 1954)

How to Add a Background Image for the Cover (on page 1854)

How to Add a Link in Headers and Footers (on page 1839)
How to Add a Link in Headers and Footers

Method 1: Using an SVG Link Attribute

It is possible to add a link inside the document header (or footer) by using the `<a>` element inside an SVG document. For example, suppose you have the following SVG document named `custom.svg`:

```xml
<svg viewBox="0 0 100 40" xmlns="http://www.w3.org/2000/svg">
    <text x="10" y="25">PDF Chemistry</text>
  </a>
</svg>
```

This creates an SVG link with *PDF Chemistry* displayed as its text (the content of the `<text>` element).

**Note:**
If you just want to add a link without text, you can define a rectangle that contains the link instead of text.

To display the link, you just need to set your SVG file as the content of one of the page margin boxes:

```css
@page {
  @top-left {
    content: url("custom.svg");
  }
}
```

Method 2: Using the CSS `-oxy-link` Property

It is also possible to add a link inside the document header (or footer) by using the `-oxy-link` property on the `@page` margin box declaration. The entire page margin box will behave as a link and will be clickable.

```css
@page {
  @top-left {
    content: "Link";
    -oxy-link: "https://www.oxygenxml.com/";
    color: blue;
  }
}
```

How to Decorate the Header by Using a Background Image on the Entire Page

If you want to precisely position artwork and the page margin boxes are not sufficient, it is possible to use a background image for the entire page.
This technique consists of creating an image (SVG is the best since it is a vector image) as wide as the page that would contain the logo and placing other decorations at the desired locations. This offers the best results and the position of the artwork does not depend on the page margin contents.

Example:

```css
@page :left, chapter: left, chapter: first: left {
    background-image: url('img/page_background_image_with_logos_and_artwork_for_left_page.svg');
    background-repeat: no-repeat;
    background-position: 50\% 50\%;
    background-size: 8.5in 11.5in; /* Optional: Adapt to your page size. */
}
```

For a list of all the possible page names, see: Default Page Definitions (on page 1821).

Related Information:

How to Add a Background Image for the Cover (on page 1854)

How to Style a Part of the Text from the Header

If you need to style a fragment of text (for example, a company slogan) with certain colors or font styles, you have several options:

- Use an SVG image as the background for a page margin box or for the entire page. See: How to Add a Background Image to the Header (on page 1837).
- Use the `oxy_label` constructor. This is a function that creates a text label with a set of styles.

```css
@page {
    @top-right {
        content: oxy_label(text, "My Company", styles, "color:red; font-size: larger;")
       oxy_label(text, "Product", styles, "color:blue; text-decoration:underline;")
    }
}
```

You can combine the `oxy_label` with `oxy_xpath`, to extract and style a piece of text from the document:

```css
content: oxy_label(text, oxy_xpath("/some(xpath)"), styles, "color:blue; ");
```

Note:

These functions work only with the Chemistry CSS processor.
Note:
You cannot use `string()` inside an `oxy_label()`. As a workaround, to apply styling on the dynamic text retrieved by a `string()` function you can define some overall styles for the entire page margin box and then use the `oxy_label` to style differently the static text.

```css
@page {
  @top-right {
    color: red;
    content: oxy_label(text, "My Company", styles, "color:black")
    , ,
    string(chaptentitle); /* This inherits the styling from @top-right*/
  }
}
```

- Use two adjacent page margin boxes, and style them differently:

```css
@page {
  @top-center {
    content: "First part";
    color: red;
    text-align:right;
  }
  @top-left {
    content: "- Second part";
    color: blue;
    text-align:left;
  }
}
```

How to Simplify the Header (Keep Only the Chapter Title)

The headers display information such as `map title`, `part title`, `chapter title`, and `section title`, ending in the page number.

```css
content: string(maptitle) string(parttitle) string(chaptentitle) string(sectiontitle) " | " counter(page);
```

This might be too much if you have long titles. The solution is to override the default header content.

In your customization CSS (on page 1814), add the following CSS rule:

```css
@page :left {
  @top-left {
    content: string(chaptentitle) " | " counter(page);
  }
```
Important:
Some of the CSS default page rules are more important. If you see that the content does not change:

- Try to also specify the name of the page, to increase the specificity of the rules:

```css
@page :left, table-of-contents:left, chapter:left{
...
}
@page :right, table-of-contents:right, chapter:right{
...
}
```

- Add an `!important` classifier just before the semi-colon.

```css
@top-right {
    content: string(chaptertitle) " | " counter(page) !important;
}
```

How to Change the Header Separators (Between Sections and Chapter Titles)

There are some strings defined for part, chapter, and sections. Each of these strings start with the " | " character as a separator. For example, in the header of a page, you may find a sequence of strings:

```
My Publication | Introduction | Getting Started
```

1. "My Publication" is the value of the `maptitle` string.
2. "Introduction" is the value of the `chaptertitle` string.
3. "Getting Started" is the value of the `sectiontitle` string.

There might be cases where you want to change this separator. You will need to recompose the header content using the above string sets. Suppose you want to use " - " as a separator. In your customization CSS (on page 1814), add the following CSS rule:

```css
*[class =~ "topic/topic"] [is-part] > *[class =~ "topic/title"] {
    string-set: parttitle " - " counter(part, upper-roman) " - " content(), chaptertitle ""; /*
    Avoid propagating a past chapter title on a new part */
}
*[class =~ "topic/topic"] [is-chapter]:not([is-part]) > *[class =~ "topic/title"] {
```
If you enabled the deep numbering for chapters and subsections *(on page 1883)*, then use:

```xml
/*
 * Alter the string sets that are shown in the header of the page.
 */
*{[class ~= "map/map"]}*{[p|numbering~='deep']}{*[class ~= "topic/topic"]}*{[is-part]} > *{[class ~= "topic/title"]}{
    string-set: parttitle " - " counter(part, upper-roman) " - " content(), chaptertitle "" ; /*
Avoid propagating a past chapter title on a new part */
}
*{[class ~= "map/map"]}*{[p|numbering~='deep']}{*[class ~= "topic/topic"]}*{[is-chapter]:not([is-part])} > *{[class ~= "topic/title"]}{
    string-set: chaptertitle " - " counter(chapter) " - " content(), sectiontitle "" ;
}
*{[class ~= "map/map"]}*{[p|numbering~='deep']}{*[class ~= "topic/topic"]}*{[is-chapter]:not([is-part])} > *{[class ~= "topic/topic"]}{
    string-set: sectiontitle " - " counter(chapter) "." counter(section1) " - " content();
}
```

**How to Change the Header Styling Depending on Page Side**

To modify the styling of the default page headers, add the following CSS rule in your customization CSS *(on page 1814)*:

```css
@page :left {
    @top-left {
        color:navy;
        font-style:italic;
    }
    @top-right {
        color:red;
    }
}
```

If you intend to modify just the headers of the table of contents, use the table-of-contents page rule selector:

```css
@page table-of-contents: :left {
    @top-left {
        color:navy;
        font-style:italic;
    }
    @top-right {
```
How to Use XPath Computed Data or Images in the Header or Footer

A very simple approach is to use the oxy_xpath directly in the content property:

```
@page front-page {
  @top-center {
    content: "Created: " oxy_xpath('/'*[contains(@class, " topic/created "])[1]');
  }
}
```

**Example 1: Compute the Number of Words**

The following example computes the number of words from the publication. It counts all the words, including the ones from the TOC, but does not take the static labels into account:

```
@page front-page {
  @bottom-center {
    content: "Number of words: 
    oxy_xpath("string-length(normalize-space(/)) - \n    string-length(translate(normalize-space(/),' ','')) +1");
  }
}
```

**Note:**

The XPath expression from the page rules is evaluated in the context of the document root element, so you will need to use absolute expressions starting with / or //. This is different from the case when the oxy_xpath is used in CSS rules that match an element. In this case, the XPath expressions are evaluated in the context of the matched element and you can use relative paths.

**Tip:**

XPath 2.0 is supported (not schema aware).

**Example 2: Retrieve Image from a Document and Insert it in the Header**

Another example is to use an image from the document in the publication header:

```
<bookmeta>
  <metadata>
    ...
  </metadata>
  <data name="cover">
    <image href="product-cover.png" outputclass="cover-image"/>
  </data>
</bookmeta>
```
If the URL returned by `oxy_xpath` is not absolute, it is considered to be relative to the CSS file. To obtain an absolute URL from one relative to the XML document, you can use in the XPath expression functions like `resolve-uri` and `document-uri`:

```xml
@page {
  @top-center {
    content: url(oxy_xpath("resolve-uri(/*[contains(@outputclass, 'cover-image')]/@href),
    document-uri(/))");
  }
}
```

Example 3: Insert the Current Date in the Footer

Another example is to use the `oxy_xpath` function to compute the current date and insert it in the publication footer:

```xml
@page {
  @bottom-left {
    content: oxy_xpath('current-date()');
  }
}
```

Example 4: Picking up Metadata from the Original Map

Another example is to use the `oxy_xpath` function to extract the title, or any other element text value from the original processed DITA map file. For this, you can use the `@xtrf` attribute that is set on the root element of the merged map. This attribute contains the URL of the input map.

```xml
:root{
  string-set: maptitle oxy_xpath("document(@xtrf)/*[contains(@class, " map/map ")/*[contains(@class, " topic/title ")]/text()");
}
```
How to Add a Line Under the Header

There are two ways to add a horizontal line under the header.

**Method 1: Add a Border in the Page Margin Boxes**

To add a horizontal line that would stretch across the width of the page, add a bottom border to each of the 5 margin boxes in the top side of the page (`top-left-corner, top-left, top-center, top-right, top-right-corner`).

If you consider that the space between the header and the bottom border is too large, you could also change the alignment by adding a `vertical-align: bottom;` declaration in the page margin boxes.

For example, if you need to set some text as a header in the top-left margin box and insert a horizontal line under it, the customization CSS would look something like this:

```css
@page chapter, chapter:first:left:right, front-page{
  padding-top: 1em;

  @top-left {
    content: "Custom header";
    color: gray;
    border-bottom: 1px solid black;
    vertical-align: bottom;
  }

  @top-center{
    content: " ";
    border-bottom: 1px solid black;
    vertical-align: bottom;
  }

  @top-right{
    content: " ";
    border-bottom: 1px solid black;
    vertical-align: bottom;
  }

  @top-right-corner{
```

Related Information:
- Oxygen PDF Chemistry User Guide: Headers and Footers
  http://zvon.org/xxl/XPathTutorial/General/examples.html
- Oxygen User Guide: `oxy_xpath()` Function
Method 2: Use a Background Image

An alternative method is to add a horizontal line/border under an existing header (or in any other part of the page) using an SVG image, as described in How to Add a Background Image to the Header (on page 1837).

How to Change the Headings Using a Parameter

Suppose you need to change the headings of your publication by specifying a static text in a parameter.

First, establish a name for your parameter (it must start with the args.css.param. prefix). For example, you could name it args.css.param.heading.text. It will have the text value that you will pass when starting the transformation. This parameter does not have to be registered anywhere as it will be automatically recognized and passed as an XML attribute on the root of the merged file, as specified in Styling Through Custom Parameters (on page 1975).

Next, alter your customization CSS to make use of the parameter value. In the example below, the text is placed in the central part of the header:

```css
@page front-page, table-of-contents, chapter {
  @top-center{
      content: oxy_xpath("/*/@heading.text");
  }
}
```

Note:
You can use any XPath 2.0 here. It will be executed in the context of the merged map document, so you can collect data from it. You can use if/then/else expressions if your parameter is a switch.
The text does not affect the first pages from the page sequences because the built-in CSS page rules (on page 1821) clear the content from the headers. If you need the text content on all pages, you might consider adding an !important keyword after the content property value, or increase the specificity of the page selectors, like this:

```css
  @top-center{
    ...
  }
}
```

Another use case is to alter the string-sets that are used in the headers (not the headers directly), as it is explained here: How to Use XPath Computed Data or Images in the Header or Footer (on page 1844). You can use this technique to alter the chapter titles as in the following example:

```css
*[class =~ "map/map"]*[numbering~='deep']
 *[class =~ "topic/topic"]*[is-chapter]:not([is-part]) > *[class =~ "topic/title"] {
  string-set:
    chartertitle " | " counters(chapter-and-sections, ":") " - "
  oxy_xpath("/*/@heading.text") content(),
  sectiontitle "";
}
```

>Note:
This is a rule copied from p-numbering-deep.css and it may change if future versions.

**How to Change the Headings depending on the Language**

It is possible to customize the text displayed in the headings depending on the language of the publication.

In this case, you can simply use of the @lang attribute in your customization CSS. In the following example, the page counter displayed in the bottom part of the page is preceded by the word "Page", according to the selected language:

```css
@page chapter {
  @bottom-center {
    content: oxy_xpath("if (@lang='es') then 'Página' \\
      else if (@lang='it') then 'Pagina' \\
      else 'Page'\") = " counter(page);
  }
}
```
Page Breaks

The page breaks can be controlled in multiple ways:

1. By creating an `@page` and assigning it to an element will create a page break between this element and the sibling elements that have a different page.
3. In your DITA topic, set the `@outputclass` attribute on the topic root (or any element) to contain one of the `page-break-before`, `page-break-after`, or `page-break-avoid` values. If you want to control the page breaking from the DITA map, use the `@outputclass` attribute on the `<topicref>`, with any of the values mentioned above.

Related Information:
- Double Side Pagination (on page 1904)
- Oxygen PDF Chemistry: Controlling Page Breaks

Page Breaks - Built-in CSS

Page break properties are used in: `[PLUGIN_DIR]css/print/p-page-breaks.css`.

How to Avoid Page Breaks in Lists and Tables

To avoid splitting elements over two pages, you can use the `page-break-inside` CSS property. For example, if you want to impose this on tables and lists, then add the following rules to your customization CSS (on page 1814):

```css
*[class =~ "topic/table"] {  
  page-break-inside:avoid;
}
*[class =~ "topic/ol"] {  
  page-break-inside:avoid;
}
*[class =~ "topic/ul"] {  
  page-break-inside:avoid;
}
```
How to Force a Page Break Before or After a Topic or Another Element

If you want to force a page break **before all** the second-level topics (for example, sections in chapters that are usually kept flowing one after another without page breaks), add the following in your customization CSS (on page 1814):

```css
*{class ~="map/map"} > *{class ~="topic/topic"} > *{class ~="topic/topic"} {  
  page-break-before:always;
}
```

If you need to break at third or fourth level topics, add more `.. > *{class ~="topic/topic"}` selectors to the expression.

If you want to force a page break **for a specific topic**, mark the topic (or any other element you need to control page breaking for) with an `@outputclass` attribute set to one of these values:

- `page-break-before`: Use this for a page break before the marked element.
- `page-break-after`: Use this for a page break after the marked element.
- `page-break-avoid`: Use this to avoid page breaks inside the marked element.

For example, to force a page break before a certain topic, use:

```xml
<topic outputclass="page-break-before" ... >
```

Note:

You can set the output class on the `<topicref>` element from the DITA map instead of the `<topic>` element. In this way you can reuse the topic in another context where the page breaking is not necessary.
You can also control page breaking for lists, paragraphs, or any other block type elements. The following example avoids page breaks inside an ordered list:

```xml
<ol outputclass="page-break-avoid" ...>
```

### How to Add a Blank Page After a Topic

If you want to add a new blank page after a topic, add the following rules to your customization CSS (on page 1814).

**Style the separating blank page:**

```css
@page topic-separating-page{
  @top-left {
    content: "";
  }
  @top-right {
    content: "";
  }
  @top-center {
    content: "This page is blank";
  }
}
```

Associate this page to the :after pseudo-element of the topic:

```css
*[class~="topic/topic"] [outputclass~="add-separator-page"]:after {
  content: " ";
  display: block;
  page: topic-separating-page;
}
```

In the XML content, on the `<topic>` element, set the `@outputclass` to the `add-separator-page` value.

```xml
<topic outputclass="add-separator-page"> ... </topic>
```

The :after pseudo-element will be created next to the topic content and will be placed on the topic-separating-page.

Use the page margin box selectors to override the default content from the headers/footers.

---

**Note:**

You can set the output class on the `<topicref>` element from the DITA map instead of the `<topic>` element. This allows you to reuse the topic in another context where the page breaking is not necessary.
How to Enforce a Number of Lines from Paragraphs that Continue in Next Page

In typography, an orphan is the first line of a paragraph that appears alone at the bottom of a page (the paragraph continues on a subsequent page), while a widow is the last line of a paragraph that appears alone at the top of a page. The default is 2 for each of them. You can control this number by adding the following to your customization CSS (on page 1814):

```
:root {
  widows: 4;
  orphans: 4;
}
```

**Note:**
As a difference from the W3C standard, the widows and orphans CSS properties are applied to lists as well (the default is 2). This means that a list that spans consecutive pages will have either zero or at least 2 lines on each of the pages.

How to Avoid Page Breaks Between Top-Level Topics (Chapters)

If you plan to publish a simple map with just one level of topics (such as a list of topics), then the automated page breaks between these topics might not be desired.

In this case, you can use the following CSS snippet to disable the page breaks between chapters:

```
*[class ~="topic/topic"] [is-chapter] {
  -oxy-page-group: auto;
}
```

Related Information:
Oxygen PDF Chemistry User Guide: Chapter Page Placement and Styling

Cover (Title) Page

Customizing the cover page is one of the most requested customization requests.

Cover Page - XML Fragment

The merged map file (on page 1816) contains the <oxy:front-page> element, as a child of the root element. This contains the metadata and an <oxy:front-page-title> element with the title structure.

```
<bookmap xmlns:ditaarch="http://dita.oasis-open.org/architecture/2005/" ...>
    <bookmeta xmlns:dita-ot="http://dita-ot.sourceforge.net/ns/201007/dita-ot" ...
```
For the **DITA Map PDF - based on HTML5 & CSS** transformation type, the merged map is further processed resulting in a collection of HTML5 `<div>` elements. These elements preserve the original DITA `@class` attribute values and add a new value derived from the DITA element name.

```xml
<oxy:front-page-title>
  <booktitle xmlns:dita-ot="http://dita-ot.sourceforge.net/ns/201007/dita-ot" class="- topic/title bookmap/booktitle ">
    <booklibrary class="- topic/ph bookmap/booklibrary ">Retro Tools</booklibrary>
    <mainbooktitle class="- topic/ph bookmap/mainbooktitle ">Tasks</mainbooktitle>
    <booktitlealt class="- topic/ph bookmap/booktitlealt ">Product tasks</booktitlealt>
  </booktitle>
</oxy:front-page-title>
</oxy:front-page>
```

Cover Page - Built-in CSS rules

The element with the class `fronpage/frontpage` is associated with a page named `front-page` with no headers or footers. The front page title is styled with a bigger font. The built-in CSS rules are in `[PLUGIN_DIR]/css/print/p-front-page.css`.

```css
@media print {

  *[class="front-page/front-page"] {
    page: front-page;
  }

  /* Prevents the front-page title margin collapsing */
  *[class="front-page/front-page"]::before(1000) {
    display:block;
    content:\\A;
```
How to Add a Background Image for the Cover

The simplest way is to create an SVG image as large as the entire physical page and set it as the background for the front-page. This makes it easy to accomplish a good positioning of the graphical elements or artwork. In the foreground, you can place text fragments using a series of :after pseudo-elements bound to the front page title.

To set the size to an SVG image, you should specify the @width and @height attributes on the <svg> root element using specified unit values (in, cm, etc.) This should be enough only if all the coordinates from your drawing have unit identifiers.

If you are using unit-less coordinates in your drawing like the following:

```xml
<polygon points="17.78 826.21 577.51 ....
```
Next, make sure you also specify the `@viewBox` attribute on the `<svg>` root element that defines the abstract rectangle that contains the drawing:

```xml
<svg xmlns="http://www.w3.org/2000/svg" width="8.5in" height="11in" viewBox="0 0 600 850">
</svg>
```

The following SVG document has the `@width`, `@height`, and `@viewBox` attributes. The width and height have physical units (in inches), while the view box and rectangle coordinates are unit-less.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.1//EN"
  "http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd">
<svg xmlns="http://www.w3.org/2000/svg" width="8.5in" height="11in" viewBox="0 0 110 110">
  <desc>A gradient as big as a page.</desc>
  <defs>
    <linearGradient id="lc"
      x1="0%" y1="0%"
      x2="0%" y2="100%"
      spreadMethod="pad">
      <stop offset="0%" stop-color="#00DD00" stop-opacity="1"/>
      <stop offset="100%" stop-color="#00AA00" stop-opacity="1"/>
    </linearGradient>
  </defs>
  <rect x="5" y="5" width="100" height="100" rx="10" ry="10"
    style="fill:url(#lc); stroke: #005000; stroke-width: 3;"/>
  <text x="33%" y="50%" color="#FFFFAA">Sample</text>
</svg>
```

This example shows a gradient. It is the size of a US-LETTER page and can be used in a publication using this page size.

**Note:**

You can use raster image formats (such as PNG or JPEG), but it is best to use vector images (such as SVG or PDF). They scale very well and produce better results when printed. In addition, the text from these images is searchable and can be selected (if the glyphs have not been converted to shapes) in the PDF viewer.

In your customization CSS (on page 1814), add the following:

```css
@page front-page {
  background-image: url("us-letter.svg");
  background-position: center;
  background-repeat: no-repeat;
```
For smaller artworks, you can use `background-position` with percentage values to position and center the artwork (for example, a company logo):

```css
@page front-page {
  background-image: url("company-logo.svg");
  background-position: 50% 5% /* The first is the alignment on the X axis, the second on the Y axis. */
  background-repeat: no-repeat;
}
```

**Note:**
The text from the SVG or PDF background images is searchable in the PDF reader.

### How to Use Different Background Cover Images Based on Bookmap or Map Info

It is common to use the same CSS file for customizing multiple publications, and you may need to set a different cover for each of them. The solution is to use an XPath expression to extract some information from the document, and based on that, select the SVG images.

```css
@page front-page {
  background-image: url(oxy_xpath("\n    if(/*[contains(@class, ' topic/prodname ')][1] = 'gardening') then 'bg-gardening.svg' else
    if(/*[contains(@class, ' topic/prodname ')][1] = 'soil') then 'bg-soil.svg'
    else 'bg-default.svg'\n  "));
  background-position: center;
}
```

The backslash (\) is used to continue the expression string on the subsequent lines (there should be no spaces after it). For more use cases solved using XPath, see: Metadata *(on page 1863).*

### Related Information:

- Oxygen PDF Chemistry: Graphics

### How to Change Styling of the Cover Page Title

Match the front page title element in your customization CSS *(on page 1814)* based on its class attribute:

```css
*{class ~= "front-page/front-page-title"} {
  margin-top: 1in;
  font-size: 3em;
}
```
Important:

Make sure the sum of the top and bottom margins and paddings for this element do not exceed the physical dimension of the page. If this happens, an extra blank page may appear before the cover page. Usually, it is enough to specify only the top margin.

How to Add Text to the Cover Page

If you need to add arbitrary text to the cover page, you can use the front page title element as an anchor and add as many blocks of text as you need after it, and style them differently.

In your customization CSS (on page 1814), add the following:

```
/*[class ~="front-page/front-page-title"]:after(1) {  
  display:block;
  content: "DRAFT VERSION";
  font-size: large;
  color: red;
  text-align:center;
}

/*[class ~="front-page/front-page-title"]:after(2) {  
  display:block;
  content: "DO NOT DISTRIBUTE WITHOUT PERMISSION";
  font-size: large;
  color: red;
  text-align:center;
  font-style: italic;
}
```

The result is:

Retro Tools
Product tasks
Tasks and what they can do

DRAFT VERSION
DO NOT DISTRIBUTE WITHOUT PERMISSION
To use content from the document, you can use the `oxy_xpath` function in the `content` property. For a more complex example, including the generation of a new page for the synthetic `after` elements, see: How to Show Metadata in the Cover Page (on page 1869).

Related Information:
How to Show Metadata in the Cover Page (on page 1869)

How to Place Cover on the Right or Left Side

In your customization CSS (on page 1814), add the following CSS rules:

```css
*[class ~="front-page/front-page"]{
    page-break-before: left;
}
```

Note:
This will create an empty page at the beginning of the publication, moving the cover content on the needed side.

For more information, see: Oxygen PDF Chemistry: Controlling Page Breaks.

Related Information:
Double Side Pagination (on page 1004)

How to Add a Second Cover Page and Back Cover Page

It is possible to add a second cover page after the front-page by defining another page-selector:

```css
@page second-cover {
    @top-left {content: none};
    @top-right {content: none};
    @bottom-center {content: none};

    background-image: url("second-cover.svg");
    background-position: center;
    background-repeat: no-repeat;
    background-size: 100% 100%;
}

*[class = 'front-page/front-page']:after{
    page: second-cover;
    page-break-after: always;
    display: block;
}
```
If you want to add a back cover page, you should use an :after pseudo element on the map itself:

```xml
*[class ~= "map/map"]:after
```

and bind it to another @page declaration:

```xml
@page back-cover {
  @top-left {content: none};
  @top-right {content: none};
  @bottom-center {content: none};

  background-image: url("back-cover.svg");
  background-position: center;
  background-repeat: no-repeat;
  background-size: 100% 100%;
}

*[class ~= "map/map"]:after {
  page: back-cover;
  content: "\2002";
}
```

**Note:**
For any background-image, it is recommended to use SVG instead of PNG (or JPG) because it scales it to the page size.

**Tip:**
To add multiple cover pages, use multiple-leveled pseudo selectors, such as :after(1), :after(2). Remember that the larger the value, the more distant the pseudo element is to the target element.

**How to Dynamically Add a Second Cover Page**

It is possible to dynamically set the path to the SVG image that will be displayed on the secondary cover page.

First, you need to declare a `<data>` element in the `bookmap`'s metadata that contains the URL to your cover image:

```xml
<bookmap>
  <booktitle>
    ...
  </booktitle>
  <bookmeta>
```

```xml
  <data/>
```

```xml
  ...
</bookmeta>
```
Note:
This can also be done on a normal DITA map by using the `<topicmeta>` after the map's `<title>`.

Next, you need to modify the page declaration inside your CSS stylesheet and replace the `background-image` property value with the result of the `oxy_xpath()` function:

```css
@page second-cover {
    ... 
    background-image: url(oxy_xpath("//*[contains(@class, 'bookmap/bookmeta')]//*[contains(@class, 'topic/data')][@name='second-cover-url']/@value");
    ...
}
```

Tip:
You can reuse the same stylesheet on multiple maps. You just need to change the data value for each of them.

How to Add a Specific Number of Empty Pages After the Cover Page

In your customization CSS (on page 1814), add the following CSS rules:

```css
@page my-blank-page {
    /* Hide the page numbers */
    @top-left {content: none};
    @top-right {content: none};
}

*[class ~='front-page/front-page']:after(1){
    page:my-blank-page;
    display:block;
    content: '\'2002';
    color:transparent;
    page-break-after:always;
}

*[class ~='front-page/front-page']:after(2){
```
Note:
The \2002 character is a space that is not shown on the pages, but gives a value for the content property.

Related Information:
How to Force an Odd or Even Number of Pages in a Chapter (on page 1906)

How to Add a Copyright Page after the Map Cover (Not for Bookmaps)

Regular DITA maps do not have the concept of a copyright notice. This is available only in the DITA bookmap structure.

If you are constrained to using a regular map and you need to add a copyright page between the front cover and the TOC, use the following technique:

In your customization CSS (on page 1814), declare a new page layout:

```css
@page copyright-notice-page {
  @top-left {
    content:none; /* Clear the headers for the copyright page */
  }
  @top-right {
    content:none;
  }
}
```

The element with the class `front-page/front-page` element contains the title of the publication and generates the cover page. A synthetic `:after` element is created that follows this element and it is placed on a different page.
*{class="front-page/front-page"}:after{
  display:block;

  page: copyright-notice-page; /* Moves the synthetic element on a new page. */

  margin-top:90%; /* use margins to position the text in the page */
  margin-left: 5em;
  margin-right: 5em;

  content: "Copyright 2018-2019 MyCorp Inc. \A All rights reserved";

  text-align:center; /* More styling */
  color:blue;
}

If you need to add more content as blocks, use the :after(2), :after(3) pseudo-elements:

*{class="front-page/front-page"}:after(2){
  display:block;

  page: copyright-notice-page; /* Continue on the same page as the first ':after'. */
  content: "Some more styled text";
  color:red;
}

If you want to extract information from the document, use the oxy_xpath() function. For example, if the copyright info is stored in the map like this:

```xml
<map ...>
  <topicmeta>
    <copyright>
      <copyryear year="2018"/>
      <copyrholder>MyCorp Inc.</copyrholder>
    </copyright>
  </topicmeta>
...
```

then use this:

*{class="front-page/front-page"}:after(3){
  display: block;
  page: copyright-notice-page;
  content:
  "Year: "
}
oxy_xpath('//*[contains(@class, " front-page/front-page ")]/*[contains(@class, " map/topicmeta ")]/*[contains(@class, " topic/copyright ")]/*[contains(@class, " topic/copyyear ")]/@year')
"\A Holder:"
oxy_xpath('//*[contains(@class, " front-page/front-page ")]/*[contains(@class, " map/topicmeta ")]/*[contains(@class, " topic/copyright ")]/*[contains(@class, " topic/copyrholder ")]/text()');
color: green;
}

Related Information:
How to Debug XPath Expressions (on page 1820)

How to Remove the Cover Page and TOC

If you need to hide or remove the cover page, the table of contents or other structures, match the elements with a "front-page/front-page" and "toc/toc" classes in your customization CSS (on page 1814):

```css
*[class =~ 'map/map'] > *[class =~ 'toc/toc'] {
    display:none !important;
}
*[class =~ 'map/map'] > *[class =~ 'front-page/front-page']{
    display:none !important;
}
*[class='topic/topic'][is-chapter] {
    -oxy-page-group : auto;
}
```

Metadata

DITA has a solid vocabulary for specifying metadata. There are `<prolog>` elements in the topics, and `<topicmeta>, <bookmeta>` elements in the bookmaps. They can be used to define authors, dates, audiences, organizations, etc. See: https://www.oxygenxml.com/dita/1.3/specs/archSpec/base/metadata-in-maps-and-topics.html

It is up to you to decide where this information should be presented, in the PDF content or in the PDF document properties.

Metadata - XML Fragment

In the merged map file (on page 1816), the metadata section is placed inside the `<oxy:front-page>` element. This is different from the original placement in the map or bookmark (after the title), but allows for the usage of information from it in the title page.
Bookmaps

This is an example of a section taken from a merged bookmap. It only contains some of the possible metadata elements. The `bookmeta` metadata section is inherited from `topicmeta`:

```xml
 xmlns:opentopic-index="http://www.idiominc.com/opentopic/index" cascade="merge"
   class="- map/map bookmap/bookmap "
 ditaarch:DITAArchVersion="1.3" >
 <bookmeta xmlns:dita-ot="http://dita-ot.sourceforge.net/ns/201007/dita-ot"
   class="- map/topicmeta bookmap/bookmeta ">
   <author class="- topic/author ">Howe Tuduit</author>
   <bookid class="- topic/data bookmap/bookid ">
     <isbn class="- topic/data bookmap/isbn ">071271271X</isbn>
     <booknumber class="- topic/data bookmap/booknumber ">SG99-9999-00</booknumber>
   </bookid>
   <maintainer class="- topic/data bookmap/maintainer ">
     <organization class="- topic/data bookmap/organization ">ACME Tools</organization>
   </maintainer>
 <bookrights class="- topic/data bookmap/bookrights ">
 ... 
 <bookowner class="- topic/data bookmap/bookowner ">
 <organization class="- topic/data bookmap/organization ">ACME Tools, Inc.</organization>
 </bookowner>
 </bookrights>
 </bookmeta>

 <oxy:front-page-title>
 ... 
 ... 
</oxy:front-page-title>

For the DITA Map PDF - based on HTML5 & CSS transformation type, the merged map is further processed resulting in a collection of HTML5 `<div>` elements. These elements preserve the original DITA `@class` attribute values and add a new value derived from the DITA element name.
Maps

The maps have a more simple structure, they use the `<topicmeta>` element for metadata sections. This is also a simplified example, as there may be many more elements in the metadata section:

```xml
     xmlns:opentopic-index="http://www.idiominc.com/opentopic/index"
     cascade="merge" class="- map/map "
     ditaarch:DITAArchVersion="1.3">
...
</map>


  <topicmeta class="- map/topicmeta ">
    <author class="- topic/author ">Dan C</author>
    <metadata class="- topic/metadata ">
      <prodinfo class="- topic/prodinfo ">
        <prodnam...
For the **DITA Map PDF - based on HTML5 & CSS** transformation type, the merged map is further processed resulting in a collection of HTML5 `<div>` elements. These elements preserve the original DITA `@class` attribute values and add a new value derived from the DITA element name.

```xml
    xmlns:openopic-index="http://www.idiominc.com/opentopic/index"
    cascade="merge" class="map/map"
    ditaarch:DITAArchVersion="1.3">
    ...
</div>
```

For the **DITA Map PDF - based on HTML5 & CSS** transformation type, the merged map is further processed resulting in a collection of HTML5 `<div>` elements. These elements preserve the original DITA `@class` attribute values and add a new value derived from the DITA element name.

```xml
<div class="front-page/front-page front-page">
    <div class="-map/topicmeta topicmeta">
        <div class="-topic/author author">Dan C</div>
    </div>
</div>
```

**Metadata - Built-in CSS rules**

The `[PLUGIN_DIR]/css/print/p-meta.css` file contains the rules that extract metadata.

**How to Create a Searchable PDF**

To make a PDF searchable, you need to add some `<keyword>` or `<indexterm>` elements inside bookmaps, maps, or topics. Most of the search engines will parse the resulting document and extract those keywords and create a search base.

**Note:**

Both `<keyword>` and `<indexterm>` elements can be combined inside the `<keywords>` element. They will be equally processed by the search engine.

In the generated PDF, keywords are displayed in the Document Properties.
### Bookmaps

If you want your keywords to appear inside a bookmap, you need to define them inside the `<bookmeta>` element:

```xml
<bookmap>
  ...
  <bookmeta>
    <keywords>
      <keyword>web server</keyword>
      <keyword>hard disk</keyword>
    </keywords>
  </bookmeta>
</bookmap>
```

### Maps

If you want your keywords to appear inside a map, you need to define them inside the `<topicmeta>` element:

```xml
<map>
  ...
  <topicmeta>
    <keywords>
      <keyword>flowers</keyword>
      <indexterm>care and preparation</indexterm>
      <keyword>seasons</keyword>
    </keywords>
  </topicmeta>
</map>
```

### Topics

If you want your keywords to appear inside one or more topics, you need to define them inside the `<prolog>` element:

```xml
<topic>
  ...
  <prolog>
    <metadata>
      <keywords>
        <indexterm>iris</indexterm>
      </keywords>
    </metadata>
  </prolog>
</topic>
```

---

**Warning:**

Keywords must be at map level or at topic level, you cannot combine them.
How to Add the Publication Audience to the Custom PDF Metadata

The audience element indicates the users the publication is addressing. This can be placed inside a `<topicmeta>` element in a `<map>` as in the following example:

```xml
<map>
    ...
    <topicmeta>
        ...
        <audience type="programmer" job="programming" experiencelevel="expert"/>
    </topicmeta>
</map>
```

To collect the `@type` attribute, add the following in your customization CSS (on page 1814):

```css
*[class =~ "map/map"] > *[class =~ "map/topicmeta"] > *[class =~ "topic/audience"] {
    -oxy-pdf-meta-custom: "Audience" attr(type);
}
```

**Notice:**
It is best to use the class selector (such as `*[class =~ "map/topicmeta"]`) instead of `topicmeta` to cover cases where the elements are specialized (for instance, in a bookmap the `bookmeta` is a `topicmeta`, so your selector will also function for bookmaps, not only simple maps).

**Note:**
The selector begins with `map >` to choose the `<topicmeta>` that is a direct child of the map, not other `<topicmeta>` elements from other `<topicref>` elements.

**Tip:**
You can define multiple key value pairs by separating them with commas:

```css
-oxy-pdf-meta-custom: "Audience" attr(type), "Job" attr(job)
```

The metadata is displayed in the **Custom** tab of the **Document Properties** dialog box from Acrobat Reader:
How to Show Metadata in the Cover Page

The following CSS extensions are used in the subsequent examples:

- **oxy_xpath** - Executes an XPath expression and returns string content. Use this whenever you need to extract data from an element other than the one matched by the CSS rule selector.

- **:after(N)** - Creates more than one after pseudo-element. The argument value represents how far the generated content is from the real content. For example, in the second code snippet in the next section (on page 1870), the content of the :after is closer to the title (upper) than the content of the :after(2).

**Note:**
The attr() CSS function can also be used but it is limited to extracting attribute values from the matched element.
Processing Metadata for Bookmaps

Suppose you need to present the Author and the ISBN (when it exists) just under the publication title and suppose your bookmap contains:

```xml
<bookmap id="taskbook">
  <booktitle>
    <booklibrary>Retro Tools</booklibrary>
    <mainbooktitle>Product tasks</mainbooktitle>
    <booktitlealt>Tasks and what they can do</booktitlealt>
  </booktitle>
  <bookmeta>
    <author>Howe Tuduit</author>
    <critdates>
      <created date="1/1/2015"/>
      <revised modified="3/4/2016"/>
      <revised modified="3/5/2016"/>
    </critdates>
    <bookid>
      <isbn>071271271X</isbn>
      <booknumber>SG99-9999-00</booknumber>
    </bookid>
  </bookmeta>
</bookmap>
```

The entire `<booktitle>` element content is displayed on the first page of the PDF, so if you need to add the information after it, in your customization CSS (on page 1814), add the following CSS rules:

```css
*[class ~="bookmap/booktitle"]:after {
  display: block;
  content: " by oxy_xpath('//*[contains(@class, " bookmap/bookmeta ")/*[contains(@class, " topic/author ")])/text()');
  margin-top: 4em;
  text-align: center;
  color: gray;
}
*[class ~="bookmap/booktitle"]:after(2) {
  display: block;
  content: oxy_xpath('if//*[contains(@class, " bookmap/isbn ")]) then concat("ISBN ", //*[contains(@class, " bookmap/isbn ")]/text()) else ");
  text-align: center;
  color: gray;
}
```
Processing Metadata for DITA Maps

Suppose you need to present the Revision Date just under the publication title and suppose your DITA map contains:

```xml
<map>
  <title>Growing Flowers</title>
  <topicmeta>
    <critdates>
      <revised modified="2021-04-26"/>
    </critdates>
  </topicmeta>
  ...
</map>
```

The entire `<title>` element content is displayed on the first page of the PDF. If you need to add the information after it, add the following CSS rules in your customization CSS (on page 1814):

```css
/*[class ~= "front-page/front-page-title"] > *{class ~= "topic/title"}:after {
  display: block;
  content: "last revision " oxy_xpath('//*[contains(@class, " map/topicmeta ")]()[1] \
  //*[contains(@class, " topic/revised ")]/@modified');
  margin-top: 4em;
  text-align: center;
  color: gray;
}
```

**Note:**
The `[1]` predicate is used to avoid duplicated results as the `topicmeta` is included in all children topics.

Generating Synthetic Pages for Metadata

Suppose you need to show this information on a page that follows the title page, instead of on the title page. In this case, you need to prepare a named page and place the content in it. Add the following rules in your customization CSS (on page 1814):

```css
@page page-for-meta {
  background-color: yellow; /* Just to see it better*/
  @top-left-corner {
    content:""; /* Remove the default header */
  }
  @top-right-corner {
    content:""; /* Remove the default header */
  }
}
/*[class ~= "bookmap/booktitle"]:after {
```
How to Show Metadata in the Header or Footer

The header and footer are composed of page margin boxes that can be populated with static text by using *string-sets*.

If you need to add some of the map metadata to the header of the front page (for example, the *creation date*), add the following CSS rules in your customization CSS (on page 1814):

```css
*[class ~="bookmap/booktitle"]:after(2) {
    page: page-for-meta;
}

@page front-page {
    @top-center {
        content: "Created: " string(mapcreated);
    }
}
```

**Note:**
The *front-page* is the name of a page that used to present the element with the class "front-page/front-page". The above page rule is combined with the default styles.

How to Remove or Change the PDF Keywords

The keywords defined in the prolog sections of topics are automatically collected and set as PDF keywords. These are shown by the readers in the PDF document properties window.

If you need to remove them, you can use the following CSS snippet in your customization CSS (on page 1814):

```css
:root {
    -oxy-pdf-meta-keywords: "";
}
```
To change them, if you have a hard-coded list, you just enumerate each of them in the property content, separating them with comma:

```css
:root {
    -oxy-pdf-meta-keywords: "alpha, beta, gamma";
}
```

If you need to extract them by other criteria from the merged map, you can use the `oxy_xpath()` function instead of the hard-coded list.

### How to Remove the PDF Publication Title Property

The title defined in the PDF reader is automatically collected from the map's main title.

If you want to display the map name instead of the title, you can use one of the following rules in your customization CSS (on page 1814):

```css
/* Titles (maps).
 */
*[class =~ "front-page/front-page-title"] *[class =~ "topic/title"]:not([class =~ 'bookmap/booktitle']) {
    -oxy-pdf-meta-title: unset;
}

/* Titles (bookmaps).
 */
*[class =~ "front-page/front-page"] *[class =~ "bookmap/booktitle"] > *[class =~ "bookmap/mainbooktitle"] {
    -oxy-pdf-meta-title: unset;
}
```

### How to Change the PDF Publication Title Property

The `<title>` element of a bookmap is quite complex and contains elements for the book library and an alternate title:

```xml
<booktitle>
    <booklibrary>Retro Tools</booklibrary>
    <mainbooktitle>Main Book Title</mainbooktitle>
    <booktitlealt>Book Title Alternative</booktitlealt>
</booktitle>
```

For the publication title, the built-in CSS uses only the content of the `<mainbooktitle>`. If you want to collect all of the text from the `<booktitle>`, you can add the following rule to your customization CSS (on page 1814):
An XPath expression is used to collect all the `<booktitlealt>` elements from the merged map, select the first one, then use its text.

The built-in CSS uses the `<booktitlealt>` as the PDF description. In the example above, this property is cleared since it was moved as a title.

## How to Use Data Elements from the Map to Create Custom PDF Metadata

To use a key value in the CSS, the key must be referenced from the content (either a topic or map). If you do not have it referenced, you may force a reference by using the `<topicmeta>` or `<bookmeta>` section of your map and a `<data>` element. This has no effect on the published content, but allows the CSS rules to use its content.

```
<bookmeta>
  ...
  <data keyref="my_key"/>
  ...
</bookmeta>
```

This is expanded in the merged HTML file to:

```
<div class="- map/topicmeta bookmap/bookmeta topicmeta bookmeta">
  ...
  <div keyref="my_key" class="- topic/data data">
    <div class="- topic/keyword keyword">KEY VALUE</div>
  </div>
  ...
</div>
```

Suppose that you need the expanded key value in the footer of the publication. You can define a string-set on this `<data>` element:

```
*@[class = "topic/data"]{keyref="my_key"] {
  string-set: key-string content(text);
}
@page {
  @bottom-left {
    content: "My key is: " string(key-string) !important;
  }
}
```
Or you can use the value from a :before pseudo-element, like the one for the title:

```css
*{class =~ "topic/title"}:before { 
  content: oxy_xpath("//*[contains(@class, 'topic/data')][@keyref = 'my_key']/text()");
}
```

Another use-case is to use the key as a source for a custom PDF document property:

```css
*{class =~ "topic/data"}[keyref="my_key"] { 
  -oxy-pdf-meta-custom: attr(keyref) content(text);
}
```

### How to Control the PDF Viewer

The PDF document may contain settings for the PDF Viewer. This helps to make the viewing experience common for all of the readers. For example, you can specify the zoom level that the document is presented, or whether the outline view should be displayed.

There are several CSS properties you can use. These properties should be set on the root element. If they are set on multiple elements, the first one will be taken into account.

#### Examples

- To hide the PDF Viewer toolbar and menu bar:

  ```css
  :root { 
    -oxy-pdf-viewer-hide-menubar: true;
    -oxy-pdf-viewer-hide-toolbar: true;
  }
  ```

- To make the document be displayed with a different zoom level:

  ```css
  :root { 
    -oxy-pdf-viewer-zoom: 50%;
  }
  ```

- To make the PDF Viewer just as large as the displayed document (e.g. if there is a zoom level that makes the document smaller, then the window of the viewer will be just as big as the page):

  ```css
  :root { 
    -oxy-pdf-viewer-fit-window: true;
  }
  ```

- If you need the pages to be displayed as a single continuous column (to be able to scroll in a single view port), use:

  ```css
  :root { 
    -oxy-pdf-viewer-page-layout: one-column;
  }
  ```
The supported include: single-page, two-columns-left, and more.

- To make the document outline view visible, use:

```xml
:root {
    -oxy-pdf-viewer-page-mode: use-outlines;
}
```

The supported values include: use-thumbs, use-none. For more details, see the list of Chemistry extension CSS properties.

Front Matter and Back Matter

The front matter is a series of topics that are usually placed after the cover page and before the TOC or the content.

The back matter is a series of topics that are usually placed after the content of the book.

Front Matter and Back Matter - XML Fragment

In the merged map file (on page 1816), the frontmatter topic references are wrapped in a `<frontmatter>` element that has the class `bookmap/frontmatter`. Then, the referenced content is marked with the attribute `@is-frontmatter="true"`.

```xml
<bookmap xmlns:ditaarch="http://dita.oasis-open.org/architecture/2005/" ...
<oxy:front-page class="- front-page/front-page ">

<opentopic:map xmlns:ot-placeholder="http://suite-sol.com/namespaces/ot-placeholder" class="- toc/toc ">

<frontmatter xmlns:dita-ot="http://dita-ot.sourceforge.net/ns/201007/dita-ot" class="- map/topicref bookmap/frontmatter ">

<topicref class="- map/topicref " href="#unique_1" type="concept">

</concept>
```
Note:
The process also applies for the backmatter topic references inside a `<backmatter>` element with the `bookmap/backmatter` class and referenced content with the `@is-backmatter="true"` attribute both in the merged map and merged HTML files.

Front Matter and Back Matter - Built-in CSS

The built-in CSS rules are in `[PLUGIN_DIR]/css/print/p-bookmap-frontmatter-backmatter.css`. By default, it associates the top-level topics that do not represent chapters to a `matter-page` style of page layout. Each child topic starts on a new page.

Related Information:
Page Headers and Footers (on page 1828)

How to Remove Page Breaks Between Front Matter Child Topics

If you do not like the fact that all the topics that enter a bookmap frontmatter start on a new page, you can disable this by using the following rules in your customization CSS (on page 1814):

```css
/*[class =~ "map/map"] > *[class =~ "topic/topic"][is-frontmatter]{
  page-break-before: auto;
}
```

How to Style the Front Matter and Back Matter Topics

Style all the Topics with the Same Aspect

All the topics referenced from the `<frontmatter>` and `<backmatter>` bookmark elements are formatted using the `matter-page` as defined in Default Page Definitions (on page 1821). In the merged file, the `<backmatter>` and
<frontmatter> elements are omitted, and their child topic content is matched using a CSS rule like the one below:

```css
* [class =~ "map/map"] > * [class =~ "topic/topic"] [is-backmatter],
* [class =~ "map/map"] > * [class =~ "topic/topic"] [is-frontmatter] {
  page: matter-page;
  ...
}
```

**Style the Topics Depending on Their Role**

There might be cases when you need to distinguish between certain types of topics that have different roles in your publication:

- Preface
- Notice
- Abstract
- Copyright

These are referenced from the DITA map by specialized `<topicref>` elements, with different class attribute values.

The class attribute values are then passed by the transformation process onto the corresponding topic elements from the merged map content. For example, a topic that was referenced by a `<preface>` map element now has a "bookmap/preface" value in its `@topicrefclass` attribute:

```
<topic
  class="- topic/topic 
  id="unique_1"
  topicrefclass="- map/topicref bookmap/preface 
  ...
...
</topic>
```

This can be used to match and apply various styling choices, or even a particular page layout:

```css
@page preface-page {
  background-color:silver;
  @top-center{
    content: "Custom Preface Header";
  }
}

* [class =~ "topic/topic"][@topicrefclass =~ "bookmap/preface"] {
  page: preface-page;
}
```
Numbering

The topics in this section contain some technical details in case you need to fine-tune the way the numbering works.

Numbering - Built-in CSS

The built-in CSS rules are in:

- [PLUGIN_DIR]/css/print/p-numbering-shallow.css
- [PLUGIN_DIR]/css/print/p-numbering-deep.css
- [PLUGIN_DIR]/css/print/p-numbering-deep-chapter-scope.css
- [PLUGIN_DIR]/css/print/p-numbering-deep-chapter-scope-no-page-reset.css

The first CSS (shallow) contains rules that add a "Chapter NN" before the first-level topics from the publication, the second one (deep) contains rules that add a deep structure of counters on all topics referenced from the map (at any level), the third one (chapter-scope) creates a chapter scope-oriented numbering (meaning that the numbering for pages, tables, figures, and links to them are reset for each chapter), and the last one is similar to the third except that page numbers do not reset. For more details, see Numbering Types (on page 1883).

Numbering - Input XML Fragments

The numbering affects multiple logical parts of your publication, the table of contents, headers/footers, chapter titles, figures and tables titles:

The Table of Contents

The table of contents is a tree of `<topicref>` elements.

```
<map xmlns:ditaarch="http://dita.oasis-open.org/architecture/2005/" ...>
                  class=" front-page/front-page ">
    ...
  </oxy:front-page>

  <opentopic:map xmlns:opentopic="http://www.idiominc.com/opentopic" class=" toc/toc ">
    <title class="- topic/title ">Publication Title</title>

    <topicref is-chapter="true" class="- map/topicref " ...
    ...
    <topicmeta class="- map/topicmeta ">
      <navtitle href="#unique_1" class="- topic/navtitle ">Overview</navtitle>
      ...
    </topicmeta>

    <topicref class="- map/topicref ">
      <topicmeta class="- map/topicmeta " data-topic-id="dcpp_resources">
        <navtitle href="#unique_2" class="- topic/navtitle ">Resources</navtitle>
```
Note: The `<opentopic:map>` element contains the effective table of contents structure.

Note: The TOC items are the elements with the class `- map/topicref`.

Note: The ones identified as chapters have the `@is-chapter` attribute set.

For the DITA Map PDF - based on HTML5 & CSS transformation type, the merged map is further processed resulting in a collection of HTML5 `<div>` elements. These elements preserve the original DITA `@class` attribute values and add a new value derived from the DITA element name.

```html
<div class="- map/map map" ...>
    ...
    </div>
</div>

<div class="- toc/toc toc">
    <div class="- topic/title title">Publication Title</div>
    ...
</div>

<div is-chapter="true" class="- map/topicref topicref" ...>
    ...
    <div href="#unique_1" class="- topic/navtitle navtitle">Overview</div>
    ...
</div>

<div class="- map/topicref " ...>
    ...
    <div href="#unique_2" class="- topic/navtitle ">Resources</div>
    ...
</div>
</div>

...
The Header and Footers

These are based on string sets generated for the titles. The complete set of strings is defined in:

[INSTALLATION_DIR]/css/print/p-pages-and-headers.css.

The CSS rules that build the string sets are matching the map title from the front page and the titles from the content.

```xml
    <oxy:front-page-title>
        <title class="- topic/title ">Publication Title</title>
    </oxy:front-page-title>
</oxy:front-page>
```

For the DITA Map PDF - based on HTML5 & CSS transformations:

```xml
<div class=" front-page/front-page front-page">
    <div class=" front-page-title/front-page-title front-page-title">
        <div class="- topic/title title ">
            Publication Title
        </div>
    </div>
</div>
```

The main content is organized as follows:

```xml
<map xmlns:ditaarch="http://dita.oasis-open.org/architecture/2005/" ...>
    ...
    <opentopic:map xmlns:opentopic="http://www.idiominc.com/opentopic" ...
        ...
    </opentopic:map>
    ...

    <topic is-chapter="true" oid="dcpp_overview">
        <title class="- topic/title ">Overview</title>
        <body class="- topic/body ">
            ...
        </body>
    </topic>
    ...

    <topic class="- topic/topic " id="unique_2" oid="dcpp_resources">
        <title class="- topic/title ">Resources</title>
        ...
    </topic>
    ...

    <topic class="- topic/topic " id="unique_2" oid="dcpp_parameters">
        <title class="- topic/title ">Parameters</title>
        ...
    </topic>
    ...
```
For the **DITA Map PDF - based on HTML5 & CSS** transformations:

```xml
<div class="map/map map" ...>
  ...
  <div class="toc/toc toc">
    ...
  </div>
  
  <div is-chapter="true" oid="dcppOverview" class="- topic/topic topic">
    <div class="- topic/title title">Overview</div>
    <div class="- topic/body body">...
    </div>
  </div>
  
  <div class="- topic/topic topic" id="unique_2" oid="dcppResources">
    <div class="- topic/title title">Resources</div>
    ...
  </div>
  
  <div class="- topic/topic topic" id="unique_2" oid="dcppParameters">
    <div class="- topic/title title">Parameters</div>
    ...
  </div>
</div>
```

**Note:**
The topic content comes after the `<opentopic:map>` element.

**Note:**
The child topics are the elements that have the class `- topic/topic` included in the parents.

**Note:**
The ones identified as chapters have the `@is-chapter` attribute set.

### The Titles of Chapters

The titles from the content are children of the topics:

```xml
<topic class="- topic/topic" id="unique_2" oid="dcppParameters">
  <title class="- topic/title">Parameters</title>
  ...
</topic>
```
For the DITA Map PDF - based on HTML5 & CSS transformations:

```html
<div class="- topic/topic topic" id="unique_2" oid="dcpp_parameters">
  <div class="- topic/title title ">
    Parameters</div>
  ...
</div>
```

**Note:**
The title elements have the class: `- topic/title`. The actual element name can be different.

### Numbering Types

The type of numbering that appears in your publication is controlled by the `args.css.param.numbering` parameter.

This parameter activates various sets of CSS rules from the built-in CSS. By default, only the first-level topics (the chapters) are numbered (shallow numbering). The following values are accepted:

**Table 42. Types of Numbering**

<table>
<thead>
<tr>
<th>Value</th>
<th>Chapters</th>
<th>Sections/ Nested Topics</th>
<th>Figures &amp; Tables</th>
<th>Pages</th>
</tr>
</thead>
<tbody>
<tr>
<td>shallow</td>
<td>numbered</td>
<td>no</td>
<td>counted from the start of the publication</td>
<td>from the start of the publication</td>
</tr>
<tr>
<td>deep</td>
<td>numbered</td>
<td>numbered</td>
<td>counted from the start of the publication</td>
<td>from the start of the publication</td>
</tr>
<tr>
<td>deep-chapter-scope</td>
<td>numbered</td>
<td>numbered</td>
<td>numbering is restarted at the beginning of each chapter, adds the chapter number in their titles (and in the links to them), and in the list of tables and list of figures sections</td>
<td>restarted at the beginning of each chapter</td>
</tr>
<tr>
<td>deep-chapter-scope-no-page-reset</td>
<td>numbered</td>
<td>numbered</td>
<td>numbering is restarted at the beginning of each chapter, adds the chapter number in their titles (and in the links to them), and in the list of tables and list of figures sections</td>
<td>from the start of the publication</td>
</tr>
</tbody>
</table>
Note:
When using any of the deep numbering types, no distinction is made between sections and nested topics. For example, if a topic contains two sections, followed by another nested topic, the sections will be numbered with 1 and 2, and the nested topic with 3.

Notice:
The deep-chapter-scope and deep-chapter-scope-no-page-reset values are only available for the DITA Map PDF - based on HTML5 & CSS transformation scenario.

Examples

Shallow

Each chapter (or first-level topic) is numbered, but sections/nested topics are not numbered. Figures, tables, and pages are numbered sequentially from the start of the publication and they do not reset.

Chapter 1. First Chapter
  Page 1
  Topic
  Section
  Table 1
  Table 2
  Topic
  Section
  Page 2
  Table 3
Chapter 2. Second Chapter
  Page 3
  Topic
  Table 4
  Table 5
  Topic
  Page 4

It will result in the following content inside the PDF:

Chapter 1. Introduction..............................................................1
Chapter 2. Care and Preparation......................................................2
  Pruning............................................................................2
  Garden Preparation..................................................................3
Chapter 3. Flowers by Season.........................................................4
  Spring Flowers.....................................................................4
  Iris.............................................................................4
Deep

All chapters (or first-level topics) and sections/nested topics are numbered (these are also prefixed with the chapter number). Figures, tables, and pages are numbered sequentially from the start of the publication and they do not reset.

1. First Chapter
   Page 1
   Topic 1.1
   Table 1
   Topic 1.2
   Table 2
   Page 2
   Table 3

2. Second Chapter
   Page 3
   Topic 2.1
   Table 4
   Table 5
   Topic 2.2
   Page 4

It will result in the following content inside the PDF:

1. Introduction......................................................................1
2. Care and Preparation..............................................................2
   2.1. Pruning.......................................................................2
   2.2. Garden Preparation.......................................................3
3. Flowers by Season.................................................................4
   3.1. Spring Flowers...............................................................4
      3.1.1. Iris......................................................................4
      3.1.2. Snowdrop................................................................6
      ...

List of Figures
   Figure 1: Iris
List of Tables
Table 1: Flowers

Deep Chapter Scope

Each chapter (or first-level topic) is independent (so it can be read separately, as a separate part of your publication). The sections/nested topics, pages, figures, and table counters (and links to them) restart at each chapter. The general cross reference links also display the chapter number before the page number to clearly specify the target.

1. First Chapter
   Page 1.1
   Topic 1.1
   Table 1-1
   Link to page 2.2
   Topic 1.2
   Page 1.2
   Table 1-2
2. Second Chapter
   Page 2.1
   Topic 2.1
   Table 2-1
   Table 2-2
   Table 2-3
   Topic 2.2
   Table 2-4
   Page 2.2
   Link to page 1.1

It will result in the following content inside the PDF:

1. Introduction......................................................................1
2. Care and Preparation..............................................................1
   2.1. Pruning.......................................................................1
   2.2. Garden Preparation............................................................2
3. Flowers by Season.................................................................1
   3.1. Spring Flowers................................................................1
      3.1.1. Iris......................................................................1
      3.1.2. Snowdrop.................................................................3
   ...

List of Figures
Figure 3-1: Iris
Deep Chapter Scope No Page Reset

Each chapter (or first-level topic) is independent (so it can be read separately, as a separate part of your publication). The sections/nested topics, figures, and table counters (and links to them) restart at each chapter, but the page numbers do not reset. The generic cross reference links contain only the page number.

1. First Chapter
   Page 1
   Topic 1.1
   Table 1-1
   Link to page 4
   Topic 1.2
   Page 2
   Table 1-2

2. Second Chapter
   Page 3
   Topic 2.1
   Table 2-1
   Table 2-2
   Table 2-3
   Topic 2.2
   Table 2-4
   Page 4
   Link to page 1

It will result in the following content inside the PDF:

1. Introduction ................................................................. 1
2. Care and Preparation .................................................... 2
   2.1. Pruning ................................................................. 2
   2.2. Garden Preparation ............................................... 3
3. Flowers by Season ...................................................... 4
   3.1. Spring Flowers ..................................................... 4
       3.1.1. Iris ............................................................. 4
       3.1.2. Snowdrop .................................................... 6
...

List of Figures
   Figure 3-1: Iris
How to Reset Page Numbering at First Chapter

By default, pages are numbered from the start of the publication, but in some cases, you may need to restart the page number at a certain location in your content (for example, at the first part/chapter).

**Note:**
The following sections do not apply for `args.css.param.numbering="deep-chapter-scope"` and `args.css.param.numbering="deep-chapter-scope-no-page-reset"` parameter values because they already define a specific numbering scheme.

### Reset Page Numbering in Shallow Context

To reset the page counter at the first part/chapter when the `args.css.param.numbering="shallow"` parameter value is set, use the following rules in your customization CSS *(on page 1814)*:

```
*[class = "map/map"] > *[class = "toc/toc"] + *[class = "topic/topic"][is-chapter],
*[class = "map/map"] > *[class = "topic/topic"][is-frontmatter] + *[class = "topic/topic"][is-chapter] {
  counter-reset: page 1;
}
*[class = "map/map"] > *[class = "toc/toc"] + *[class = "topic/topic"][is-part],
*[class = "map/map"] > *[class = "topic/topic"][is-frontmatter] + *[class = "topic/topic"][is-part] {
  counter-reset: page 1 chapter;
}
```

### Reset Page Numbering in Deep Context

To reset the page counter at the first part/chapter when the `args.css.param.numbering="deep"` parameter value is set, use the following rules in your customization CSS *(on page 1814)*:

```
*[class = "map/map"][numbering ^= "deep"] > *[class = "toc/toc"] + *[class = "topic/topic"][is-chapter],
*[class = "map/map"][numbering ^= "deep"] > *[class = "topic/topic"][is-frontmatter] + *[class = "topic/topic"][is-chapter] {
  counter-reset: page 1 section1;
}
```
How to Remove the 'Chapter NN' Prefix from Table of Contents and Content.

If you are using the shallow numbering CSS (this is the default), and you want to hide the "Chapter" prefix, use the following rules in your customization CSS (on page 1814):

For the TOC:

```css
[class ~="map/topicmeta"] > *[class ~="topic/navtitle"]:before {
  display:none !important;
}
```

For the titles in the content, change the content of the title :before (this example just uses the chapter number):

```css
*[class ~="topic/topic"]{is-chapter}:not({is-part}) > *[class ~="topic/title"]:before {
  content: counter(chapter) ". " !important;
}
```

How to Use Part, Chapter, and Subtopics Numbers in Links

This topic is applicable if you have enabled deep numbering (on page 1883). Suppose you have a link in the third chapter that points to a paragraph in the second subtopic of the first chapter and you need this structural information (1.2) presented to the user, just after the link text. To do this, you can use the target-counters CSS function to extract the entire context of the counters from the target. The chapter-and-sections built-in counter is already updated with both the chapter number and the nested topics:

```css
*[class ~="topic/xref"]:after {
  content: target-counters(attr(href), chapter-and-sections, ".") !important;
}
```

This counter does not include the part number, so be careful when linking between parts (consider adding the target part number explicitly):

```css
*[class ~="topic/xref"]:after {
  content: "" target-counter(attr(href), part, upper-roman) "" target-counters(attr(href), chapter-and-sections, ".") "" !important;
  color:blue;
}
```
How to Include Topic Sections in TOC

To include topic sections in the table of contents, set the args.css.param.numbering-sections transformation parameter (on page 1773) to yes. In this case, they are numbered according the numbering scheme set by the args.css.param.numbering parameter (on page 1883).

If you want to prevent topic sections from being numbered in your output, set the value of the args.css.param.numbering-sections parameter to no.

Table of Contents

The table of contents is a hierarchy of topic titles with links to the topic content.

For plain maps, the TOC is automatically generated. For DITA bookmaps, you will need to add a <toc> element in the <booklists> element (inside the <frontmatter>):

```xml
<bookmap>
  ...
  <frontmatter>
    <booklists>
      <toc/>
      <figurelist/>
      <tablelist/>
    </booklists>
  </frontmatter>
  ...
  ...
</bookmap>
```

Table of Contents - XML Fragment

In the merged map file (on page 1816), the <opentopic:map> contains a hierarchy of <topicref> elements, or other elements (such as <chapter> or <part>) that are specializations of <topicref>.

Each of the <topicref> elements include a metadata section that includes the topic title.
For the DITA Map PDF - based on HTML5 & CSS transformation type, the merged map is further processed resulting in a collection of HTML5 `<div>` elements. These elements preserve the original DITA `@class` attribute values and add a new value derived from the DITA element name.
<div class="toc/toc-title toc-title" empty="true"/>

Retro Tools</div>

Tasks</div>

Product Tasks</div>

Installing

Installing

Installing

Note:
The <oxy:toc-title> element is used as a placeholder for the name of the TOC. For instance, you can use the string "Contents", specified on a pseudo-element, in the CSS.

Table of Contents - Built-in CSS

The built-in CSS rules are in: [PLUGIN_DIR]/css/print/p-toc.css.

Related Information:

Page Headers and Footers (on page 1828)

How to Increase TOC Depth

By default, only the first three levels of topics are displayed in the Table of Contents of the PDF output.

The CSS rule (see Table of Contents - Built-in CSS (on page 1892)) that hides topics on higher levels is:
/* Hide sections below level 3. */
*[class ~="map/topicref"]{is-chapter} >
*[class ~="map/topicref"]:not([is-chapter]) >
*[class ~="map/topicref"] >
*[class ~="map/topicref"] {
  display: none;
}

If you want to increase the TOC depth so that topic references on level 3 or higher are visible, you can overwrite this rule in your customization CSS like this:

*[class ~="map/topicref"]{is-chapter} >
*[class ~="map/topicref"]:not([is-chapter]) >
*[class ~="map/topicref"] >
*[class ~="map/topicref"] {
  display: block;
}

If the args.css.param.numbering parameter is set to a value other than shallow, you also need to add the following rules in your customization CSS:

*[class ~="map/map"]{numbering ^= 'deep'}
*[class ~="map/topicref"]{is-chapter}:not([is-part]) >
*[class ~="map/topicref"] >
*[class ~="map/topicref"]
*[class ~="map/topicref"] {
  counter-increment: toc-chapter-and-sections;
}

*[class ~="map/map"]{numbering ^= 'deep'}
*[class ~="map/topicref"]{is-chapter}:not([is-part]) >
*[class ~="map/topicref"] >
*[class ~="map/topicref"]
*[class ~="map/topicref"]
*[class ~="map/topicmeta"] + *[class ~="map/topicref"] {
  counter-reset: toc-chapter-and-sections;
}

*[class ~="map/map"]{numbering ^= 'deep'}
*[class ~="map/topicref"]{is-chapter}:not([is-part]) >
*[class ~="map/topicref"] >
*[class ~="map/topicref"]
*[class ~="map/topicref"] > *[class ~="map/topicmeta"]:before {
content: counters(toc-chapter-and-sections, ".") ". ";

How to Style the Table of Contents Entries

Note:
Each of the items from the table of contents is an element that has the map/topicref class.

The following example uses the italic font for the label and changes the color and style of the connecting line between the title and the page number.

In your customization CSS (on page 1814), add the following two selectors:

/* The toc item label - the topic title */
*[class ~="map/topicref"] *[class ~="topic/navtitle"] {
    font-style:italic;
    color: navy;
}

/* The dotted line between the topic name and the page number. */
*[class ~="map/topicref"] *[class ~="topic/navtitle"]:after {
    content: leader('-') target-counter(attr(href), page);
    color: navy;
}

And if you need to alter the indent of the nested table of content items, use the following selector:

*[class ~="map/topicref"] *[class ~="map/topicref"] {
    margin-left: 1em;
}

The numbers can be styled like this:

*[class ~="map/topicref"] > *[class ~="map/topicmeta"]:before,
*[class ~="map/topicref"]
    > *[class ~="map/topicmeta"] > *[class ~="topic/navtitle"]:before{
        color:blue;
    }

The following is an example of customizing the font size for the items representing chapters. The chapters are level one topics and are marked in the merged DITA document TOC with the attribute @is-chapter.

*[class ~="map/topicref"][is-chapter = "true"] > *[class ~="map/topicmeta"] > *[class
    ~="topic/navtitle"][
        font-size:2em;
    ]
How to Change the Header of the Table of Contents

In the built-in CSS, there is a page named table-of-contents. The default is to have the word 'Contents' in its header (this is localized, using the toc-header string defined in the p-18n.css) alternating in the left or right side of the header:

```css
@page table-of-contents:left {
  @top-left {
    content: string(toc-header) " | " counter(page, lower-roman);
    font-size: 8pt;
  }
}
@page table-of-contents:right {
  @top-right {
    content: string(toc-header) " | " counter(page, lower-roman);
    font-size: 8pt;
  }
}
```

If you need to change this string, or change the color, you should use the following `@page` selectors as a starting point in your customization CSS (on page 1814):

```css
@page table-of-contents:left {
  @top-left {
    content: "My publication table of contents | " counter(page, lower-roman);
    color:red;
  }
}
@page table-of-contents:right {
  @top-right {
    content: "My publication table of contents | " counter(page, lower-roman);
    color:red;
  }
}
```

⚠️ **Important:**

The first page from the table of contents does not have any content displayed in the header. The default CSS contains rules that disable the content. If you need to also display the numerals on the first page, use the following:

```css
@page table-of-contents:first:left {
  @top-left {
    content: string(toc-header) " | " counter(page, lower-roman);
  }
}
```
How to Make the Table of Contents Start on an Odd Page

In your customization CSS (on page 1814), add the following snippet for the table-of-contents page:

```
@page table-of-contents{
  oxy-initial-page-number: auto-odd;
}
```

How to Display a Topic Before the Table of Contents

To display a topic before the table-of-contents page, follow these steps:

1. Make sure the topic is referenced on the first level in the DITA map.
2. Set the @outputclass to before-toc on the <topicref>.

```
<topicref href="pathToMyTopic" outputclass="before-toc">
```

Result: When the PDF is processed, the topic will automatically appear before the table of contents.

How to Display Short Descriptions in the TOC

To display the short descriptions from the topics in the table of contents, you need to make the <shortdesc> element visible.

The following example only makes the short descriptions associated with the chapters visible. The chapters are level one topics and are marked in the merged DITA document TOC with the attribute @is-chapter.

In your customization CSS (on page 1814), add the following CSS selector:
How to Remove Entries from the TOC

To remove entries from the table of contents, set the `@toc="no"` attribute on the topicrefs from the map that need to be removed. This is sometimes desirable for the topics listed in the frontmatter or backmatter when using a bookmap.

How to Hide the TOC

To hide the TOC, you have multiple options:

- [Recommended] Use a DITA `<bookmap>` instead of a `<map>`, and omit the `<toc>` element from the `<booklists>`. An example bookmap can be found in the DITA 1.3 Spec.
- Use the transformation parameter: `hide.frontpage.toc.index.glossary (on page 1777)`.  
- Use a `display:none` property to hide the element that contains the TOC structure, and also remove it from the PDF bookmarks tree:

```
*[class ~= "map/map"] > *[class ~= "toc/toc"] {  
  display:none;
}
```  

```
*[class ~= "map/map"] > *[class ~= "toc/toc"] > *[class ~= "toc/title"]{  
  bookmark-label: none;
  -ah-bookmark-label: none;
}
```

Related Information:

Transformation Parameters (on page 1773)

Table of Contents on a Page (Mini TOC)

To add a mini table of contents for each chapter, you need to:

- Use DITA bookmaps instead of regular maps.
- Set the `args.chapter.layout` transformation parameter to either of the following values: MINITOC or MINITOC-BOTTOM-LINKS.
Note:
If the chapter does not have child topics, it will not have a mini TOC in the PDF output.

Layout for MINITOC

This table of contents is positioned between the chapter title and the chapter child topics. It consists of a list of links pointing to the child topics, positioned in the left side of the page, and a description in the right side. This content is collected from the topic file referenced by the chapter <topicref> in the map.

Chapter 1. Introduction

Topics:

About this framework.

Description

About this framework.
The framework is DITA.

Description

The framework is composed by a large set of modules.

Layout for MINITOC-BOTTOM-LINKS

This table of contents is positioned between the chapter title and the chapter child topics. It consists of a chapter description and list of links pointing to the child topics, under the description. This description is collected from the topic file referenced by the chapter <topicref> in the map.
Chapter 1. Introduction

DITA Open Toolkit, or DITA-OT for short, is a set of Java-based, open-source tools that provide processing for content authored in the Darwin Information Typing Architecture.

The DITA Open Toolkit documentation provides information about installing, running, configuring and extending the toolkit.

Topics:

About this framework.
Description

About this framework.

The framework is DITA.

Description

The framework is composed by a large set of modules.

The above chapter example has the following DITA map fragment:

```xml
<chapter href="topics/chapter-introduction.dita">
  <topicref href="topics/introduction-about.dita" />
  <topicref href="topics/introduction-description.dita" />
</chapter>
```

The `chapter-introduction.dita` file provides the description content that is in the right side of the page. The children `<topicref>` elements generate the mini TOC links.

Table of Contents for Chapters (Mini TOC) - XML Fragment

In the merged XML file, the mini TOC is built from a related links section and some `<div>` elements that wrap the entire mini TOC and the description area.

`chapter/minitoc`

Wraps the entire structure, including the content of the chapter `<topicref>`.

`chapter/minitoc-links`

Wraps the `<related-links>` element. Note that the label of the related links list is internationalized.
chapter/minitoc-desc

Contains the entire content of the topic file referenced by the chapter <topicref> element in the map.

When using the pdf-css-html5 transformation, this structure is converted to a set of HTML elements, preserving the class values:
Table of Contents for Chapters (Mini TOC) - Built-in CSS

The built-in CSS rules are in: [PLUGIN_DIR]/css/print/p-chapters-minitoc.css.

How to Style the Table of Contents for Chapters (Mini TOC)

Suppose that you do not want the links and the chapter description to be side by side, but instead place the links above the description. Also, you may choose to remove the label above the links and put all the links in a colored rectangle with decimal numbers before them.

In your customization CSS (on page 1814), add the following selectors:
/* Change from inline to blocks to stack them one over the other. */

*[class="chapter/minitoc-desc"],
*[class="chapter/minitoc-links"] {
  display: block;
  width: 100%;
}

/* No need for the 'Topics:' label. */
*[class="chapter/minitoc-links"] *[class="topic/desc"] {
  display: none;
}

/* Add background for the links list. */
*[class="chapter/minitoc-links"] {
  background-color: silver;
  padding: 0.5em;
}

/* Remove the border and the padding from the description. We do not need that separator. */
*[class="chapter/minitoc-desc"] {
  border-left: none;
  padding-left: 0;
}

/* Add a number before each of the links. */
*[class="chapter/minitoc-links"] *[class="topic/link"] {
  display: list-item;
  list-style-type: decimal;
  margin-left: 1em;
}

Related Information:
How to Speed up CSS Development and Debugging (on page 1819)

List of Tables/Figures

To activate these:

1. The map must be a DITA bookmark.
2. There must be a `{figurelist}` or `{tablelist}` in the *frontmatter* or *backmatter*. In the following example, both of the lists are added just after the table of contents (the `{toc}` element is the placeholder where the table of contents will be created):
How to Set a Header for a List of Tables/Figures

Suppose you want to set the headline "Figure List" on the second and subsequent pages associated to a list of figures and something similar for a list of tables.

Start by associating pages to the list of figures and tables from the merged file:

```
*{[class~="placeholder/tablelist"]} {
  page:tablelist;
  color:green;
}
*{[class~="placeholder/figurelist"]}{
  page:figurelist;
  color:green;
}
```

Note:
The "placeholder/tablelist" is the class name of the output generated from the `<tablelist>` bookmap element.

Then define the pages:

```
@page figurelist {
  @top-left { content: none; }
  @top-center { content: "Figure List"; }
  @top-right { content: none; }
}

@page figurelist:first {
  @top-left { content: none; }
  @top-center { content: none; }
  @top-right { content: none; }
}

@page tablelist {
```
How to Remove the Numbers Before a List of Tables or Figures

Suppose you need to remove the "Figure NN" prefix before each entry of a list of figures.

An entry in the generated list of figures from the merged map looks like this:

```xml
<entry class="- listentry/entry " href="#unique_6_Connect_42_fig_rjy_spn_xgb">
  <prefix class="- listentry/prefix ">Figure</prefix>
  <number class="- listentry/number ">4</number>
  <title class="- topic/title ">This is another figure</title>
</entry>
```

For the HTML merged map, the element names are all `<div>` elements but they have the same class.

So, to hide the label and the number, use:

```css
* [class~="listentry/prefix"],
* [class~="listentry/number"] {
  display: none;
}
```

This works for both a list of tables and list of figures since the structure of each entry is the same.

To make it more specific (for example, to apply it only for the list of figures), you can add the selector:

```css
* [class="placeholder/figurelist"] * [class="listentry/prefix"],
* [class="placeholder/figurelist"] * [class="listentry/number"] {
  display: none;
}
```

Double Side Pagination

By default, the processor generates pages that are mirror images (the right page has the header on the right side, the left pages have the header on the left side). The chapters follow one another with no constraint on the page side.
Note:
For a plain DITA map, the chapters are the `<topicref>` elements that are placed on the first level. For bookmaps, the chapters are the topics referenced by a `<chapter>` element.

This section contains information about how to position the start of the chapters on an odd folio number. Some of the CSS rules given here as examples are already listed in: [INSTALLATION_DIRECTORY]/css/print/p-optional-double-side-pagination.css. You may choose to import this file from your customization CSS (on page 1814).

How to Start Chapters on Odd Pages

A common use case is to arrange the chapters of the publication to start on an odd page number.

In your customization CSS (on page 1814), add the following:

```css
@page chapter {
  -oxy-initial-page-number: auto-odd;
}
@page table-of-contents {
  -oxy-initial-page-number: auto-odd;
}
```

Supported values for `-oxy-initial-page-number` include: `auto`, `auto-even`, `auto-odd`, or a number.

How to Style the Empty (Blank) Pages

By making the chapters start on an odd page, the CSS processor might add blank pages to the previous page sequence as padding.

To style those blank pages add the following code in your customization CSS (on page 1814):

```css
@page chapter:blank, table-of-contents:blank {
  @top-left      { content: none; }
  @top-center    { content: none; }
  @top-right     { content: none; }
  @bottom-left   { content: none; }
  @bottom-center { content: none; }
  @bottom-right  { content: none; }
}
```

Note:
This just removes the headers and footers, but you can use a background image or a header with "Intentionally left blank" text.
How to Force an Odd or Even Number of Pages in a Chapter

Another use case is to specify a number of pages for a section. Suppose that you have a table of contents that follows the cover page and you need to have an even number of pages. Hence, the next chapter would start on an even page.

In your customization CSS (on page 1814), use the -oxy-force-page-count property with an even value:

```css
@page table-of-contents {
    -oxy-force-page-count: even;
}
```

Supported values for -oxy-force-page-count include: even, odd, end-on-even, end-on-odd, auto, no-force.

How to Style the First page of a Chapter

You can use the :first page rule selector to control how the first page of a chapter looks. Suppose that you have defined the following layout for your default page and you want to put the publication title (the maptile string) on the header of the first page (instead of the chapter name that is displayed on this page):

In your customization CSS (on page 1814), add the following:

```css
@page chapter:first {
    @top-right-corner { content: string(maptitle); } 
    @top-left { content: none; }
}
```

Multiple Column Pages

This section contains information about how to handle pages that have multiple columns.

How to Use a Two Column Layout

Change Layout for Predefined Pages.

First you need to identify which of the pages need to be changed. Pages are already defined for the cover page, table of contents, chapter content, and others. The complete list is here: Default Page Definitions (on page 1821).

Next, add the column-count and column-gap properties to that page. For example:

```css
@page chapter{
    column-count:2;
}
```
| If you need some of the elements to expand on all the columns, use the `column-span:all` CSS property. The next snippet makes the chapter titles span both columns:

```css
*[^class =~ "topic/topic"]*[is-chapter] > *[^class =~ "topic/title"] { 
  column-span:all;
}
```

**Limitation:**

You cannot use multiple column configurations on the same page. Oxygen XML Editor only takes the `column-count` and `column-gap` properties into account if they are set on `@page` rules, not on elements from the content.

---

**Change Layout for a Specific Topic.**

If you need to have a different column layout just for one topic, you can use the following technique:

1. Define an outputclass on the topic root element.

```xml
<topic outputclass="two_columns" ... 
```

2. Define a CSS rule that changes the `page` property for the matching element.

```css
*[^class =~ "two_columns"],
[^outputclass =~ "two_columns"] { 
  page: two_column_page !important;
}
```

**Tip:**

In the selector, use the `class` attribute for the HTML transformation, or `outputclass` for the direct transformation, or leave them both if you are not sure.

**Note:**

The topics from the first level use the `chapter` page. You must use `!important` because the built-in rules are more specific and you need to override the `page` property.

3. Define a page layout.

```css
@page two_column_page { 
  column-count: 2;
}
```

Note that the topic will be separated from other sibling topics with different page layouts by page breaks.
Related Information:
Page Formatting in Oxygen PDF Chemistry

PDF Bookmarks

The PDF Bookmarks are used to generate a hierarchical structure similar to a table of contents in a specialized view of your PDF Reader.

By default, the titles defined in the topics are used as bookmark labels.

PDF Bookmarks - Built-in CSS

The PDF bookmarks are generated by matching the titles from the topics in the content. The built-in CSS rules are in: \[\text{PLUGIN}\_\text{DIR}]\text{/css/print/p-bookmarks.css}.\n
How to Change the Bookmark Labels using the Navigation Title

To change the bookmark labels, you can specify a navigation title in a DITA map or topic.

This will be used as the bookmark label instead of the topic title in the table of contents and the bookmark views. There are two possibilities to do specify it:

1. Place a \texttt{<navtitle>} element in the topic reference in the DITA map:

```xml
...  
<topicref href="topics/my_topic.dita" locktitle="yes">  
  <topicmeta>  
    <navtitle>Introduction</navtitle>  
  </topicmeta>  
</topicref>  
...  
```

\textbf{Note:}
As a best practice, a \texttt{@locktitle} attribute with the value ‘yes’ is needed to activate the navigation title. The plugin applies the navigation title even if the attribute is missing.

2. Place a \texttt{<navtitle>} element in the topic, as a title alternative.

```xml
<topic id="other_topic" xml:lang="en-us">  
  <title>Normal Title</title>  
  <titlealts>  
    <navtitle>Navigation Title</navtitle>  
  </titlealts>  
</topic>  
...  
```
How to Control Bookmarks Depth and Sections Display in PDF.

By default, the PDF bookmarks are generated for up to 7 levels. If you need to limit them (for example, to 2 levels), you can use the following CSS rules in your customization CSS (on page 1814):

```
*[^class~="topic/topic"] *[^class~="topic/topic"] *[^class~="topic/topic"] > *[^class~="topic/title"],
*[^class~="topic/topic"] *[^class~="topic/topic"] *[^class~="topic/topic"] *[^class~="topic/topic"] >
  *[^class~="topic/title"],
*[^class~="topic/topic"] *[^class~="topic/topic"] *[^class~="topic/topic"] *[^class~="topic/topic"]
  *[^class~="topic/topic"] *[^class~="topic/topic"] *[^class~="topic/topic"] > *[^class~="topic/title"],
*[^class~="topic/topic"] *[^class~="topic/topic"] *[^class~="topic/topic"] *[^class~="topic/topic"]
  *[^class~="topic/topic"] *[^class~="topic/topic"] *[^class~="topic/topic"] > *[^class~="topic/title"],
*[^class~="topic/topic"] *[^class~="topic/topic"] *[^class~="topic/topic"] *[^class~="topic/topic"]
  *[^class~="topic/topic"] *[^class~="topic/topic"] *[^class~="topic/topic"] > *[^class~="topic/title"],
*[^class~="topic/topic"] *[^class~="topic/topic"] *[^class~="topic/topic"] *[^class~="topic/topic"]
  *[^class~="topic/topic"] *[^class~="topic/topic"] *[^class~="topic/topic"] > *[^class~="topic/title"],
*[^class~="topic/topic"] *[^class~="topic/topic"] *[^class~="topic/topic"] *[^class~="topic/topic"]
  *[^class~="topic/topic"] *[^class~="topic/topic"] *[^class~="topic/topic"] > *[^class~="topic/title"],
*[^class~="topic/topic"] *[^class~="topic/topic"] *[^class~="topic/topic"] *[^class~="topic/topic"]
  *[^class~="topic/topic"] *[^class~="topic/topic"] *[^class~="topic/topic"] > *[^class~="topic/title"],
*[^class~="topic/topic"] *[^class~="topic/topic"] *[^class~="topic/topic"] *[^class~="topic/topic"]
  *[^class~="topic/topic"] *[^class~="topic/topic"] *[^class~="topic/topic"] > *[^class~="topic/title"],
*[^class~="topic/topic"] *[^class~="topic/topic"] *[^class~="topic/topic"] *[^class~="topic/topic"]
  *[^class~="topic/topic"] *[^class~="topic/topic"] *[^class~="topic/topic"] > *[^class~="topic/title"],
*[^class~="topic/topic"] *[^class~="topic/topic"] *[^class~="topic/topic"] *[^class~="topic/topic"]
  *[^class~="topic/topic"] *[^class~="topic/topic"] *[^class~="topic/topic"] > *[^class~="topic/title"],
*[^class~="topic/topic"] *[^class~="topic/topic"] *[^class~="topic/topic"] *[^class~="topic/topic"]
  *[^class~="topic/topic"] *[^class~="topic/topic"] *[^class~="topic/title"]
  { bookmark-label: none; }
```

These rules clear the labels generated by the titles starting with the depth of 3 (the topic nesting level is given by the selectors *[^class~="topic/topic"]*).

By default, the PDF bookmarks also include the sections. If you need to remove them, you can use the following CSS rule in your customization CSS (on page 1814):

```
*[^class~="topic/topic"] *[^class~="topic/section"] > *[^class~="topic/title"],
*[^class~="topic/topic"] *[^class~="topic/topic"] *[^class~="topic/section"] > *[^class~="topic/title"],
*[^class~="topic/topic"] *[^class~="topic/section"] *[^class~="topic/section"] > *[^class~="topic/title"],
*[^class~="topic/topic"] *[^class~="topic/topic"] *[^class~="topic/section"] *[^class~="topic/section"]
  *[^class~="topic/title"]
  { bookmark-label: none; }
```

How to Specify the Open/Closed PDF Bookmark State

If you want to specify the initial state for the bookmarks (opened/expanded or closed/collapsed), you can use the `bookmark-state` property in your customization CSS (on page 1814).
For example, to specify that all bookmarks for the first three levels are opened (expanded) in the initial state, use:

```css
*{class~="topic/topic"} > *{class~="topic/title"},
*{class~="topic/topic"} *{class~="topic/topic"} > *{class~="topic/title"},
*{class~="topic/topic"} *{class~="topic/topic"} *{class~="topic/topic"} > *{class~="topic/title"} {
  bookmark-state:open;
}
```

**How to Remove the Numbering From the PDF Bookmarks**

By default, the PDF bookmark labels are generated while taking the text set before the chapters titles into account. Since this usually contains the part, chapter, or section numbers, the PDF Bookmarks will make use of them.

The solution is to remove the `content(before)` from the `bookmark-label`, leaving just the `content(text)`.

In your customization CSS *(on page 1814)*, add the following CSS rules:

```css
*{class~="topic/topic"} > *{class~="topic/title"} {
  bookmark-label: content(text);
  -ah-bookmark-label: content();
}
```

**Important:**

This is a simple example that does not use the possible navigation titles, just the content of the `<title>` element. Copy and modify the built-in CSS for the full CSS rule that matches the `<title>` and `<titlealts>` elements:

```css
*{class~="topic/topic"}:has(*{class~="topic/titlealts"}) > *{class~="topic/title"} {...}
```

**Related Information:**

Numbering *(on page 1879)*

**Index**

The content of an `<indexterm>` element is used to produce an index entry in the generated index. You can nest `<indexterm>` elements to create multi-level indexes. The content is not output as part of the topic content, only as part of the index tree.

To add an index to your publication, you just need to add `<indexterm>` elements inside the `<prolog>` section (inside a `<metadata>` element):

```xml
<title>The topic title.</title>

<prolog>
  <metadata>
    ...
  </metadata>
```
or in the content itself:

```xml
...  
<p>Open the lid then turn the body pump to the right.  
</p>
...  

If you are using a bookmap, you need to specify where the index list should be presented (for instance in the <i>backmatter</i> of the book. Technically, it is possible to also add it to the <i>frontmatter</i>, but this is unusual). This is done using an <indexlist/> element in the <booklists/> element (inside the <backmatter/>):

```xml
<bookmap>

...</p>Open the lid then turn the body pump to the right.<p>
</bookmap>
```

For plain maps, the index list is automatically added at the end of the publication, with no need to modify the map.

**Index - XML Fragment**

In the merged map file (on page 1816), the structure that holds the index tree is the <opentopic:index:index.groups/> element.

```xml
<map class="- map/map ">

<oxy:front-page>

...</oxy:front-page>

<opentopic:map xmlns:opentopic="http://www.idiominc.com/opentopic">
```
Each of the groups contain:

- A label, the starting letter ("T" in the following example).
- A tree of `<opentopic-index:index.entry>` elements.
Each of the entries contain:

- The formatted value (opc:formatted-value).
- A link to the publication content (opc:refID/ox:index-link).
- Possibly other child entries.

For the DITA Map PDF - based on HTML5 & CSS transformation type, the merged map is further processed resulting in a collection of HTML5 <div> elements. These elements preserve the original DITA class attribute values and add a new value derived from the DITA element name.

```
<map>
  ...
  <div class="- topic/topic topic">
    <div class="- topic/title title">Request Support</div>
    ...
  </div>
  ...
</map>
```

The index group content becomes:

```
<map>
  ...
  <div class="index/group group">
    ...
    <div class="index/entry entry">
      <div class="index/formatted-value formatted-value">Table of contents</div>
      <div class="index/refid refid">
        <div class="index/link link" href="#d16e3988">[d16e3988]</div>
      </div>
    </div>
    <div class="index/entry entry">
      <div class="index/formatted-value formatted-value">change header</div>
      <div class="index/refid refid">
        <div class="index/link link" href="#d16e3988">[d16e3988]</div>
      </div>
    </div>
  </div>
</map>
```
Index - Built-in CSS

All index styling is found in: [PLUGIN_DIR]css/print/p-index.css.

How to Style the Index Page Title and the Grouping Letters

In your customization CSS (on page 1814), add the following CSS rules:

```
* [class =~ "index/groups"] * [class =~ "index/group"] * [class =~ "index/label"] {
  font-size: 1.5em;
  color: navy;
}

* [class =~ "index/groups"]:before {
  content: "- Index - ";
  color: navy;
  font-size: 4em;
}
```

The result is:
How to Style the Index Terms Labels

In your customization CSS (on page 1814), add the following CSS rule:

```css
* [class ~='index/groups'] * [class ~='index/formatted-value'] {
    font-style: oblique;
    color: gray;
}
```

The result is:

```
Index
```

How to Add Filling Dots Between the Index Labels and the Page Numbers

Suppose you want the leader CSS content to generate a row of dots. It is necessary that the parent entry has the text justified.
In your customization CSS (on page 1814), add the following CSS rule:

```css
/* Hide the sequences of links that actually do not contain links. */
*[class="index/group"] *[class="index/entry"] > *[class="index/refid"]{
  display:none;
}

/* Hide the sequences of links that actually do not contain links. */
*[class="index/group"] *[class="index/entry"] > *
 *[class="index/refid"]:has(*[class="index/link"]){
  display:inline;
}

*[class="index/group"] *[class="index/entry"] {
  text-align:justify;
}

*[class="index/group"] *[class="index/entry"] > *
 *[class="index/refid"]:before{
  content:leader('.');  
}
```

The output now contains the dots:

```
Index

F
  footer................................................. 37

H
  header................................................ 37

T
  table of contents.................................... 32
    change header..................................... 32
    style............................................. 34
```
How to Change the Index Page Number Format and Reset its Value

The page number is reset at the beginning of the index page by the built-in CSS rule:

```css
*[class ~= "index/groups"] { 
    counter-reset: page 1;
}
```

If you want to start the page counter from a different initial number, just change the value of this counter. For example, to continue the normal page counting, use:

```css
*[class ~= "index/groups"] { 
    counter-reset: none;
}
```

If you need to style the page number differently (for example, using decimals), add the following CSS rule in your customization CSS (on page 1814):

```css
@page index {
    @bottom-center { content: counter(page, decimal) }
}
```

How to Impose a Table-like Index Layout

In case you need to place the index labels and links on the same line but with some extra alignment constraints, you can use inline blocks to give the index a table-like appearance:

```
Index

<table>
<thead>
<tr>
<th>C</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Close programs</td>
<td>8</td>
</tr>
<tr>
<td>Computer cover</td>
<td>7,8</td>
</tr>
<tr>
<td>configure</td>
<td>10, 10,10</td>
</tr>
<tr>
<td>D</td>
<td></td>
</tr>
<tr>
<td>database</td>
<td>8, 8, 8, 9, 10, 11,12</td>
</tr>
<tr>
<td>H</td>
<td></td>
</tr>
<tr>
<td>hard drive</td>
<td>7, 7, 7, 8, 10, 11,12</td>
</tr>
<tr>
<td>I</td>
<td></td>
</tr>
<tr>
<td>install</td>
<td>7, 8,9</td>
</tr>
<tr>
<td>M</td>
<td></td>
</tr>
<tr>
<td>maintain</td>
<td>11,11</td>
</tr>
<tr>
<td>hdd</td>
<td></td>
</tr>
<tr>
<td>drive</td>
<td>11</td>
</tr>
</tbody>
</table>
```
You need to place the elements that have the following class on the same line:

**indexformatted-value**

This is the text of the index term.

**index/refid**

This element contains a list of links.

A fixed width is used for the formatted value and the links container (almost half of the available width). To achieve the index hierarchical layout, set progressive padding to the formatted value text.

In your customization CSS *(on page 1814)*, add the following CSS rule:

```css
/* Hide the sequences of links that actually do not contain links. */
*[class="index/groups"] *[class="index/entry"] > *[class="index/refid"] {
  display: none;
}

/* Move the nesting of indexterms from margin to padding */
*[class="index/groups"] *[class="index/entry"] {
  margin-left: 0;
}

*[class="index/groups"]
*[class="index/entry"]
*[class="index/formatted-value"] {
  padding-left: 0.2em;
}
```

/* Hide the sequences of links that actually do not contain links. */

```css
*[class="index/groups"] *[class="index/entry"] > *[class="index/refid"] {
  display: none;
}

/* Move the nesting of indexterms from margin to padding */
*[class="index/groups"] *[class="index/entry"] {
  margin-left: 0;
}

*[class="index/groups"]
*[class="index/entry"]
*[class="index/formatted-value"] {
  padding-left: 0.2em;
}
```

/* Hide the sequences of links that actually do not contain links. */

```css
*[class="index/groups"] *[class="index/entry"] > *[class="index/refid"] {
  display: none;
}

/* Move the nesting of indexterms from margin to padding */
*[class="index/groups"] *[class="index/entry"] {
  margin-left: 0;
}

*[class="index/groups"]
*[class="index/entry"]
*[class="index/formatted-value"] {
  padding-left: 0.2em;
}
```

/* Hide the sequences of links that actually do not contain links. */

```css
*[class="index/groups"] *[class="index/entry"] > *[class="index/refid"] {
  display: none;
}

/* Move the nesting of indexterms from margin to padding */
*[class="index/groups"] *[class="index/entry"] {
  margin-left: 0;
}

*[class="index/groups"]
*[class="index/entry"]
*[class="index/formatted-value"] {
  padding-left: 0.2em;
}
```

/* Hide the sequences of links that actually do not contain links. */

```css
*[class="index/groups"] *[class="index/entry"] > *[class="index/refid"] {
  display: none;
}

/* Move the nesting of indexterms from margin to padding */
*[class="index/groups"] *[class="index/entry"] {
  margin-left: 0;
}

*[class="index/groups"]
*[class="index/entry"]
*[class="index/formatted-value"] {
  padding-left: 0.2em;
}
```

/* Hide the sequences of links that actually do not contain links. */

```css
*[class="index/groups"] *[class="index/entry"] > *[class="index/refid"] {
  display: none;
}

/* Move the nesting of indexterms from margin to padding */
*[class="index/groups"] *[class="index/entry"] {
  margin-left: 0;
}

*[class="index/groups"]
*[class="index/entry"]
*[class="index/formatted-value"] {
  padding-left: 0.2em;
}
```
To avoid bleeding of the index term label, you may need to mark it as being hyphenated:

```xml
*[class="index/formatted-value"] {  
  hyphens:auto;
}
```

To activate hyphenation, see: How to Enable Hyphenation for Entire Map (on page 1926).

**Appendices**

The `<appendices>` element that is available in the DITA bookmap has a special behavior (based on its sibling nodes):

1. If the bookmap contains `<part>` elements, the `<appendices>` will behave as a part.
2. If the bookmap contains `<chapter>` (and no `<part>`) elements, the `<appendices>` will behave as a chapter.

**Note:**

The behavior includes page-break, numbering, and title rendering.

For example, if I define a bookmap with a `<part>` element, I will obtain:

```xml
<part>
  <chapter/>
  <topicref/>
  <chapter/>
```
For another example, if I define a bookmap with a `<chapter>` element only, I will obtain:

```
<chapter/>
<topicref/>
<chapter/>
<appendices> <!-- Appendices behaves like a Chapter -->
  <appendix/> <!-- Appendix behaves like a TopicRef -->
  <appendix/>
</appendices>
```

Warning: If the `<appendices>` element is not defined and the `<appendix>` is used directly instead, then it will behave like a Part or Chapter using the same pattern as for `<appendices>`.

How To Control Page Break Within Appendices

If you define a bookmap with `<appendices>` and some `<appendix>` elements, you may want the parent `<appendices>` to be on a separate page than its children. This is done automatically if the bookmap contains `<part>` elements. Otherwise, you may need to use the following in your CSS:

```
*[topicrefclass ~="bookmap/appendix"]:first-of-type {
  page-break-before: always;
}
```

Footnotes

Footnotes are pieces of information placed at the bottom of the page. A footnote has a number that is left in the content (the footnote call) and the same value placed before the footnote text (the footnote marker).

Footnotes - Built-in CSS

Footnote properties are defined in `[PLUGIN_DIR]css/print/p-foot-notes.css`.

How to Change Style of the Footnote Markers and Footnote Calls

To bold the footnotes numbers and use some colors, add the following rules to your customization CSS (on page 1814):

```
*[class ~="topic/fn"]:footnote-call {
  font-weight: bold;
}
```
How to Add a Separator Above the Footnotes

The `@footnote` part of a `@page` declaration controls the style of the separator between the page content and the footnotes. For the content, you should set a `leader`. The leader uses a letter or a line style to fill the entire width of the page.

```xml
@page {
    margin:0.5in;
    ...
    @footnote {
        content: leader(solid);
        color:silver;
    }
}
```

To create a dotted line, you can use the dot character: `leader('.')`. Other commonly used characters are: 
"-" (dash) and "." (underscore).

How to Reset the Footnotes Counter

There are cases where you want to reset the footnote counter.

For example, if you need to reset it at the beginning of each chapter, add the following rules to your customization CSS *(on page 1814)*:

```xml
*[class ~="bookmap/chapter"],
*[class ~="topic/topic"]{is-chapter} {
    counter-reset: footnote 1;
}
```

Or you can mark any element with an `@outputclass` value, match that value, and reset the counter at any point in your counter:

```xml
<p outputclass="reset-footnotes"/>
```
How to Display Footnotes Below Tables

In your PDF output, you may want to group all the footnotes contained in a table just below it instead of having them displayed at the bottom of the page.

To add this functionality, use an Oxygen Publishing Template and follow these steps:

1. If you have not already created a Publishing Template, you need to create one. For details, see How to Create a Publishing Template (on page 1791).
2. Link the folder associated with the publishing template to your current project in the Project view.
3. Using the Project view, create an xslt folder inside the project root folder.
4. In the newly created folder, create an XSL file (for example, named merged2mergedExtension.xsl) with the following content:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    xmlns:xs="http://www.w3.org/2001/XMLSchema"
    exclude-result-prefixes="xs"
    version="2.0">

<!-- Match only top level tables (i.e tables that are not nested in other tables), that have footnotes. -->
<xsl:template match="
    *[contains(@class, 'topic/table')]
    [not(ancestor::*[contains(@class, 'topic/table')])]
    [/]*(contains(@class, 'topic/fn'))">
    <xsl:next-match>
      <xsl:with-param name="top-level-table" select="/" tunnel="yes"/>
    </xsl:next-match>

<!-- Create a list with all the footnotes from the current table. -->
<ol class="- topic/ol " outputclass="table-fn-container">
  <xsl:for-each select="/*[contains(@class, 'topic/fn')]">
    <li class="- topic/li " id="{generate-id(.)}" outputclass="table-fn">
      <xsl:apply-templates select="node()"/>
    </li>
  </xsl:for-each>
</ol>
</xsl:template>
```
5. Open the template descriptor file (on page 1785) associated with your publishing template (the .opt file) and set the XSLT stylesheet created in the previous step with the `com.oxygenxml.pdf.css.xsl.merged2merged` XSLT extension point:

```xml
<publishing-template>
  ...
  <pdf>
    ...
    <xslt>
      <extension
        id="com.oxygenxml.pdf.css.xsl.merged2merged"/>
      <file>"xslt/merged2mergedExtension.xsl"
    </xslt>
  </pdf>
</publishing-template>
```

6. Create a css folder in the publishing template directory. In this directory, save a custom CSS file with rules that style the glossary structure. For example:

```css
/* Customize footnote calls, inside the table. */
*[outputclass = "table-fn-call"] {
  color: black;
  text-decoration: none;
  font-size: smaller;
  vertical-align: super;
  line-height: none;
}

/* Hide the "on page .." text that follows xrefs. */
*[outputclass = "table-fn-call"]:after {
  display: none;
}
```
/* Customize the list containing all the table footnotes. */
*[outputclass ~='table-fn-container'] {
  border-top: 1pt solid black;
}

/* Customize footnotes display, below the table. */
*[outputclass ~='table-fn'] {
  font-size: 1em;
}
*[outputclass ~='table-fn']::marker {
  font-size: 1em;
}

7. Open the template descriptor file (on page 1785) associated with your publishing template (the .opt file) and reference your custom CSS file in the resources element:

    <publishing-template>
        ...
        <pdf>
            ...
            <resources>
                ...
                <css file="css/custom.css"/>
            </resources>
        </pdf>
    </publishing-template>

8. Edit the DITA Map PDF - based on HTML5 & CSS transformation scenario.
9. In the Templates tab, click the Choose Custom Publishing Template link and select your template.
10. Click OK to save the changes and run the transformation scenario.

Hyphenation

Hyphenation specifies how words should be hyphenated when text wraps across multiple lines.

The transformation plugin uses the capabilities of the PDF Chemistry processor to perform hyphenation.

Hyphenation Dictionaries

The Oxygen XML Editor provides built-in hyphenation patterns for the following languages:

<table>
<thead>
<tr>
<th>Code</th>
<th>Language</th>
</tr>
</thead>
<tbody>
<tr>
<td>da</td>
<td>Danish</td>
</tr>
<tr>
<td>de</td>
<td>German</td>
</tr>
<tr>
<td>de_CH</td>
<td>German (Switzerland)</td>
</tr>
<tr>
<td>en</td>
<td>English</td>
</tr>
<tr>
<td>en-GB</td>
<td>English (Great Britain)</td>
</tr>
</tbody>
</table>
The built-in hyphenation pattern license terms are listed in the XML files in the 
[CHEMISTRY_INSTALL_DIR]/config/hyph folder. Most of them comply with the LaTex distribution policy.

### Installing New Hyphenation Dictionaries

Oxygen XML Editor uses the TeX hyphenation dictionaries converted to XML by the OFFO project: [https://sourceforge.net/projects/offo/](https://sourceforge.net/projects/offo/).

The .xml files allow you to access the licensing terms and you can use them as a starting point to create customized dictionaries (see How to Alter a Hyphenation Dictionary (on page 1925)).

The .hyp files are the compiled dictionaries that the Oxygen XML Editor actually uses.

One simple way to add more dictionaries:

1. Download and extract the offo-hyphenation-compiled.zip file. This file is a bundle of many dictionary files.
2. Copy the fop-hyph.jar file to the [OXYGEN_INSTALL_DIR]/lib directory.
3. If you just need a single dictionary, place the .hyp or .xml file in the [OXYGEN_INSTALL_DIR]/config/hyph directory (create that directory if it is missing).

### How to Alter a Hyphenation Dictionary

You can copy the dictionaries you need to change in another directory, then use the -hyph-dir parameter to refer them inside your transformation.

Each file is named with the language code and has the following structure:

<table>
<thead>
<tr>
<th>Code</th>
<th>Language</th>
</tr>
</thead>
<tbody>
<tr>
<td>es</td>
<td>Spanish</td>
</tr>
<tr>
<td>fr</td>
<td>French</td>
</tr>
<tr>
<td>it</td>
<td>Italian</td>
</tr>
<tr>
<td>nb</td>
<td>Norwegian Bokmål</td>
</tr>
<tr>
<td>nl</td>
<td>Dutch</td>
</tr>
<tr>
<td>ro</td>
<td>Romanian</td>
</tr>
<tr>
<td>ru</td>
<td>Russian</td>
</tr>
<tr>
<td>sv</td>
<td>Swedish</td>
</tr>
<tr>
<td>th</td>
<td>Thai</td>
</tr>
<tr>
<td>pt</td>
<td>Portuguese</td>
</tr>
<tr>
<td>da</td>
<td>Danish</td>
</tr>
</tbody>
</table>
<hyphenation-info>

<hyphen-min before="2" after="3"/>

<exceptions>
o-mni-bus
...
</exceptions>

<patterns>
préémi3nent.
proémi3nent.
surémi3nent.
....
</patterns>

</hyphenation-info>

To change the behavior of the hyphenation, you can modify either the patterns or the exceptions sections:

exceptions

Contains the list of words that are not processed using the patterns, each on a single line. Each of the words should indicate the hyphenation points using the hyphen ("-"), character. If a word does not contain this character, it will not be hyphenated.

For example, o-mni-bus will match the omnibus word and will indicate two possible hyphenation points.

Note:
Compound words (i.e. e-mail) cannot be controlled by exception words.

patterns

Contains the list of patterns, each on a single line. A pattern is a word fragment, not a word. The numbers from the patterns indicate how desirable a hyphen is at that position.

For example, tran3s2act indicates that the possible hyphenation points are "tran-s-act" and the preferable point is the first one, having the higher score of "3".

How to Enable Hyphenation for Entire Map

To enable hyphenation for your entire map:
1. Make sure you set an `@xml:lang` attribute on the root of your map, or set the `default.language` parameter in the transformation.

2. In your customization CSS (on page 1814), add:

```css
:root {
  hyphens: auto;
}
```

3. To except certain elements from being hyphenated, use `hyphens:none`. The following example excludes the `<keyword>` elements from being hyphenated:

```css
/*[class ~="topic/keyword"] {
  hyphens: none;
}
```

### How to Enable/Disable Hyphenation for an Element

1. Make sure you set an `@xml:lang` attribute on the root of your map, or set the `default.language` parameter in the transformation.

2. You have two options to control hyphenation inside an XML element:

   **CSS Approach**
   
   Use the `hyphens` property.
   
   For example, if you want to enable hyphenation in codeblocks:
   
   ```xml
   /*[class="pr-d/codeblock"] {
     hyphens: auto;
   }
   ```
   
   If you want to disable hyphenation inside tables:
   
   ```xml
   /*[class="topic/table"] {
     hyphens: none;
   }
   ```

   **Attribute Approach**
   
   Use the `@outputclass="hyphens"` or `@outputclass="no-hyphens"` attributes/values.
   
   For example, if you want to enable hyphenation in codeblocks:
   
   ```xml
   <codeblock outputclass="hyphens">
     ...
   </codeblock>
   ```
   
   If you want to disable hyphenation inside tables:
Note:
The default built-in CSS enables hyphenation for tables:

```css
*[^class ~="topic/table"][] {
    hyphens: auto;
}
```

Related information
How to Disable Hyphenation for a Word

**Accessibility**

By default, the PDF documents produced using this plugin are partially accessible in the sense that most of the paragraphs, tables, lists, headers, and footers are tagged automatically so a PDF reader can use this information to present the content.

Related Information:
Oxygen PDF Chemistry: Accessibility

**Accessibility - Built-in CSS**

Accessibility properties are defined in `[PLUGIN_DIR]css/print/p-accessibility.css`.

**How to Create Fully Accessible Documents**

To make your documents fully accessible (PDF/UA1 compliant), do the following:

1. The accessibility standard requires that all the fonts be embedded in the PDF. To force font embedding, you have to specify fonts for all elements and for all page margin boxes in your customization CSS (on page 1814). For instance, you can use:

```css
body { font-family: Arial } 
@page {
    @top-left {font-family: Arial }
    @top-right {font-family: Arial }
    @top-center {font-family: Arial }
    @top-left-corner {font-family: Arial }
    @top-right-corner {font-family: Arial }
```
2. Create a new transformation scenario, based on the DITA Map PDF - based on HTML5 & CSS built-in scenario.

3. In the Parameters tab, change the value of the pdf.accessibility parameter to yes.

4. Run the transformation.

Fonts

Fonts are an important part of the publication. Your font selection should take into consideration both design and the targeted ranges of characters.

To use them in the customization CSS (on page 1814):

- You can place the font files in the same folder as your CSS and use a @font-face definition to reference them.
- You can use web fonts (for example, Google Fonts), and import the CSS snippet into your CSS.
- You can use system fonts.

All these techniques are explained in: Oxygen PDF Chemistry User Manual: Fonts.

How to Avoid Characters Being Rendered as 

When the processor renders text with a font that does not include certain characters, those characters are replaced with the # symbol. This can easily be seen from Oxygen's Results view. For example:

```
[CH] Glyph "??" (0x7ae0) not available in font "Roboto-Regular".
```

To prevent this, make sure you use the proper font.

As an example, suppose the right arrow character is used in a definition list like this:

```
<dlentry>
  <dt>&#8594;</dt>
  <dd><ph>This is the right arrow.</ph></dd>
</dlentry>
```

If the font does not include this character, the output will look something like this:

```
#
  This is the right arrow.
```
To fix this you can either:

1. Install Arial Unicode MS font on your system. This is one of the PDF processor fallback fonts. Starting with version 24 there are additional fonts used as fallback as SimSun, Malgun Gothic and more, so if on of these are found on the system they will be used directly.

2. Specify the fallback font in your customization.

For the second case, if you use *Times New Roman* for the entire publication, you could add *Symbol* as the fallback font. In your customization CSS *(on page 1814)*, add:

```css
*{class =~ "topic/dlentry"} {
  font-family: "Times New Roman", Symbol;
}
```

To change the font for the entire publication, not just an element, you can use:

```css
:root {font-family: "Times New Roman", Symbol !important; }
@page {
  @top-left {font-family: "Times New Roman", Symbol !important; }
  @top-right {font-family: "Times New Roman", Symbol !important; }
  @top-center {font-family: "Times New Roman", Symbol !important; }
  @top-left-corner {font-family: "Times New Roman", Symbol !important; }
  @top-right-corner {font-family: "Times New Roman", Symbol !important; }
  @bottom-left {font-family: "Times New Roman", Symbol !important; }
  @bottom-right {font-family: "Times New Roman", Symbol !important; }
  @bottom-center {font-family: "Times New Roman", Symbol !important; }
  @bottom-left-corner {font-family: "Times New Roman", Symbol !important; }
  @bottom-right-corner {font-family: "Times New Roman", Symbol !important; }
}
```

**Tip:**
On Windows, one simple way to determine the font needed to display the text is to copy the text fragment that has rendering problems from the DITA source document and paste it into Microsoft WordPad or Word. It will automatically select a font capable of rendering the text. Simply click on the text to see the name of the font from the "Font" ribbon toolbar. Then you can use it as a fallback font in the CSS. Make sure there are no licensing restrictions on that particular font.

**Note:**
It is also possible to use a generic family name as fallback, like *serif*, *sans-serif* or *monospace*, like this you will call upon the processor default fallback fonts system.
Warning:
Even if the message is a warning, sometimes it can lead to a failed transformation. This usually occurs for really special characters (different from letters or common characters).

How to Set Fonts in Titles and Content

Suppose that in your customization CSS (on page 1814), you have defined your font (for example, Roboto) using a Google web font:

```html
@import url('https://fonts.googleapis.com/css?family=Roboto');
```

You can force a font on all elements, then style the ones that need to be different. The advantage of this method is that you do not need to trace all elements that have a font family defined in the built-in CSS files, you just reset them all.

In your customization CSS (on page 1814), add an !important rule that associates a font to all the elements from the document:

```css
* {
    font-family: "Roboto" !important;
}
```

Note:
If you want to use the :root selector instead of the * selector, without the !important qualifier, the elements that have a predefined font specified in the built-in CSS will keep that font. If your content uses non-Latin glyphs, it is possible that the built-in fonts do not render them.

Important:
These settings do not apply to page margin boxes, only to the text inside the page. If you also want to change the font used in headers and footers, see: How to Change the Font of the Headers and Footers (on page 1831).

Next, identify the selectors for the elements that need to be styled with a different font than the one associated above. For information on how to do this, see: Debugging the CSS (on page 1816).

For example, if you want the titles or the pre-formatted text to have a different font from the rest, matched by the above * selector, you need to use more specific CSS selectors:

```css
*[class="front-page/front-page-title"],
*[class="topic/title"] {
    font-family: Arial !important;
}
*[class="pr-d/codeph"],
```
Related information

How to Change the Font of the Headers and Footers (on page 1831)

How to Use Fonts for Asian Languages

For Asian languages, you must use a font or a sequence of fonts that cover the needed character ranges. If the characters are not found, the # symbol is used.

When you specify a sequence of fonts, if the glyphs are not found in the first font, the next font is selected, and so on until one is found that includes all the glyphs. A common font sequence for Asian languages is as follows:

```xml
<font-family>Calibri, SimSun, "Malgun Gothic", "Microsoft JhengHei";</font-family>
```

To apply this font sequence, see: How to Set Fonts in Titles and Content (on page 1931).

Some of the Asian fonts do not have italic, bold, or bold-italic variants. In this case, you may use the regular font file with multiple font face definitions to simulate (synthesize) the missing variants. You need to use the -oxy-simulate-style:yes CSS property in the font face definition as explained in: Using Simulated/Synthetic Styles in Oxygen Chemistry.

How to Use Asian Fonts in Linux

For Asian languages on Linux distributions, PDF Chemistry automatically uses DejaVu and Noto CJK as fallback fonts for Serif, Sans-Serif, and Monospace content.

**Warning:**

On some distributions, the Noto CJK fonts are not available. In this case, you need to install them using the system package manager:

- fonts-noto-cjk on Debian family distributions (e.g. Ubuntu).
- google-noto-cjk-fonts on Red Hat family distributions (e.g. CentOS).

How to Add a New Asian Font

If you want to add a specific font for Asian languages, you need to declare it inside your customization CSS (on page 1814). The following example uses the Noto Sans Tamil font-family:

```xml
/* Font Declaration */
@font-face {
  font-family: "Noto Sans Tamil";
}
```
How to Set Fonts for Displaying Music

Music is rendered as normal text in most fonts, but some of them will render them as musical glyphs. For example, the MusGlyphs font converts the text to music and adjusts it to the surrounding text.

This font is divided in two sub-fonts that act for each of the following categories:

- **MusGlyphs** - Converts all characters that match a musical pattern into music glyphs. It should be used inside the elements that contain only music.
- **MusGlyphs-Text** - Converts only the text prefixed with the @ symbol into music glyphs. The remaining text is displayed normally.

To use this font, you simply need to declare each sub-font then use them in appropriate elements:
Comments, Highlights, and Tracked Changes

The comments and tracked changes can be made visible in the PDF output by setting the `show.changes.and.comments` transformation parameter to yes.

Figure 511. Chemistry Annotations in Acrobat Reader

By default, they are shown as PDF text annotations (sticky notes). These are graphical markers in the document content and are also listed in the Comments section when opening the output file in Acrobat Reader.

Note:

Comments with the Mark as Done flag selected appear with a check mark in the Comments section and with a Completed label (John Doe Completed).
To avoid rendering the elements as PDF annotations and show them as footnotes instead, you can use the `show.changes.and.comments.as.pdf.sticky.notes` transformation parameter set to `no`.

The comments and changes are included in the merged map file ([on page 1816](#)) either as XML elements (`<oxy-insert>`, `<oxy-delete>`, `<oxy-comment>`, `<oxy-attributes>`) in the case of the XML merged map, or as HTML elements with similar classes (`oxy-insert`, `oxy-delete`, `oxy-comment`, `oxy-attributes`) in the case of the HTML merged map. Sub-elements contain meta-information about each change.

**Tip:**
These elements are automatically recognized and transformed in PDF annotations when using Chemistry as PDF processor.

**Note:**
The inserted text, deleted text, and deleted markup are included in the sticky notes, you can change this behavior by using the `show.changed.text.in.pdf.sticky.notes.content` parameter ([on page 1779](#)).

Related Information:
Transformation Parameters ([on page 1773](#))
Debugging the CSS ([on page 1816](#))

**Comments and Tracked Changes - XML Fragment**

This section contains information about how each type of tracked change is structured in the merged map XML file ([on page 1816](#)).

**Insertions**

For an insertion type of tracked change, the structure that defines the insertion details is inside the `range` (`<oxy-range-start>` to `<oxy-range-ends>`), the inserted text is highlighted by an `<oxy-insert-hl>` element, and the details are stored in the `<oxy-insert>` element.

```
<oxy:oxy-range-start id="sc_1" hr_id="1"/>

<oxy:oxy-insert href="#sc_1" hr_id="1">
  <oxy:oxy-author>dan</oxy:oxy-author>
  <oxy:oxy-content>insert</oxy:oxy-content>
  <oxy:oxy-date>2018/03/15</oxy:oxy-date>
  <oxy:oxy-hour>09:38:29</oxy:oxy-hour>
  <oxy:oxy-tz>+02:00</oxy:oxy-tz>
</oxy:oxy-insert>

<oxy:oxy-insert-hl>This is an insert!!</oxy:oxy-insert-hl>
```
Comments

Similar to insertions, comments are defined in a range (oxy-range-start to oxy-range-end), the comment details is in the oxy-comment element, and the highlighted content is wrapped in the oxy-comment-hl element.

Note:

Comments that are marked as done have a flag="done" attribute:

Attribute changes

The attribute changes are more complex. The range is empty, and is directly above the affected element (the one that has modified attributes). The oxy-attributes element contains details about multiple attribute changes, each stored in the oxy-attributed-change element.
Deletions

For a deletion, there are some elements that define the start and end of the deletion, and the highlighted text is wrapped in the `<oxy-delete-hl>` element.

```xml
<oxy:oxy-range-start id="sc_2" hr_id="2"/>
<oxy:oxy-delete-hl>This is a deleted text.</oxy:oxy-delete-hl>
<oxy:oxy-range-end hr_id="2"/>
```

There is a structure that offers details about the deletion change, using the `<oxy-delete>` element. This is linked to the above deletion range by the same ID value:

```xml
<oxy:oxy-delete href="#sc_2" hr_id="2">
  <oxy:oxy-author>dan</oxy:oxy-author>
  <oxy:oxy-content><image href="../img/ex.gif"></oxy:oxy-content>
  <oxy:oxy-date>2018/03/14</oxy:oxy-date>
  <oxy:oxy-hour>11:38:06</oxy:oxy-hour>
  <oxy:oxy-tz>+02:00</oxy:tz>
</oxy:oxy-delete>
```

Colored Highlights

To show some text as highlighted with a background color:

```xml
<oxy:oxy-color-hl color="rgba(140,255,140,50)">Some colored text.</oxy:oxy-color-hl>
```

Comments and Tracked Changes - HTML Fragment

This section contains information about how each type of tracked change is structured in the merged map HTML file (on page 1816).

Insertions

For an insertion type of tracked change, the structure that defines the insertion details is inside a range (`oxy-range-start` to `oxy-range-end`), the inserted text is highlighted by a `<span>` element with the class `oxy-insert-hl`, and the details are stored in a `<span>` element with the `oxy-insert` class.
Comments

Similar to insertions, comments are defined in a range (oxy-range-start to oxy-range-end), the comment details in an element with the class oxy-comment, and the highlighted content is wrapped in the oxy-comment-hl element.

Note:
Comments that are marked as done have a flag="done" attribute:

Attribute changes

The attribute changes are more complex. The range is empty, and is directly above the affected element (the one that has modified attributes). The element with the class oxy-attributes contains details about multiple attribute changes, each stored in an element with the class oxy-attribute-change.
Deletions

For a deletion, there are some elements that define the start and end of the deletion, and the highlighted text is wrapped in an element with the class oxy-delete-hl.

There is a structure that offers details about the deletion change, using the element with the class oxy-delete. This is linked to the above deletion range by the same ID value:

Colored Highlights

To show some text as highlighted with a background color:
Comments and Tracked Changes - Built-in CSS

The built-in CSS that controls the way tracked changes and comments are displayed is found in:

```
[PLUGIN_DIR]/css/print/p-side-notes.css
```

How to Style Changed or Commented Text

To style the highlighted text from the document content, use the `<oxy-comment-hl>` element (or `<oxy-delete-hl>`,
`<oxy-insert-hl>`, respectively, by local name or class name):

```
ox+y-comment-hl,  
.oxy-comment-hl {  
  color:magenta;
}
```

**Note:**

The class name selector is useful when using the DITA Map PDF - based on HTML5 & CSS or DITA PDF - based on HTML5 & CSS transformation scenarios.

If you want to change the small labels that define the affected start and end ranges (they are a number formatted like: “[n] .. [/n]” where n is the number of the change), you can use the following selectors:

```
ox+y-range-start:before,  
.oxy-range-start:before {  
  display:inline;  
  content:'[' attr(hr_id) ']';  
  color:red;
}
oxy-range-end:before,  
.oxy-range-end:before {  
  display:inline;  
  content:'[/' attr(hr_id) ']';  
  color:red;
}
```

Use `display:none` if you want to hide them.

If you want to change the background color of the comment footnote (assuming you set the
`show.changes.and.comments.as.pdf.sticky.notes` transformation to `no`), add the following snippet in your
customization CSS (on page 1814):

```
ox+y-comment,  
.oxy-comment {  
  background-color:inherit;
```
Similarly, you can style the other footnotes for `<oxy-attributes>`, `<oxy-delete>`, and `<oxy-insert>` elements.

**How to Hide Tracked Change Footnotes**

This topic is relevant if you have set the `show.changes.and.comments.as.pdf.sticky.notes` transformation parameter to `no`, and therefore the changes are shown as footnotes instead of PDF annotations.

In some cases, when your document contains a lot of tracked changes, you may need to hide the footnotes associated with the insertions, deletions, or attribute changes. Using the following example, the deletions and insertions are hidden, but the comments remain visible. In your customization CSS (on page 1814), add:

```css
oxy-attributes, 
oxy-delete, 
oxy-insert {
  float: none;
  display: none;
}
```

**How to Show Only Change Bars Without Other Styling for Tracked Changes**

It is possible to only display the change bars for tracked changes (inserted or deleted content) in the PDF document while hiding the other styling for the tracked changes. This is helpful if you want to see the document in a final version while still seeing change bars where content was inserted or deleted.

To achieve this, follow these steps:

1. Set the `show.changes.and.comments` parameter to `yes` and the `show.changes.and.comments.as.pdf.sticky.notes` parameter to `no`.

   **Step Result:** The first parameter causes tracked changes to be visible in your document and styled (e.g. insertions are blue and underlined, while deletions are red with a strike-through). Changing the second parameter to `no` causes the tracked changes to be displayed as a footnote instead of a PDF annotation.

2. Hide the footnotes by adding the following in your customization CSS (on page 1814):

   ```css
   .oxy-attributes, 
   .oxy-comment, 
   .oxy-delete, 
   .oxy-insert {
     float: initial; 
     display: none;
   }
   ```

3. Remove the change range markers (the { and } symbols):
4. Remove the styling for the insertions and deletions:

```css
.oxy-insert-hl{
  color: unset;
  text-decoration: none;
}
.oxy-delete-hl {
  content: "\200b";
  text-decoration: none;
}
.oxy-comment-hl{
  background-color: unset;
}
.oxy-color-hl[color]{
  background-color: unset;
}
```

5. **[Optional]** You can improve the visibility of the change bars with this construct:

```css
.oxy-range-start[is-changebar]:before(100) {
  -oxy-changebar-color: red;
  -oxy-changebar-width: 3pt;
}
```

**Troubleshooting**

This section contains information about fixing various change tracking, highlights and comments publishing problems.

**Highlights are Spanning Unexpectedly to the End of the Page**

**Problem**

Tracked changes and highlights span beyond what is expected.

**Cause**

If the change tracking insertions, comments, or highlights span over an area that is larger than expected, the markup that signals their end is missing.
Solution

To fix this, open the topic where the highlights start and check if the XML processing instructions that define the end of the highlighted interval are correct. The intervals are defined as follows:

For highlights:

```xml
<?oxy_custom_start type="oxy_content_highlight" color="140,255,140"?>
<?oxy_custom_end?>
```

For comments:

```xml
<?oxy_comment_start author="dan" timestamp="20201102T092905+0200" comment="Test"?>
<?oxy_comment_end?>
```

For inserted text:

```xml
<?oxy_insert_start author="dan" timestamp="20201102T093034+0200"?>
<?oxy_insert_end?>
```

Make sure all the ending processing instructions are located before the root element end tag.

Draft Watermarks

A watermark is an image displayed as the background of a printed document and it is faded enough to keep the publication text readable. Draft watermarks are used to indicate that a document is under construction or has not yet been approved.

How to Add a Draft Watermark on All Pages

To add a draft watermark to all of your publication pages, you can use the following page selector in your customization CSS (on page 1814):

```css
@page {
  background-image: url("draft.svg");
  background-position: center;
  background-repeat: no-repeat;
  background-color: inherit;
}
```

If you have already set a background image for other pages (for example, the front-page or table-of-contents), the above selector won't change them, as they are more specific.

The best practice is to use a different draft.css CSS file that imports the customization CSS where the rest of the style changes reside. If you need to publish the content as a draft, use the draft.css in your transformation scenario, otherwise directly reference the customization CSS (on page 1814).
How to Add a Draft Watermark Depending on Metadata

Suppose you want to apply a Draft watermark until your DITA bookmap is approved and the map is approved when an <approved> element has been added to the metadata section (for example, in the bookmeta/bookchangehistory element).

```xml
<bookmeta>
  <author>John</author>
  <critdates>
    <created date="1/1/2015"/>
    <revised modified="3/4/2016"/>
    <revised modified="3/5/2016"/>
  </critdates>
  <bookchangehistory>
    <approved/>
  </bookchangehistory>
  ...
</bookmeta>
```

Use oxy_xpath every time you need to probe the value from an element other than the one matched by the CSS selector, and test the expression on the merged HTML file using the Oxygen XPath Builder view.

You can either use a page selector that imposes the draft watermark on the entire page surface (recommended):

```css
@page {
  background-image: url(oxy_xpath("if(/*[contains(@class, 'bookmap/approved')]]) then '' else 'draft-watermark.png'"));
  background-position: center;
  background-repeat: no-repeat;
}
```

or use an element selector that restricts the watermark image only to the page area covered by that element:

```css
:root, body{
  ... /* same as properties above */
}
```

**Note:**

You can use another element selector to target a specific part of your publication (for example, marking only the tables as drafts).
Flagging Content

In DITA, you can mark certain content to flag it or draw attention to it. This is done by defining a flag in a DITAVAL file.

You can attach the DITAVAL file to the DITA map using the `<ditavalref>` element in the map, or by specifying it in the `args.filter` transformation parameter.

In the following example, all the elements that have the attribute `@product` set to `YourProd` is flagged to have a purple background:

```
<val>
  ...
  <prop action="flag" att="product" val="YourProd" backcolor="purple"/>
  ...
</val>
```

How to Flag Content Using Change Bars

As an example, to add a change bar (revision mark) for particular content, you can use the following in the DITAVAL file:

```
<val>
  <revprop action="flag"
    changebar="color:blue;style:solid;width:2pt;offset:1.25mm;placement:start" val="new"/>
</val>
```

This would result in any content that is marked with `@rev="new"` having a blue change bar.

How to Flag Content Using Images

You can mark the elements that match a specific profiling condition using images (one for the start, one for the end). The image references are relative to the DITAVAL file.

```
<val>
  <prop action="flag"
    att="product" val="MyProd"
</val>
```
Styling the Content

If you need to change the styles of the elements from the topic contents, you should create a customization CSS (on page 1814) and then add CSS rules. To create the CSS rules, you can use the development tools described in Debugging the CSS (on page 1816).

Reusing the Styling for WebHelp and PDF Output

If you are using the pdf-css-html5 transformation type, then the generated HTML5 document that is later converted to PDF is very similar to the generated HTML5 pages from the WebHelp Responsive output.

This is an output example from the WebHelp transformation:

```xml
<h1 class="title topic/title1" id="ariaid-title2">Care and Preparation</h1>
<div class="body">
  <p class="shortdesc">When caring ...</p>
  <p class="p">When caring for your flower garden you want ...</p>
</div>
```

And the same example from the PDF transformation (note the additional emphasized class values):

```xml
<h1 class="- topic/title title topic/title1" id="ariaid-title2">Care and Preparation</h1>
<div class="- topic/body body">
  <p class="- topic/shortdesc shortdesc">When caring ...</p>
  <p class="- topic/p p">When caring for your flower garden you want ...</p>
</div>
```

It makes sense to reuse the same CSS rules you developed for one transformation type to the other. The main rule is to use the short class names instead of the long ones. For example, to style the short descriptions with italic font, use:

```css
.shortdesc {
  font-style: italic;
}
```
The rule of thumb is that if you have a CSS rule that successfully styles an element in WebHelp, it should apply without any modification in the PDF output.

**Titles**

Titles can be styled by matching the `topic/title` class attribute.

```css
*{class="topic/title"} {
    color:navy;
}
```

**How to Avoid Wrapping the Title Text Without Indentation**

By default, the chapter/section number is on the same line as the title text. If the title is too long, the text wraps to the next line without any indentation.

```
4.5.5 This is a long title
text that wraps.
```

If you want the wrapped text to include indentation, you need to set the value of the `args.css.param.title.layout` transformation parameter to `table`. This results in the chapter/section number being placed in one cell while the rest of the title content is in another cell with wrapped text and it is displayed with an indent:

```
4.5.5 This is a long title
text that wraps.
```

**Equations**

This processor supports MathML equations.

**How to Change the Font of MathML Equations**

Suppose that you need to change the font of MathML equations from the documentation, and also add some padding. The MathML fragments are wrapped in elements that have the class `equation-d/equation-block` or `equation-d/equation-inline`, so you can match them with:

```css
*{class="equation-d/equation-block"},
*{class="equation-d/equation-inline"} {
    font-family:"courier new";
    font-size:1.5em;
    padding:1em;
}
```

**Note:**

An equation can be rendered using multiple classes of fonts (e.g. the `serif`, `sans serif`, `monospace`, `fraktur`, and `doublestruck` classes. Depending on each of the equation symbols, a class is selected...
for it. The font specified in the CSS rule (as in the preceding example), applies only to the serif class. However, if a symbol codepoint is not covered by the currently selected class fonts, it falls back to the font specified in the CSS.

**Attention:**
Some of the fonts may not be supported. In that case, a default serif font is used.

**Lists**

This is the default layout for lists (values are in px):

Markers are displayed in the padding area, so they are not included in the principal block box.

The lists are treated differently than ordinary block elements in the sense that their margins are not collapsed with the margins of the neighboring blocks or lists. This is also visible for nested lists. To summarize:

- Setting the `padding-left` or `margin-left` properties on lists will move the whole list.
- Setting the `margin-left` property on list items will move the whole list.
- Setting the `padding-left` property on list items will only move the list item content (not the marker).

**Note:**
If the `padding-left` property is set on lists and the `margin-left` property is set on list items, the result will move the whole list with a combination of both padding and margin values.

**How to Align Lists with Page Margins**

It is possible to reposition the lists to align them with the rest of the text from the body.

The default CSS rules for the lists are as follows:

```css
ol {
  display: block;
  margin-top: 1.33em;
  margin-bottom: 1.33em;
  list-style-type: decimal;
  padding-left: 40px;
}
```
To align the lists, the following rules are sufficient in the customization CSS (on page 1814):

```
*[class~="topic/ol"],
*[class~="topic/ul"] {
    padding-left: 0;
    list-style-position: inside;
}
```

**Note:**
By default, the `list-style-position` property is set to `outside`.

### How to Style the List Markers

For the media print, the default CSS rules for the list items use the `:marker` pseudo-class.

```
@media print {
    *[class="topic/li"]:marker {
        text-align: center;
        font-weight: bold;
    }
    *[class="topic/li"] {
        margin-left: 0.25in;
    }
}
```

To eliminate the bold font weight for the items form ordered lists then add the following rules to your customization CSS (on page 1814):

```
*[class="topic/ol"] > *[class="topic/li"]:marker {
    font-weight: normal;
}
```

For the unordered lists:

```
*[class="topic/ul"] > *[class="topic/li"]:marker {
    font-weight: normal;
}
```
How to Continue List Numbering

It is possible to continue the numbering of an ordered list even when the content is split on multiple `<ol>` elements.

You just need to define an `@outputclass` attribute on the lists where numbering should continue:

```xml
<ol
  @outputclass="continue">
  <li>First Item</li>
  <li>Second Item</li>
</ol>
<p>A paragraph</p>
<ol outputclass="continue">
  <li>Third Item</li>
</ol>
```

Then set the following content inside your CSS customization:

```css
*[class~="topic/ol"] {
  counter-reset: list-counter;
}

*[class~="topic/ol"] [outputclass="continue"] {
  counter-reset: none;
}

*[class~="topic/ol"] > *[class~="topic/li"] {
  counter-increment: list-counter;
}

*[class~="topic/ol"] > *[class~="topic/li"]:marker {
  content: counter(list-counter)*.00a0;
}
```

How to Change the Numbering System of Ordered Lists

It is possible to change all lists to have a different numbering system and there are several methods that can be used to achieve this.

Use the `list-style-type` CSS Property.

The Chemistry engine supports the following types: decimal, decimal-leading-zero, lower-roman, upper-roman, lower-latin, upper-latin, lower-alpha, upper-alpha.

```css
*[class = "topic/ol"] {
  list-style-type: lower-roman;
}
```
Change the Content of the :marker CSS Pseudo-Element.

The following example emulates the Cyrillic numbering for the list items for an ordered list that has the @outputclass attribute set to cyrillic:

```css
/*[class ~="topic/ol"][@outputclass ~="cyrillic"] > /*[class ~="topic/li"]:marker {
  width:3em;
}
/*[class ~="topic/ol"][@outputclass ~="cyrillic"] > /*[class ~="topic/li"]:nth-of-type(1):marker{ content:"а" }
/*[class ~="topic/ol"][@outputclass ~="cyrillic"] > /*[class ~="topic/li"]:nth-of-type(2):marker{ content:"б" }
/*[class ~="topic/ol"][@outputclass ~="cyrillic"] > /*[class ~="topic/li"]:nth-of-type(3):marker{ content:"в" }
/*[class ~="topic/ol"][@outputclass ~="cyrillic"] > /*[class ~="topic/li"]:nth-of-type(4):marker{ content:"г" }
/*[class ~="topic/ol"][@outputclass ~="cyrillic"] > /*[class ~="topic/li"]:nth-of-type(5):marker{ content:"д" }
/*[class ~="topic/ol"][@outputclass ~="cyrillic"] > /*[class ~="topic/li"]:nth-of-type(6):marker{ content:"е" }
/*[class ~="topic/ol"][@outputclass ~="cyrillic"] > /*[class ~="topic/li"]:nth-of-type(7):marker{ content:"ж" }
/*[class ~="topic/ol"][@outputclass ~="cyrillic"] > /*[class ~="topic/li"]:nth-of-type(8):marker{ content:"з" }
/*[class ~="topic/ol"][@outputclass ~="cyrillic"] > /*[class ~="topic/li"]:nth-of-type(9):marker{ content:"и" }
/*[class ~="topic/ol"][@outputclass ~="cyrillic"] > /*[class ~="topic/li"]:nth-of-type(10):marker{ content:"к" }
/*[class ~="topic/ol"][@outputclass ~="cyrillic"] > /*[class ~="topic/li"]:nth-of-type(11):marker{ content:"л" }
/*[class ~="topic/ol"][@outputclass ~="cyrillic"] > /*[class ~="topic/li"]:nth-of-type(12):marker{ content:"м" }
/*[class ~="topic/ol"][@outputclass ~="cyrillic"] > /*[class ~="topic/li"]:nth-of-type(13):marker{ content:"н" }
/*[class ~="topic/ol"][@outputclass ~="cyrillic"] > /*[class ~="topic/li"]:nth-of-type(14):marker{ content:"о" }
/*[class ~="topic/ol"][@outputclass ~="cyrillic"] > /*[class ~="topic/li"]:nth-of-type(15):marker{ content:"п" }
```

⚠️ Important:

This example will work only for lists up to 28 items. You will have to extend it for longer lists!
Links allow the users to navigate through the documentation.

How to Remove 'on page NNN' Link Label

For printed material, it is usually desirable for the links to display a label after the text content (such as "on page 54"). This makes it easier the user to identify the target page. However, if the produced PDF is not printed and is intended only for electronic use, this label may create clutter and make the document harder to read. To eliminate this label, add the following in your customization CSS (on page 1814):

```css
* [link]:after, *
```
Another use-case is to remove the labels only from links shown in tables cells, and leave the others as they are. For this, you could use a more specific selector:

```css
* [class = "topic/entry"] * [class = "topic/xref"] [href]:after {
    content: none !important;
}
```

### How to Change Link Styles

Suppose you want the links to be bold and with an underline. In your customization CSS (on page 1814), add this snippet:

```css
* [class = "topic/xref"] [href]:after,
* [class = "topic/link"] [href]:after {
    font-weight: bold;
    text-decoration: underline;
}
```

### How to Hide Descriptions in Related Links Sections

The link descriptions that come from DITA relationship tables or related link elements within topics, are structured in the merged map (on page 1816) like this:

```xml
<related-links class="- topic/related-links ">
    <linkpool class="- topic/linkpool ">
        <link class="- topic/link "
            ...
            <role="friend" scope="local" type="topic">
                <linktext class="- topic/linktext ">Salvia</linktext>
                <desc class="- topic/desc ">The salvia plant</desc>
            </role>
        </link>
    </linkpool>
    ...
</related-links>
```

If you need to hide these descriptions, add the following code in your customization CSS (on page 1814):

```css
* [class = "topic/link"] > * [class = "topic/desc"] {
    display: none;
}
```
How to Group Related Links by Type

By default, all links from DITA relationship tables or related link elements within topics are grouped under one "Related information" heading:

<table>
<thead>
<tr>
<th>Related information</th>
</tr>
</thead>
<tbody>
<tr>
<td>Target Topic</td>
</tr>
<tr>
<td>Target Concept</td>
</tr>
<tr>
<td>Target Task</td>
</tr>
</tbody>
</table>

It is possible to group the links by target type (topic type) by setting the `args.rellinks.group.mode=group-by-type` parameter. The output will look like this:

<table>
<thead>
<tr>
<th>Related concepts</th>
</tr>
</thead>
<tbody>
<tr>
<td>Target Concept</td>
</tr>
<tr>
<td>Related tasks</td>
</tr>
<tr>
<td>Target Task</td>
</tr>
<tr>
<td>Related information</td>
</tr>
<tr>
<td>Target Topic</td>
</tr>
</tbody>
</table>

Images and Figures

Images are an important part of a publication.

Note:

You can use raster image formats (such as PNG or JPEG), but it is best to use vector images (such as SVG or PDF). They scale very well and produce better results when printed. In addition, the text from these images is searchable and can be selected (if the glyphs have not been converted to shapes) in the PDF viewer.

Images - Built-in CSS

Image properties are defined in `[PLUGIN_DIR]css/print/p-figures-images.css`.

```
*[class =~ "topic/image"] {  
    prince-image-resolution: 96dpi;
    -ah-image-resolution: 96dpi;
    image-resolution: 96dpi;
    max-width: 100%;
}
```

How to Fix Image Bleeding - Control Image Size

Sometimes the images may be too big for the page. The built-in CSS rules specify a maximum size for images, limiting to the width of the parent block. But if the parent block is itself too wide and bleeds out of page, you might consider specifying a length.
In your customization CSS (on page 1814), add the following snippet:

```
*[class =~ "topic/image"] {
  ...
  /* The US-letter page size minus page margins. See p-page-size.css for the current page size. */
  max-width: 6.5in;
}
```

Pay attention to images that have an image map (on page 1959) associated. The built-in rules set the `max-width: auto` for them to avoid scaling. Otherwise, it would cause a misalignment between the image and its clickable areas. These images are best to have a `@width` and `@height` attribute.

**How to Change Image Resolution**

**How to Change the Resolution for Raster Images**

This technique changes the size of all raster images from your documentation. It will not work for vector images, such as PDF or SVG.

The default resolution is 96 dpi (same as in web browsers). You can change it by adding the following in your customization CSS (on page 1814):

```
*[class =~ "topic/image"] {
  prince-image-resolution: 300dpi;
  -ah-image-resolution: 300dpi;
  image-resolution: 300dpi;
}
```

**Important:**

The above selector does not apply to images from the `<imagemap>` element. You can use the following selector for that purpose:

```
*[class =~ "ut-d/imagemap"] > *[class =~ "topic/image"] {
  ...
}
```

Make sure you verify the area shapes to match the new image boundaries. The pixels specified in the image map area coordinates are always 1/96 in. For more details, see: How to Use Image Maps (on page 1959).

**How to Change the Resolution for Vector Images**

This technique will change the size of all vector images (such as PDF or SVG) and will not affect raster images.
Vector images are rendered with a default resolution of 96 dpi. You can change this default value by setting the `image.resolution` transformation parameter (on page 1773) to another value (from 72, 120, 300 and 600).

**How to Place Big Images on Rotated Pages**

Very wide images may bleed out of the page. One solution for this is to use landscape pages for these wide images.

In your customization CSS (on page 1814), add:

```css
*[class="topic/image"] [outputclass='land'] {
  page: landscape-page;
}
```

Setting the attribute `@outputclass = 'land'` on the table element will force the table on a new landscape page.

Another solution is to set an `@outputclass` attribute on the image, then create a rule that matches it, and associate a landscape page for it.

**How to Place a Text and Image Side by Side**

If you need to align text and an image side by side, you can use the following technique:

1. Organize your text and image under a `<div>` element like this:

   ```xml
   <div outputclass="side-by-side">
     <p>This will be in the left side, the next figure in the right. </p>
     <fig>
       <image href="cactus.jpeg"/>
     </fig>
   </div>
   ```

   **Note:**
   You can use the `@outputclass` attribute to mark the `<div>` elements that have this special layout.

2. In your customization CSS (on page 1814), add:

   ```css
   *[outputclass="side-by-side"] > *[class="topic/p"] {
      display:inline-block;
      width: 45%;
    }

    *[outputclass="side-by-side"] > *[class="topic/fig"] {
      display:inline-block;
      width: 45%;
    }
   ```
The image should fill the entire width of the parent `<fig>` element:

```css
*[^outputclass ~="side-by-side"] > *[^class ~="topic/fig"] > *[^class ~="topic/image"] { width:100%; }
```

By default, the bottom of the image is on the same line as the text baseline. If you want the text and the image to be aligned at the top, add these lines:

```css
*[^outputclass ~="side-by-side"] > *[^class ~="topic/p"] { vertical-align:top; }

*[^outputclass ~="side-by-side"] > *[^class ~="topic/fig"] { vertical-align:top; font-size:0pt; }
```

**Note:**
The `font-size:0pt` is needed to remove the font ascent and descent around the image rectangle.

### How to Control the Image Size in Complex Static Content

It is common to have text and images mixed together in a `:before` or `:after` pseudo-element. For example, for notes you may have both artwork and text:

```css
[^class ~="topic/note"]:before { content: url('note.png') "Some text";
```

If you want to change the size of the image, you have two options:

- **Use the `image-resolution` CSS property:**

  ```css
 [^class ~="topic/note"] { image-resolution:300dpi; }
  ```

- **Separate the image from the text and apply the width and height CSS properties only on the image, using the width and height properties. You could use multiple `:before` pseudo-elements for that, considering that the farthest content presented before the actual content of an element is matched by the `:before` with the highest number in the brackets:**

  ```css
 [^class ~="topic/note"]:before(2) { content: url('note.png');
  ```
How to Center Images

DITA defines a `@placement` attribute for the `<image>` elements. The implicit value is `inline`. Suppose that you need to center the images that have the placement set to `break` (for example, they are not on the same line with other content and the images from the `<fig>` element).

In your customization CSS (on page 1814), add:

```css
/* Other images, with break placement. */
*[class ~="topic/image"] {placement='break']{
    display:block;
    text-align:center;
}

/*
Scaled images are getting a computed width attribute, so we can use the auto margins.
Auto margins function only if the block they apply to has a width.
*/
*[class ~="topic/image"] {placement='break'] [width] {
    margin-left:auto;
    margin-right:auto;
    border: 2pt solid red;
}
```

How to Change/Reset the Figure Numbering

Note:

This topic is applicable for the DITA Map PDF - based on HTML5 & CSS DITA PDF - based on HTML5 & CSS transformation types.

There are cases when you need to change the aspect of the figure counter that is shown before the figure titles. By default, the figure titles are formatted like this:
Figure NN. Lore Ipsum Title

NN is the number of the figure that starts being counted from the beginning of the publication.

One use-case is to have the NN counter be incremented only within one chapter (for example, the first chapter contains "Figure 1" and "Figure 2", and the second chapter starts over with "Figure 1" instead of incrementing to "Figure 3").

You should reset the figure counter on each topic marked as chapter, then hide the label from the figure <figcaption> (this is an HTML element generated by the XSL transformation), and create another label using a :before selector on the <figcaption>.

```html
/*[class ~="topic/topic"] [is-chapter] { 
    counter-reset: figcount;
}

 fig--title-label{
    display:none;
}

/*[class ~="topic/fig"] > .figcap:before{
    /* Add more styling or change the content if needed */
    content: "Figure " counter(figcount) ". ";
    counter-increment: figcount;
}
```

How to Fix Missing Images

If your images are not accessible, you may receive an error message in the transformation console like this:

```plaintext
Image not found. URI:file:/path/to/my/image
```

This is usually because they are in a folder that is not in the folder subtree of the transformed map or topic.

To solve this, you can set the following transformation parameter: fix.external.refs.com.oxygenxml=true.

How to Use Image Maps

To use the DITA <imagemap> element in a PDF transformation, follow this procedure:
1. Start by determining the width and height of your image in *CSS pixels* and specify it on the `<image>` element using the `@width` and `@height` attributes.

**Notes:**
- A CSS *pixel* is 1/96 in, so if the image is created at a 96dpi resolution, one dot from the image is one pixel in the CSS space. If your image is displayed at another resolution (*on page 1955*), then it is adjusted accordingly (for example, 192dpi results in two dots from the image being equal to one pixel in the CSS space).
- You can use other CSS units, including percentages. The percentages are solved relative to the image size and represent a way of creating *responsive* image maps.

**Warning:**
If you publish the content for both PDF and HTML web output, make sure you only use *pixels*, as some browsers only support these units.

**Example:**
Suppose you have a very large image that is 6400x4800 dots, but you want to make it fit in a box of 640x480 CSS *pixels*. In the following snippet, this is done by specifying the width and height attributes. The areas must use coordinates relative to these values.

```xml
<imagemap>
  <image href="../images/Gear_pump_exploded.png" id="gear_pump_exploded"
        width="640"
        height="480">
    <alt>Gear Pump</alt>
  </image>
</imagemap>
```

2. In the map element, add areas (each with a shape and a set of coordinates):

```xml
<imagemap>
  <image ...> ... </image>
  <area>
    <shape>circle</shape>
    <coords>172, 265, 14</coords>
    <xref href="parts/bushings.dita#bushings_topic/bushings"
          format="dita">Bushings</xref>
  </area>
  <area>
    <shape>poly</shape>
    <coords>568, 81, 576, 103, 468, 152, 455, 130</coords>
  </area>
</imagemap>
```
The type of areas are the ones defined in the HTML standard: circle, poly, rect, default. For more details, see: https://html.spec.whatwg.org/multipage/image-maps.html#the-area-element.

3. Verify how the shapes look in the output. You can make the shapes visible by one of the following methods:

- Using the `show.image.map.area.numbers` and `show.image.map.area.shapes` transformation parameters.
- Adding a CSS snippet to your customization. The shapes have the `image-map-shape` class, the bullet around the image map number (`image-map-number`), and the text inside the bullet (`image-map-number-text`). To make them translucent yellow:

```css
.image-map-shape{
  fill: yellow;
  fill-opacity: 0.5;
  stroke-opacity: 0.5;
}
.image-map-number-text {
  visibility: visible;
}
.image-map-number {
  fill: yellow;
  fill-opacity: 0.4;
  stroke-opacity: 0.7;
}
```

**Tip:**

An SVG with links can be used as an alternative to the DITA `<imagemap>` element. Make sure that each link is a relative URI to an ID inside the publication content.

### How to Hide the Image Map Links List

Below every image map, a list of links that point to the image map targets is displayed. This list can be hidden from the final output by using the following CSS selector:

```css
.imagemap--areas {
  display: none;
}
```
How to Use SVG Syntax Diagrams

The DITA `<syntaxdiagram>` element is supported by the PDF transformation. To use SVG syntax diagrams, follow this procedure:

1. Download the latest version of the `svg-syntaxdiagrams` plugin, unzip it, and copy all the folders into your `DITA-OT-DIR\plugins` folder (they all start with "com.").
2. Open a command prompt inside `DITA-OT-DIR\bin` and run the dita install command.
3. You can now add your custom `<syntaxdiagram>` element in your topic, as in the following example:

   ```xml
   <syntaxdiagram id="syntaxdiagram_ok4_clk_xnb">
     <title>CopyFile</title>
     <groupseq><kwd>COPYF</kwd></groupseq>
     <groupcomp><var>input-filename</var><kwd>*INFILE</kwd></groupcomp>
     <groupseq><var>output-filename</var><kwd>*OUTFILE</kwd></groupseq>
     <groupchoice><var>input-filename</var><kwd>*INFILE</kwd></groupchoice>
     <groupchoice><var>output-filename</var><kwd>*OUTFILE</kwd></groupchoice>
   </syntaxdiagram>
   ```

4. Run the DITA Map PDF - based on HTML5 & CSS (or DITA PDF - based on HTML5 & CSS) transformation.

   **Warning:**
   If you are not publishing the content for the first time, you may need to delete the `out/` and `temp/` folders to see the syntax diagram correctly in the `.merged.html` file.

Result: The PDF is generated and the syntax diagram is displayed as a referenced SVG file like this:

![SVG Diagram]

Tables

Tables are widely used in technical documentation. This section contains information about the CSS rules that are used to style them and how to fix some problems.

Tables - Built-in CSS

There is a combination of CSS files that address tables:

- `[PLUGIN_DIR]/css/core/-table-html-cals.css`
- `[PLUGIN_DIR]/css/print/p-tables.css`
**How to Handle Wide Tables - Page Rotation**

Some of the tables can have a large number of columns. In this case, the table may bleed out of the page. One solution is to use landscape pages for these tables.

Setting the attribute `orient = 'land'` attribute on the table element will force the table to be on a new landscape page.

Another solution is to use automatic detection of wide tables (5 or more columns):

```css
* [class~="topic/table"] [data-cols='5'],
* [class~="topic/table"] [data-cols='6'],
* [class~="topic/table"] [data-cols='7'],
* [class~="topic/table"] [data-cols='8'],
* [class~="topic/table"] [data-cols='9'],
* [class~="topic/table"] [data-cols='10'] {
  page: landscape-page;
  max-width: 100%;
  max-height: 100%;
  width: 100%;
  page-break-before: avoid;
}
```

**Note:**
The `landscape-page` page layout is defined in the `[PLUGIN_DIR]/css/print/p-pages-and-headers.css` file.

If you want to rotate the entire topic that contains the big table, use:

```css
* [class~="topic/table"] [data-cols='5'],
* [class~="topic/table"] [data-cols='6'],
* [class~="topic/table"] [data-cols='7'],
* [class~="topic/table"] [data-cols='8'],
* [class~="topic/table"] [data-cols='9'],
* [class~="topic/table"] [data-cols='10'] {
  max-width: 100%;
  table-layout: auto;
}
* [class~="topic/topic"] :has(':not([class~="topic/topic"]) >
  * [class~="topic/table"] [data-cols='5']),
* [class~="topic/topic"] :has(':not([class~="topic/topic"]) >
  * [class~="topic/table"] [data-cols='6']),
* [class~="topic/topic"] :has(':not([class~="topic/topic"]) >
  * [class~="topic/table"] [data-cols='7']),
```
**Note:**

It is also possible to import the [PLUGIN_DIR]/css/print/p-optional-auto-rotate-wide-tables.css stylesheet into your custom CSS.

### How to Avoid a Table Exceeding the Page Width

The DITA specification indicates that tables should have a fixed layout. This can be done in two different ways:
1. **Using proportional or relative measures** - It includes percent values and shares values (i.e. "3%" or "12").

2. **Using fixed measures** - It includes all the values followed by units (i.e. in, pt, px, and others).

---

**Important:**

Although the specification allows you to combine these values, it is **highly recommend** that you only use one method at a time. Combining both methods could lead to a table exceeding the page width and will make the content unreadable.

---

### How to Fix Text Bleeding From Table Cells

Slim tables or tables that have many columns make the text from the cells be confined to a small horizontal space. Sometimes this causes long words to bleed outside the cell boundaries.

By default, the built-in CSS automatically activates the hyphenation for the text inside tables as long as your topics have the language specified.

In case the text is still bleeding outside the boundaries, you can also use the `overflow-wrap` property to force the word to break:

```css
*[class =~ "topic/table"] { 
  overflow-wrap: break-word;
}
```

---

**Related Information:**

- [Hyphenation](#)
- [How to Enable/Disable Hyphenation for an Element](#)

---

### How to Fix Small Images in Table

Tables contained in the output of DITA Map PDF - based on HTML5 & CSS (and DITA PDF - based on HTML5 & CSS) transformations have an automatic layout by default. This means that DITA-OT defines a preferred size on them, optimizing their width/height inside the content to make them as small as possible.

If, for example, you have a two-column table **without defined column widths** and one column contains images while the other column contains text, the table in the generated PDF will have its first column shrunk with smaller images and an enlarged second column (to occupy the least amount of space in the output).

To avoid this, you must **unset** the default image `max-width` so that the original size of the image will be used instead:

```css
*[class =~ "topic/image"] { 
  max-width: unset;
}
```
How to Rotate Content from a Table Cell

In DITA CALS tables, you can rotate the content of a cell by setting the @rotate attribute to 1.

In the following example, the Sport, All terrain, and Family header cells are rotated.

```
<table frame="all" rowsep="1" colsep="1" id="table_dlp_fib_cr1">  
  <title>Car Features</title>  
  <group cols="4">  
    <colspec colname="c1" colnum="1" colwidth="14*/">  
    <colspec colname="c2" colnum="2" colwidth="1*" align="center">  
    <colspec colname="c3" colnum="3" colwidth="1*" align="center">  
    <colspec colname="c4" colnum="4" colwidth="1*" align="center">  
  </group>  
  <thead>  
    <row>  
      <entry morerows="1">Car Name</entry>  
      <entry namest="c2" nameend="c4">Features</entry>  
    </row>  
    <row>  
      <entry rotate="1">Sport</entry>  
      <entry rotate="1">All terrain</entry>  
      <entry rotate="1">Family</entry>  
    </row>  
  </thead>  
  <tbody>  
    <row>  
      <entry>Tesla Model S</entry>  
      <entry>X</entry>  
      <entry>x</entry>  
    </row>  
    ...  
  </tbody>  
</table>
```

Table 43. Car Features

<table>
<thead>
<tr>
<th>Car Name</th>
<th>Features</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Sport</td>
</tr>
<tr>
<td>Tesla Model S</td>
<td>X</td>
</tr>
</tbody>
</table>

The resulting output will be:
The built-in CSS matches the cells with this attribute and applies the following properties:

```
*{[class~="topic/entry"] [rotate='1'] {  
    transform: rotate(270deg);  

    /* Avoid wrapping, including hyphenation */
    white-space: pre;
    hyphens: manual;

    /* The rotated content will start from the lower side of the cell */
    vertical-align: bottom;
}
```

To change the vertical alignment of the content (for example, to move it to the middle of the cell), use the following in your CSS customization:

```
*{[class~="topic/entry"] [rotate='1'] {  
    vertical-align: middle;
}
```

The resulting output will be:
To make the text wrap (for instance, the "All terrain" could be split on two lines), you need to inhibit the whitespace preservation from the built-in CSS. In this case, all spaces will create a line break in the rotated layout. Thus, you can add this in your customization:

```css
*[^class*="topic/entry"][^rotate*="1"] {
  vertical-align:middle;
  white-space:normal;
}
```

The resulting output will be:

<table>
<thead>
<tr>
<th>Car Name</th>
<th>Features</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Sport</td>
</tr>
<tr>
<td>Tesla Model S</td>
<td>X</td>
</tr>
<tr>
<td>Nissan Leaf</td>
<td></td>
</tr>
<tr>
<td>Dacia Duster</td>
<td></td>
</tr>
</tbody>
</table>

Note:
The padding and borders set on the table cells are not rotated (only the content of the cell is rotated). You can use `padding-left` (for instance) to move the labels to the horizontal axis.

```css
*[^class*="topic/entry"][^rotate*="1"] {
  padding-left:2em;
}
```

How to Add Horizontal Lines to a Choice Table

To add horizontal lines that separate the options within a `<choicetable>`, you can use borders set on each of the rows. The following CSS styles the top header and the first column with some background colors. In a choice table, the first column represents the choice labels.

```css
*[^class*="task/choptionhd"],
*[^class*="task/choptionhd"],
*[^class*="task/chdeschd"],
*[^class*="task/choption"] {
  background-color: #EEEEEE;
  text-align: left;
}
```

```css
*[^class*="task/choicetable"] {
```
Note:
Using the frame attribute on the choice table will make these selectors apply partially. Please make sure you are designing your customization CSS taking into account all possible values for the frame attribute.

How to Remove the Table NN Label

For the DITA Map PDF - based on HTML5 & CSS transformation scenario, the label for a table’s title is wrapped in a span element with the class: `table--title-label`.

```html
<table ... >
  ...
  <caption class="- topic/title title tablecap">
    <span class="table--title-label">Table</span>
    <span class="table--title-label-number">1.</span>
  </caption>
  ...
```

To hide it, set its display to none:

```
.table--title-label {
  display: none;
}
```

For the direct transformation, use:

```
*[class ^= "topic/table"] > *[class ^= "topic/title"]:before {
  content: none;
}
```
How to Center Tables

You can center the tables by using margins auto, while the table caption (title) can be centered using the text-align property:

```css
* [class = "topic/table"] {
    margin-left: auto;
    margin-right: auto;
    width: 50%;
    border: 1pt solid blue;
}
* [class = "topic/table"] * [class = "topic/title"]{
    text-align: center;
}
```

How to Add Stripes to a Table

To create a striped look for your tables, you can use the following CSS rules:

```css
/* Header background and foreground */
* [class = "topic/table"] [outputclass = "stripes"] > * [class = "topic/thead"] > * [class = "topic/row"] {
    background-color: blue;
    color: white;
}

/* A default background for the entire table body */
* [class = "topic/table"] [outputclass = "stripes"] > * [class = "topic/tbody"] {
    background-color: #eeeeee;
}

/* Color for the stripes */
* [class = "topic/table"] [outputclass = "stripes"] > * [class = "topic/tbody"] > * [class = "topic/row"]:nth-child(odd) {
    background-color: cyan;
}

/* Border for the cells */
* [class = "topic/table"] [outputclass = "stripes"] * [class = "topic/entry"] {
    border: blue;
}
```

The above rules assume that tables that are to be painted with stripes are marked with an `@outputclass` attribute:
If you want to make all tables look the same, you can ignore this attribute and remove the 

<code class="html">[outputclass ~="stripes"]</code> simple selector from the above rules.

**CAUTION:**
Applying stripes and thin cell borders can cause rendering issues in the PDF renderer on screen display devices. For more information, see How to Fix Disappearing Thin Lines or Cell Borders (on page 1986).

### How to Display Borders on a Split Cell

By default, if a cell extends onto a second page, its bottom and top borders are discarded. To display these borders, you need to add the following property in your CSS customization:

```css
*[class =~ "topic/entry"] {
    -oxy-borders-conditionality: retain;
}
```

### Code Blocks

Code blocks are used to render section of programming code.

### How to Enable Code Syntax Highlighting

This topic refers only to the DITA Map PDF - based on HTML5 & CSS transformation type.

You can use syntax highlighting to make it easier to read your code snippets by displaying each type of code in different colors and fonts. In the DITA topics, set the `@outputclass` attribute on the `<codeblock>` elements to one of these values:

- language-bourne
- language-c
- language-cpp
- language-csharp
- language-css
- language-ini
- language-java
- language-javascript
- language-json
- language-lua
- language-perl
- language-php
- language-python
- language-ruby
For example, for a java snippet:

```html
<codeblock outputclass="language-java">
   for (int i=0; i < 100; i++) {
      // do something
   }
</codeblock>
```

The resulting HTML fragment in the merged HTML5 document is:

```html
<pre class="hl keyword language-java">
   for (<span class="hl-number">int</span> i=0; i <span class="hl-number">100</span>; i++) {
      // do something
   }
</pre>
```

And in the output, it will be rendered as:

```java
for (int i=0; i < 100; i++) {
    // do something
}
```

### Changing the Colors for the Syntax Highlighting

As you can see in the above example, the HTML elements `<span>` and `<strong>` are used to color the content. Since they have a `@style` attribute set, the overriding properties need to be marked with `!important`.

Suppose you want to color the keywords in red and the comments in blue. To do so, add the following to your customization CSS (on page 1814):

```css
.hl-keyword {
   color: red !important;
}
.hl-comment {
   color: blue !important;
}
```
How to Disable Line Wrapping in Code Blocks

By default, code blocks have the content wrapped to avoid the bleeding of long lines out of the page. To avoid wrapping, add the following in your customization CSS (on page 1814):

```
*{class~="pr-d/codeblock"} {
  white-space: pre;
}
```

For the DITA Map PDF - based on HTML5 & CSS transformation type, the best solution to distinguish between lines is to leave them wrapped, but color each line with a different background (zebra coloring). An example is provided here: XSLT Extensions for PDF Transformations (on page 1794).

How to Deal with Unwanted Returns in Code Blocks

There are cases where the source file contains long lines of code that need to continue onto the next line in the rendered PDF (to wrap visually).

When the user copies the block from the PDF reader, they get two separated lines. This means that the command fails when users copy it from the PDF to the command-line terminal (because it comes in as two commands).

For example, the command:

```
$gist = ls -l * | count -n | some more
```

May be rendered in the PDF on two lines:

```
$gist = ls -l * | count -n
| some more
```

And this is invalid when used in the terminal.

There is no CSS workaround for this, but to manually format the command line, add a line continuation character like this:

```
$gist = ls -l * | count -n \
| some more
```

**Note:**

For Linux/macOSX, the continuation character is the backslash \. For Windows, this is the shift character ^.

The command-line processor will now recognize that the first line is continuing on to the next one.
Notes

Notes contain an additional piece of information that calls attention to particular content. They may have various types (tip, caution, danger, restriction, important, warning).

For information on how to add and manage mixed content before the note icons and labels, see How to Control the Image Size in Complex Static Content (on page 1957).

How to Change Note Icons

The recommended icon format is SVG.

To change the default icons for notes that do not have a @type attribute, add the following rule to your customization CSS (on page 1814):

```css
div.note {
    background-image:url("../img/note.svg");
}
```

For a note with a @type attribute set to warning, caution, or trouble, add the following corresponding CSS rule:

```css
div.warning {
    background-image:url("../img/warning.svg");
}
div.caution {
    background-image:url("../img/caution.svg");
}
div.trouble {
    background-image:url("../img/troubleshooting.svg");
}
```

Abbreviated Forms

When using the <abbreviated-form> element in your content, it is possible to style the subsequent occurrences differently than the first occurrence. To achieve this, add something similar to the following rule in your customization CSS (on page 1814):

```css
a:has(dfn[class ~="abbreviated-form"]) {
    color: oxy_xpath("let $cdf:= dfn return if (preceding::dfn[@keyref = $cdf/@keyref]) then 'black' else 'red'");
    text-decoration: oxy_xpath("let $cdf:= dfn return if (preceding::dfn[@keyref = $cdf/@keyref]) then 'none' else 'underline'");
}
```

This example would render the first occurrence with a red color and an underline, while the subsequent occurrences would be rendered with a black color and no underline.
Trademarks

Trademarks are used to specify legally registered words and they are often used in technical documentation. To specify a trademark, your DITA content could use a structure similar to this:

```
<tm tmtype="tm">My Product Name</tm>
```

Depending on the value of the `tmtype` attribute, a different symbol is appended to the text: (®, ™, or ℠).

The structure of the merged HTML document the CSS will apply to is:

```
<span class="- topic/tm tm tmtype="tm">My Product Name</span>
```

How to Style the Trademark Element Text

To change the style of the entire trademark text, you can match the `topic/tm` class like this:

```
*[@class = "topic/tm"] {
  font-weight: bold;
}
```

How to Style the Trademark Symbol

To change the aspect of the trademark symbol, you can use the `topic/tmmark` class. Usually, common fonts already render these symbols smaller and with superscript by default. The following example does it from the CSS:

```
*[@class = "topic/tmmark"] {
  vertical-align: super;
  font-size: smaller;
}
```

Styling Through Custom Parameters

You can activate parts of your CSS by using custom transformation parameters that start with the `args.css.param` prefix.

These parameters are recognized by the publishing pipeline and are forwarded as synthetic attributes on the root element of the merged map. The last part of the parameter name will become the attribute name, while the value of the parameter will become the attribute value. The namespace of these synthetic attributes is: `http://www.oxygenxml.com/extensions/publishing/dita/css/params`.

When using the DITA Map PDF - based on HTML5 & CSS or the DITA PDF - based on HTML5 & CSS transformations, the generated attribute will be in no namespace.
Notes:

- Make sure the name of your custom parameter does not conflict with an attribute name that may already exist on the root element.
- Use only Latin alphanumeric characters for parameter names.
- You can set multiple styling parameters at the same time.

How to Limit the Depth of the TOC Using a Parameter

In the following example, a custom parameter is used to switch from a full depth table of contents to a flat one that shows only the titles of the first-level topics (such as chapters, notices, or the preface).

The custom parameter is:

```xml
<param name="args.css.param.only-chapters-in-toc" value="yes"/>
```

The CSS that hides the `topicrefs` at level 2 or more:

```
> *[class ^="toc/toc"] > *[class ^="map/topicref"] { display:none; }
```

The `:root[only-chapters-in-toc='yes']` selector makes the rule activate only when the attribute is set.

How to Change the Page Size Using a Parameter

In the following example, a custom parameter is used to modify the page size. The parameter is defined in the transformation scenario as:

```xml
<param name="args.css.param.page-size" value="A4"/>
```

Then in the CSS, the attribute value is extracted and used as follows:

```
@page {
  size: oxy_xpath('//*[@local-name()="page-size"]')[1];
}
```

How to Change the Cover Page Using a Parameter

In the following example, a custom parameter is used to set the path of the cover page. The parameter points to an image by using its URL and is defined in the transformation scenario as:

```xml
<param name="args.css.param.cover-page" value="file:/path/to/cover-page.svg"/>
```

Then in the CSS, the attribute value is extracted and used as follows:
Controlling the Publication Content

Using a plain DITA map, the transformation will produce a publication with a front page, a table of contents, chapters with content, and an index at the end. This is appropriate for most cases, but there are use cases where some adjustments are necessary. For example, if you want to do one of the following:

- Remove the TOC or index.
- Add a glossary.
- Change the position of the TOC or the index relative to the sibling topics.
- Add a preface, frontmatter, or backmatter with copyright notices, abstracts, list of tables, list of figures, etc.

All of these can be achieved using a DITA `<bookmap>` element.

A bookmap has a more elaborate structure than a regular map. You should start by defining the title structure, with a main title and alternative title:

```xml
<!DOCTYPE bookmap PUBLIC "-//OASIS//DTD DITA BookMap//EN" "bookmap.dtd">
<bookmap id="taskbook">
  <booktitle>
    <mainbooktitle>Publication Title</mainbooktitle>
    <booktitlealt>A very short description of the publication</booktitlealt>
  </booktitle>
</bookmap>
```

Then you may define a `frontmatter`. For this, you can link the topics that need to appear before the main content. You can also define the location where the table of contents will be placed. In the example below, it appears between the `abstract.dita` and `foreword.dita` topics:

```xml
<frontmatter>
  <topicref href="topics/abstract.dita"/>
  <booklists>
    <toc/>
  </booklists>
  <topicref href="topics/foreword.dita"/>
</frontmatter>
```

Note:
To remove the TOC from the publication, just omit the `<toc>` element from the `<booklists>` element.

Next, the topics are grouped into chapters:
At the end, you could define the structure of the backmatter. Just like for the frontmatter, you can include some topics and some generated content (such as the index). In the example below, the glossary is defined to come after the index, followed by a list of figures and list of tables. At the very end, there is a topic with some thank you notes.

```xml
<backmatter>
  <topicref href="topics/conclusion.dita"/>
  <booklists>
    <indexlist/>
    <glossarylist>
      <topicref href="topics/xp.dita" keys="xp" print="yes"/>
      <topicref href="topics/anti_lock_braking_system.dita" keys="abs" print="yes"/>
    </glossarylist>
    <figurelist/>
    <tablelist/>
  </booklists>
  <topicref href="topics/thanks.dita"/>
</backmatter>
```

As you can see, the bookmap offers much better control over the final content of the publication. It also offers more options in controlling the metadata that will go into the PDF (see the Metadata (on page 1863) topic).

**How to Omit the Front Page, TOC, Glossary, Index for a Plain DITA Map**

For a plain DITA map, there are no elements that allow you to control if and where to place the generated content such as the title page, table of contents, list of tables, glossary, or index. For the most common use-case, when you want to hide them all and just keep the content, you can use the transformation parameter `hide.frontpage.toc.index.glossary`. See: Transformation Parameters (on page 1773).

**Related Information:**
- How to Remove Entries from the TOC (on page 1897)
- How to Hide the TOC (on page 1897)

**How to Make Chapters Look Like Individual Publications**

Note:
This topic is only applicable for the DITA Map PDF - based on HTML5 & CSS transformation scenario.
Sometimes you want to make each chapter independent (i.e. it can be read separately, as a separate part of your publication). For this, you need the page counter, figure, and table counters to restart at each chapter. You can control this by using the `args.css.param.numbering` (on page 1879) command-line parameter.

In addition to numbering, you can force the creation of a chapter TOC (on page 1897).

**Localization**

DITA-OT supports more than 40 languages. The full list of supported languages (and their codes) is available here: https://www.dita-ot.org/3.6/topics/globalization-languages.html.

There are two ways to switch the labels to a specific language:

- Set the `@xml:lang` attribute on the DITA maps and/or topics root element with one of the supported values (e.g. `de`, `fr-FR`, `ru`, `zh-CN`).
- Set the `default.language` parameter in the transformation dialog box to the desired language code.

You can create language-dependent CSS rules in your customization CSS (on page 1814) by adding rules using the `:lang` pseudo-class (see https://developer.mozilla.org/en-US/docs/Web/CSS/:lang).

Tip:

It is recommended that you do this customization on a DITA-OT distribution deployed outside of the **Oxygen** installation. Otherwise, you will lose the customization when upgrading **Oxygen**. You can contact the **Oxygen support team** to ask for the **Oxygen Publishing Engine** package.

**How to Customize CSS Strings**

Some of the labels come from CSS files located in the `DITA-OT-DIR/plugins/com.oxygenxml.pdf.css/css/print/i18n` directory. These strings can be overridden directly from a custom CSS stylesheet. Simply identify (by debugging the CSS) and copy the rules that apply on your content and change their values. For example:

```css
/* Title of the TOC page */
*[class =~ "toc/title"] [empty]:lang(es):before {
    content: "Contenidos";
}
```

Note:

If you want to use a language without a corresponding `p-i18n-xx.css` stylesheet, follow these instructions:
1. Copy one of the available stylesheets (located in the `DITA-OT-DIR/plugins/com.oxygenxml.pdf.css/css/print/i18n` directory) into your CSS customization (other than the English one because it does not have the `:lang` pseudo-class since it is the default language).

2. For each rules, replace the `:lang(xx)` pseudo-class with your expected language code, then replace each property value with the expected label.

---

**Related information**

Debugging the CSS *(on page 1816)*

---

**How to Modify Existing Strings**

If the label you want to modify is not available from the CSS, you need to modify the XML strings. The default XML strings are available at the following two locations:

- `DITA-OT-DIR/plugins/org.dita.base/xsl/common`
- `DITA-OT-DIR/plugins/com.oxygenxml.pdf.css/resources/localization`

To modify the generated text, you need to create a DITA-OT extension plugin that uses the `dita.xsl.strings` extension point. The following example uses English, but you can adapt it for any language:

1. In the `DITA-OT-DIR\plugins\` folder, create a folder for this plugin (for example, `com.oxygenxml.pdf.css.localization`).

2. Create a `plugin.xml` file (in the folder you created in step 1) that specifies the extension points, your parameter file, and your customization stylesheet. For example:

   ```xml
   <plugin id="com.oxygenxml.pdf.css.localization">
     <require plugin="com.oxygenxml.pdf.css"/>

     <feature extension="dita.xsl.strings" file="pdf-extension-strings.xml"/>
   </plugin>
   ```

3. Create a `pdf-extension-strings.xml` file with the following content:

   ```xml
   <langlist>
     <lang xml:lang="en" filename="strings-en-us.xml"/>
     <lang xml:lang="en-us" filename="strings-en-us.xml"/>
   </langlist>
   ```

4. Copy the strings you want to change from the default files to the `strings-en-us.xml` file, then replace their values:

   ```xml
   <strings xml:lang="en-US">
     <str name="Figure">Fig</str>
   </strings>
   ```
Warning:
Make sure the string @name attribute remains the same, it is used by the process as a key to retrieve the strings text.

5. Use the Run DITA-OT Integrator transformation scenario (on page 1455) found in the DITA Map section in the Configure Transformation Scenario(s) dialog box.

6. Run the DITA Map PDF - based on HTML5 & CSS transformation scenario.

How to Add New Strings

Some strings are not translated in all languages. In this case, they will appear in English. To add a new language for a given string, you need to create a DITA-OT extension plugin that uses the dita.xsl.strings extension point. The following example uses Polish, but you can adapt it for any language:

1. In the DITA-OT-DIR\plugins\ folder, create a folder for this plugin (for example, com.oxygenxml.pdf.css.localization).

2. Create a plugin.xml file (in the folder you created in step 1) that specifies the extension points, your parameter file, and your customization stylesheet. For example:

   ```xml
   <plugin id="com.oxygenxml.pdf.css.localization">
     <require plugin="com.oxygenxml.pdf.css"/>
     <feature extension="dita.xsl.strings" file="pdf-extension-strings.xml"/>
   </plugin>
   
   <langlist>
     <lang xml:lang="pl" filename="strings-pl-pl.xml"/>
     <lang xml:lang="pl-pl" filename="strings-pl-pl.xml"/>
   </langlist>
   
   4. Copy the strings you want to change from the default files to the strings-pl-pl.xml file, then replace their values:

   ```xml
   <strings xml:lang="pl-PL">
     <str name="Continued">ciąg dalszy</str>
   </strings>
   ```

   Warning:
   Make sure the string @name attribute remains the same, it is used by the process as a key to retrieve the strings text.
5. Use the **Run DITA-OT Integrator** transformation scenario *(on page 1455)* found in the **DITA Map** section in the **Configure Transformation Scenario(s)** dialog box.

6. Run the **DITA Map PDF - based on HTML5 & CSS** transformation scenario.

### Security

You can restrict the use of the PDF files by specifying a set of permissions. For example, you may want to set a password on the document, restrict the available actions inside the PDF reader, or encrypt the PDF content.

The `pdf.security.*` parameters listed in the **Transformation Parameters (on page 1773)** section can be used for this purpose.

#### How to Protect PDF Files by Setting Security Permissions

For example, to permit only the users that have a password to access the document and also to restrict their printing and copying capability, you can use the following parameter combination:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>pdf.security.owner.password</code></td>
<td><code>&lt;OWNER PASSWORD&gt;</code></td>
<td>People using this password will be able to open the document, with full permissions.</td>
</tr>
<tr>
<td><code>pdf.security.user.password</code></td>
<td><code>&lt;USER PASSWORD&gt;</code></td>
<td>People using this password will be able to open the document, but they will not be able to print.</td>
</tr>
<tr>
<td><code>pdf.security.restrict.print</code></td>
<td>yes</td>
<td>Restricts users from printing.</td>
</tr>
<tr>
<td><code>pdf.security.restrict.copy</code></td>
<td>yes</td>
<td>Restricts users from copying content.</td>
</tr>
</tbody>
</table>

**Important:**

If you specify just the user password (without an owner password), then the people using it will be considered owners, and no restrictions will apply to them.

### Troubleshooting

This section contains information about fixing various change tracking, highlights and comments publishing problems.

#### PDF file is damaged

**Problem**

It is possible to get a PDF that cannot be opened in the PDF viewer. In this case, you might get an error similar to:
Error: PDF file is damaged - attempting to reconstruct xref table...
Error: Couldn’t find trailer dictionary
Error: Couldn’t read xref table

**Cause**

This usually means that your PDF viewer does not support a PDF version greater than 1.4. The main difference with newer PDF versions is that the xref table is compressed in a stream and is not available as a table.

**Solution**

You need to re-run the PDF transformation with the `pdf.version` parameter set to 1.4.

**Failed to Run Pipeline: The Entity Cannot Be Resolved Through Catalogs**

**Problem**

You can get a *Failed to run pipeline* error message that looks something like this:

```
Failed to run pipeline: The entity SOME_ENTITY cannot be resolved through catalogs.
For security reasons files that are not listed in the DITA-OT catalogs and are not located in the DITA-OT directory are not read
```

**Cause**

This happens when the security checks that are implemented in the default transformation have blocked the reading of files that are not part of the DITA-OT (*Oxygen Publishing Engine*) installation directory and not part of the transformed DITA map.

**Solution**

If the origin of the transformed content is known and trusted, you can disable these checks by setting the `args.disable.security.checks` transformation parameter to *yes*.

**The format-date() XPath Function Does not Respect the Specified Locale**

**Problem**

Formatting a date using another language code, as in this example:

```
title:before {
    content: oxy_xpath('format-date(current-date(), "[Mn] [Y]", "ru", (), ())');
}
```

results in an output like: *[Language: en]*september 2019, with the date being formatted in English.

**Cause**

The XPath expressions are evaluated using the Saxon HE processor. This processor does not support languages other than English.
Solution

As a solution, you can either switch to a more language-neutral format that avoids the months names:

```xml
<template name="title-before">
  <content>
    oxy_xpath('format-date( current-date(), "[M] [Y]", "en", (), ())');
  </content>
</template>
```

or you can use a more complex XPath expression like this:

```xml
<template name="title-before">
  <content>
    oxy_xpath("let $cm:= format-date(current-date(), '[MNn]') \n    return concat( \n      if ($cm= 'January') then 'JAN' else \n      if ($cm= 'February') then 'FEB' else \n      if ($cm= 'March') then 'MAR' else \n      if ($cm= 'April') then 'APR' else \n      if ($cm= 'May') then 'MAY' else \n      if ($cm= 'June') then 'JUNE' else \n      if ($cm= 'July') then 'JUL' else \n      if ($cm= 'August') then 'AUG' else \n      if ($cm= 'September') then 'SEPT' else \n      if ($cm= 'October') then 'OCT' else \n      if ($cm= 'November') then 'NOV' else '' \n    , ', ' , \n    format-date(current-date(), '[Y0001]') \n  ");
</content>
</template>
```
Make sure the entire expression is rendered blue in the CSS editor. Replace the capitalized month names with the translation in the desired language.

**Error Parsing CSS File - Caused by a Networking Problem**

**Problem**

My custom styles are not applied and in the transformation results console, I get an error containing one of the following: I/O exception, Unknown host, Error parsing.

**Cause**

One of the CSS files contains references to resources from another website that is currently inaccessible. These resources may include:

- Fonts
- Images
- Other CSS files

**Note:**

If you exported one of the built-in publishing templates from the transformation scenario dialog, it is possible that the associated CSS files use an imported Google Font.

**Remedy**

1. Check your proxy settings (ask the system administrator for help).
2. If the server is still inaccessible from the transformation process, download the remote resources using a web browser, save them in the customization CSS file folder, and refer them directly from your CSS.

**Note:**

If the problem is caused by a remote font, see [Using Local Fonts](#).

**Glossary Entries Referenced Using ‘glossref’ are not Displayed**

**Problem**

I have a `<glossgroup>` that contains multiple `<glossentry>` elements and all the entries are referenced using `<glossref>` elements inside my map. When I add an `<abbreviated-form>` element linked to one of my `<glossentry>` elements (using a `@keyref`), the entry is not resolved in the PDF output.
Solution

Make sure every `<glossentry>` has an `@id`. Then, for each `<glossentry>`, declare a `<glossref>` element like this:

```
<glossref href="concepts/glossary.dita#flowers.genus" print="yes" keys="genus"/>
```

⚠️ Important:

For bookmaps, the `<glossref>` elements should be declared in a separate ditamap.

How to Fix Disappearing Thin Lines or Cell Borders

Problem

There are cases where thin lines disappear from the PDF viewer at certain zoom levels.

Cause

This is caused by the limited resolution of the display, while a printer has a superior resolution and there should be no problem printing thin lines on paper.

Solution

If the primary PDF target is the display, then you have to use thicker lines in your CSS customization (for example, avoid using 1px and use 1pt or larger instead).

If you are using Adobe Acrobat Reader, then you can enhance the display of thin lines. This behavior can be changed by going to Edit > Preferences > Page Display > Enhance Thin Lines. Deselecting this option makes thin lines displayed as a row of gray pixels (through antialiasing) and they do not disappear. You can experiment by selecting and deselecting the option.

XSL FO-based DITA to PDF Customization

Oxygen XML Editor comes bundled with the DITA Open Toolkit that provides a mechanism for converting DITA maps (on page 3218) to PDF output. Oxygen XML Editor includes a built-in DITA Map PDF - based on XSL-FO transformation scenario (on page 1451) that converts DITA maps to PDF using an `xsl:fo` processor.

There are several methods that can be used to customize DITA to PDF output:

- Create a customization directory that contains your customized files and reference that directory in the PDF transformation scenario (using the `customization.dir` parameter).
- Creating a DITA Open Toolkit plugin that adds extensions to the PDF output. More details can be found in the DITA Open Toolkit Documentation.

Tip:

Some sample plugins are available on GitHub that could help you to get started with creating a plugin:
Using a Customization Directory

One way to customize the PDF output generated by the DITA Map PDF - based on XSL-FO transformation scenario (on page 1451) is to create a dedicated folder to store customized files. With this approach, you will copy the contents of the built-in customization directory to a new directory where you can customize the files according to your needs and reference the new directory using the customization.dir parameter in the transformation scenario. The biggest advantage of this method is that the contents of your customization directory will remain unaffected when the DITA-OT is upgraded.

How to Create a Customization Directory

Follow this procedure to create a customization directory:

1. Copy all the entire DITA-OT-DIR\plugins\org.dita.pdf2\Customization directory to another location where you have write access.
2. Modify any of the files in whatever way necessary to achieve your specific goal. For inspiration, see Embedding a Company Logo (on page 1987) for a specific example of how you can modify contents of the directory to embed a logo in the output.

Tip:
For other specific examples, see DITA-OT Documentation - PDF Customization Plugin.

3. Edit the DITA Map PDF - based on XSL-FO transformation scenario (on page 1451), go to the Parameters tab, and set the customization.dir parameter to point to the location of your customization directory.

Related information
Automatic PDF plugin customization generator by Jarno Elovirta.
DITA-OT Documentation - PDF Customization Plugin

Embedding a Company Logo

The following procedure explains how to embed a company logo image in the front matter of the book.

1. Create a customization directory (on page 1987) (if you have not already done so).
2. Create a cfg\common\artwork directory structure in your customization directory and copy your logo to that directory (for example, [C:\Customization\common\artwork\logo.png]).

Important:
Make sure that your logo image is named: logo.png.
3. Rename `Customization\catalog.xml.orig` to `Customization\catalog.xml`.

4. Open the `catalog.xml` in Oxygen XML Editor and *uncomment* this line:

   ```xml
   <!--uri name="cfg:fo/xsl/custom.xsl" uri="fo/xsl/custom.xsl"/>-->
   ```

   It now looks like this:

   ```xml
   <uri name="cfg:fo/xsl/custom.xsl" uri="fo/xsl/custom.xsl"/>
   ```

5. Rename the file `Customization\fo\xsl\custom.xsl.orig` to `C:\Customization\fo\xsl\custom.xsl`

6. Open the `custom.xsl` file in Oxygen XML Editor and create the template called `createFrontCoverContents` for DITA-OT 3.7.0.

   **Tip:**
   You can copy the same template from `DITA-OT-DIR\plugins\org.dita.pdf2\xsl\fo\front-matter.xsl` and modify it in whatever way necessary to achieve your specific goal.
   
   This new template in the `custom.xsl` file will override the same template from `DITA-OT-DIR\plugins\org.dita.pdf2\xsl\fo\front-matter.xsl`.

**Example:**

For example, the `custom.xsl` could look like this:

```xml
<?xml version='1.0'?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    xmlns:fo="http://www.w3.org/1999/XSL/Format"
    version="2.0">

<xsl:template name="createFrontCoverContents">
    <!-- set the title -->
    <fo:block xsl:use-attribute-sets="__frontmatter__title">
        <xsl:choose>
            <xsl:when test="$map/*[contains(@class,' topic/title ')][1]">
                <xsl:apply-templates select="$map/*[contains(@class,' topic/title ')][1]"/>
            </xsl:when>
            <xsl:when test="$map//*[contains(@class,' bookmap/mainbooktitle ')][1]">
                <xsl:apply-templates select="$map//*[contains(@class,' bookmap/mainbooktitle ')][1]"/>
            </xsl:when>
            <xsl:when test="/*[contains(@class, ' map/map ')]/@title">
                <xsl:value-of select="/*[contains(@class, ' map/map ')]/@title"/>
            </xsl:when>
            <xsl:otherwise>
```

```
7. Edit the DITA Map PDF - based on XSL-FO transformation scenario (on page 1451), go to the Parameters tab, and set the customization.dir parameter to point to the location of your customization directory.

Tip:
For other specific examples, see DITA-OT 3.x Documentation - Customizing PDF Output.

Related Information:
Using a Customization Directory (on page 1987)

Customizing the Header and Footer in PDF Output

The XSLT stylesheet DITA-OT-DIR/plugins/org.dita.pdf2/xhtml/xsl/fo/static-content.xsl contains templates that output the static header and footers for various parts of the PDF such as the prolog, table of contents, front matter, or body.

The templates for generating a footer for pages in the body are called insertBodyOddFooter or insertBodyEvenFooter.
These templates get the static content from resource files that depend on the language used for generating the PDF. The default resource file is `DITA-OT-DIR/plugins/org.dita.pdf2/cfg/common/vars/en.xml`. These resource files contain variables (such as `Body odd footer`) that can be set to specific user values.

Instead of modifying these resource files directly, they can be overwritten with modified versions of the resources in a PDF customization directory.

1. **Create a customization directory** *(on page 1987)* (if you have not already done so).
2. Locate the stylesheets and templates listed above in your customization directory and modify them in whatever way necessary to achieve your specific goal.

   **Tip:**
   For more information and examples, see the Oxygen PDF Customization Plugin project on GitHub.

3. Edit the **DITA Map PDF - based on XSL-FO** transformation scenario *(on page 1451)*, go to the **Parameters** tab, and set the **customization.dir** parameter to point to the location of your customization directory.

**Related Information:**

Using a Customization Directory *(on page 1987)*

**Adding a Watermark to PDF Output**

To add a watermark to the PDF output of a **DITA Map PDF - based on XSL-FO** transformation scenario *(on page 1451)*, follow this procedure:

1. **Create a customization directory** *(on page 1987)* (if you have not already done so).
2. Create a `cfg\common\artwork` directory structure in your customization directory and copy your watermark image to that directory (for example, `C:\Customization\cfg\common\artwork\watermark.png`).
3. Rename the `Customization\catalog.xml.orig` file to: `Customization\catalog.xml`.
4. Open the `catalog.xml` in Oxygen XML Editor and **uncomment** this line:
   ```xml
   <!--uri name="cfg:fo/xsl/custom.xsl" uri="fo/xsl/custom.xsl"/>-->
   ```

   The uncommented line should look like this:
   ```xml
   <uri name="cfg:fo/xsl/custom.xsl" uri="fo/xsl/custom.xsl"/>
   ```
5. Rename the file: `Customization\fo\xsl\custom.xsl.orig` to: `Customization\fo\xsl\custom.xsl`.
6. Open the `Customization\fo\xsl\custom.xsl` file in Oxygen XML Editor to overwrite two XSLT templates:

° The first template is located in the XSLT stylesheet `DITA-OT-DIR\plugins\org.dita.pdf2\xsl\fo\static-content.xsl`. Override by copying the original template content in the `custom.xsl` and specifying a watermark image for every page in the PDF content, using a `block-container` element that references the watermark image file:

```xml
<fo:static-content flow-name="odd-body-header">
  <fo:block-container absolute-position="absolute"
                        top="-2cm" left="-3cm" width="21cm" height="29.7cm"
                        background-image="{concat($artworkPrefix,'Configuration/OpenTopic/cfg/common/artwork/watermark.png')}">
    <fo:block/>
  </fo:block-container>
  <fo:block xsl:use-attribute-sets="__body_odd__header">
    <xsl:call-template name="insertVariable">
      <xsl:with-param name="theVariableID" select="'Body odd header'"/>
      <xsl:with-param name="theParameters">
        <prodname>
          <xsl:value-of select="$productName"/>
        </prodname>
        <heading>
          <fo:inline xsl:use-attribute-sets="__body_odd__header__heading">
            <fo:retrieve-marker retrieve-class-name="current-header"/>
          </fo:inline>
        </heading>
        <pagenum>
          <fo:inline xsl:use-attribute-sets="__body_odd__header__pagenum">
            <fo:page-number/>
          </fo:inline>
        </pagenum>
      </xsl:with-param>
    </xsl:call-template>
  </fo:block>
</fo:static-content>
```

° The second template to override is located in the XSLT stylesheet `DITA-OT-DIR\plugins\org.dita.pdf2\xsl\fo\commons.xsl` and is used for styling the first page of the output. Override it by copying the original template content in the `custom.xsl` and adding the `block-container` element that references the watermark image file:
<xsl:template name="createFrontMatter_1.0">
  <fo:page-sequence master-reference="front-matter"
xsl:use-attribute-sets="__force_page_count">
    <xsl:call-template name="insertFrontMatterStaticContents"/>
    <fo:flow flow-name="xsl-region-body">
      <xsl:call-template name="insertFrontMatterStaticContents"/>
      <fo:block-container absolute-position="absolute"
        top="-2cm" left="-3cm" width="21cm" height="29.7cm"
        background-image="{concat($artworkPrefix,
          'Configuration/OpenTopic/cfg/common/artwork/watermark.png')}">
        <fo:block/>
      </fo:block-container>
      <fo:block xsl:use-attribute-sets="__frontmatter">
        <!-- set the title -->
        <fo:block xsl:use-attribute-sets="__frontmatter__title">
          <xsl:choose>
            <xsl:when test="$map/*[contains(@class,' topic/title ')][1]">
              <xsl:apply-templates select="$map/*[contains(@class,' topic/title ')][1]"/>
            </xsl:when>
            <xsl:when test="$map//*[contains(@class,' bookmap/mainbooktitle ')][1]">
              <xsl:apply-templates select="$map//*[contains(@class,' bookmap/mainbooktitle ')][1]"/>
            </xsl:when>
            <xsl:when test="//*[contains(@class, 'map/map')]/@title">
              <xsl:value-of select="//*[contains(@class, 'map/map')]/@title"/>
            </xsl:when>
            <xsl:otherwise>
              <xsl:value-of select="/descendant::*[contains(@class, 'topic/topic')][1]/*[
                contains(@class, 'topic/title')]/"/>
            </xsl:otherwise>
          </xsl:choose>
        </fo:block>
        <!-- set the subtitle -->
        <xsl:apply-templates select="$map//*[contains(@class,' bookmap/booktitlealt ')]"/>
        <fo:block xsl:use-attribute-sets="__frontmatter__owner">
          <xsl:apply-templates select="$map//*[contains(@class,' bookmap/bookmeta ')]"/>
        </fo:block>
      </fo:block>
    </fo:flow>
  </fo:page-sequence>
</xsl:template>
7. Edit the DITA Map PDF - based on XSL-FO transformation scenario (on page 1451), go to the Parameters tab, and set the customization.dir parameter to point to the location of your customization directory.

Related Information:
Adding a Watermark in DITA Map to XHTML Output (on page 3122)

Adding an Edit Link in PDF Output to Launch Oxygen XML Web Author

You can embed Edit links in the DITA Map PDF output that will automatically launch a particular document in Oxygen XML Web Author. A reviewer can then simply click the link and they will be redirected to the Oxygen XML Web Author editing page with that particular file open and editable.

To embed an Edit link in the DITA Map PDF output, follow these steps:

1. Edit a DITA Map PDF - based on XSL-FO transformation scenario (on page 1451) and open the Parameters tab.
2. Set values for the following parameters:
   - editlink.ditamap.edit.url - The URL of the DITA map used to publish your content. The easiest way to obtain the URL is to open the map in Web Author and copy the URL from the browser's address bar.
   - editlink.additional.query.parameters - Optional query parameters to be appended to each generated edit link. Each parameter must start with & (e.g. &tags-mode=no-tags).
3. Run the transformation scenario.

Result: In the PDF output, all topics will have an Edit link to the right side of the title and clicking the link will launch that particular document in Oxygen XML Web Author.

Force Page Breaks Between Two Block Elements in PDF Output

Suppose that in your DITA content you have two block elements (on page 3216), such as two paragraphs:

```xml
<p>First para</p>
<p>Second para</p>
```
and you want to force a page break between them in the PDF output.

Here is how you can implement a DITA Open Toolkit plugin (on page 3222) that would achieve this:

1. Define your custom processing instruction that marks the place where a page break should be inserted in the PDF, for example:

   `<p>First para</p>
   <?pagebreak?>
   <p>Second para</p>`

2. Locate the DITA Open Toolkit distribution and in the plugins directory create a new plugin folder (for example, DITA-OT-DIR/plugins/pdf-page-break).

3. In this new folder, create a new plugin.xml file with the following content:

   ```xml
   <plugin id="com.yourpackage.pagebreak">
     <feature extension="package.support.name" value="Force Page Break Plugin"/>
     <feature extension="package.support.email" value="support@youremail.com"/>
     <feature extension="package.version" value="1.0.0"/>
     <feature extension="dita.xsl.xslfo" value="pageBreak.xsl" type="file"/>
   </plugin>
   ```

   The most important feature in the plugin is that it will add a new XSLT stylesheet to the XSL processing that produces the PDF content.

4. In the same folder, create an XSLT stylesheet named pageBreak.xsl with the following content:

   ```xml
   <xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
                   xmlns:fo="http://www.w3.org/1999/XSL/Format" version="1.0">
     <xsl:template match="processing-instruction('pagebreak')">
       <fo:block break-after="page"/>
     </xsl:template>
   </xsl:stylesheet>
   ```

5. Install your plugin in the DITA Open Toolkit. (on page 3164)

   The source code for the plugin can be found on GitHub here: https://github.com/dita-community/org.dita-community.pdf-page-break.

Show Comments and Tracked Changes in PDF Output

To include comments and tracked changes (stored within your DITA topics) in the PDF output, follow these steps:

1. Edit a DITA Map PDF - based on XSL-FO transformation scenario.
2. In the Parameters tab, set the value of the show.changes.and.comments parameter to yes. If you also want to display change bars for inserted or deleted content in the PDF, set the show.changebars parameter to yes. If you want to only show the changebars, without
other styling of the changes, you should set both the `show.changes.and.comments` and `show.changes.and.comments.as.changebars.only` parameters to `yes`.

3. Optionally, you can configure any of these other parameters to adjust the colors of the comments and tracked changes:
   - `ct.insert.color` - Specifies the color for insertion type tracked changes, as a plain color (e.g. red, yellow, blue), or with a hexadecimal equivalent (e.g. `#FFFFFF`). The default value is 'blue'.
   - `ct.delete.color` - Specifies the color for deletion type tracked changes, as a plain color (e.g. red, yellow, blue), or with a hexadecimal equivalent (e.g. `#FFFFFF`). The default value is 'red'.
   - `ct.comment.bg.color` - Specifies the background color for comment type tracked changes, as a plain color (e.g. red, yellow, blue), or with a hexadecimal equivalent (e.g. `#FFFFFF`). The default value is 'yellow'.

4. Click **OK** and then the **Apply Associated** button to run the transformation scenario.

**Result:** Comment threads and tracked changes will now appear in the PDF output. Details about each comment or change will be available in the footer section for each page.

### Set a Font for PDF Output Generated with FO Processor

When a DITA map (on page 3218) is transformed to PDF using an FO processor and it contains some Unicode characters that cannot be rendered by the default PDF fonts, a font that is capable of rendering these characters must be configured and embedded in the PDF result.

The settings that must be modified for configuring a font for the built-in FO processor are detailed in *Add a Font to the Built-in FO Processor - Advanced Version (on page 1528).*

### DITA-OT PDF Font Mapping

The DITA-OT contains a file `DITA-OT-DIR/plugins/org.dita.pdf2/cfg/fo/font-mappings.xml` that maps logical fonts used in the XSLT stylesheets to physical fonts that will be used by the FO processor to generate the PDF output.

The XSLT stylesheets used to generate the XSL-FO output contain code like this:

```xml
<xsl:attribute name="font-family">monospace</xsl:attribute>
```

The font-family is defined to be `monospace`, but `monospace` is just an alias. It is not a physical font name. Therefore, another stage in the PDF generation takes this `monospace` alias and looks in the `font-mappings.xml`.

If it finds a mapping like this:

```xml
<aliases>
  <alias name="monospace">Monospaced</alias>
</aliases>
```

then it looks to see if the `monospace` has a `logical-font` definition and if so, it will use the `physical-font` specified there:
Adding Libraries to the Built-in FO Processor (DITA-OT)

Starting with Oxygen XML Editor version 20.0, both hyphenation and PDF image support are enabled by default in the built-in DITA-OT FO processor. For older version of Oxygen XML Editor, use the following procedures to enable such support.

Adding Hyphenation Support for DITA-OT Transformation Scenarios

1. Download the pre-compiled JAR (on page 3220) from OFFO.
2. Edit the DITA-OT transformation scenario and switch to the Advanced tab.
3. Click the Libraries button and add the path to the fop-hyph.jar library.

Adding Support for PDF Images

1. Download the fop-pdf-images JAR libraries.
2. Edit the DITA-OT transformation scenario and switch to the Advanced tab.
3. Click the Libraries button and add the path to the libraries.

Adding Support for CGM Images

1. Go to the JCGM page and download the jcgm-image-0.1.1.jar and jcgm-core-0.2.0.jar libraries.
2. Edit the DITA-OT transformation scenario and switch to the Advanced tab.
3. Click the Libraries button and add the path to the libraries.

Debugging DITA PDF Transformations

To debug a DITA PDF transformation scenario, follow these steps:
1. Open the Preferences dialog box (Options > Preferences) (on page 124), go to XML > XML Catalog, click Add, and select the file located at \DITA-OT-DIR\plugins\org.dita.pdf2\cfg\catalog.xml.

2. Open the map in the DITA Maps Manager (on page 2897) and create a DITA Map PDF - based on XSL-FO transformation scenario.

3. Edit the scenario, go to the Parameters tab and change the value of the clean.temp parameter to no.

4. Run the transformation scenario.

5. Open the stage1.xml file located in the temporary directory and format and indent (on page 555) it.

6. Create a transformation scenario for this XML file by associating the topic2fo_shell_fop.xsl stylesheet located at \DITA-OT-DIR\plugins\org.dita.pdf2\xsl\fo\topic2fo_shell_fop.xsl. If you are specifically using the RenderX XEP or Antenna House FO processors to build the PDF output, you should use the XSL stylesheets topic2fo_shell_xep.xsl or topic2fo_shell_axf.xsl located in the same folder.

Note: For validation purposes, you need to add the main debugged stylesheet (usually topic2fo_shell_fop.xsl) to the Main Files folder (on page 421) in the Project view.

7. In the transformer drop-down menu, select the Saxon EE XSLT processor (the same processor used when the DITA-OT transformation is executed).

8. Click the Parameters button and set the locale parameter with the value en_GB and the customizationDir.url parameter to point either to your customization directory or to the default DITA-OT customization directory. Its value should have a URL syntax like this: file://c:/path/to/DITA-OT-DIR/plugins/org.dita.pdf2/config.

9. If your XSLT stylesheet uses Java extensions, you need to reference the extra JAR libraries by clicking the Extensions button and add the libraries in the resulting dialog box. For example, if you have enabled the show.changes.and.comments parameter, you need to add the following JAR library for the parameter to have an effect: DITA-OT\plugins\com.oxygenxml.common\lib\oxygen-dita-publishing-xslt-extensions.jar.

10. Apply the transformation to continue the debugging process.

Related Information:
- Debugging XSLT Stylesheets and XQuery Documents (on page 2102)
- How to Enable Debugging for FO Processor Transformations (on page 1529)

DocBook to PDF Output Customization

When the default layout and output of the DocBook to PDF transformation needs to be customized, follow these steps:
   
   You could start from a copy of the file [DocBook XSL directory]/fo/titlepage.templates.xml (for example, [OXYGEN-INSTALL-DIR]/frameworks/docbook/xsl/fo/titlepage.templates.xml) and customize it. More information about the spec file can be found here.

2. Generate a new XSLT stylesheet from the title spec file from the previous step.
   
   Apply [DocBook XSL directory]/template/titlepage.xsl to the title spec file. The result is an XSLT stylesheet (for example, mytitlepages.xsl).

3. Import mytitlepages.xsl in a DocBook customization layer.
   
   The customization layer is the stylesheet that will be applied to the XML document. The mytitlepages.xsl should be imported with an element like this:

   ```xml
   <xsl:import href="dir-name/mytitlepages.xsl"/>
   ```

4. Insert a logo image in the XML document.
   
   The path to the logo image must be inserted in the book/info/mediaobject structure of the XML document.

5. Apply the customization layer to the XML document.
   
   A quick way is to duplicate the transformation scenario DocBook PDF that is included with Oxygen XML Editor and set the customization layer in the XSL URL property of the scenario (on page 1464).

Related Information:


Video demonstration for creating a DocBook customization layer in Oxygen XML Editor.
Working with XPath Expressions

XPath is a language for addressing specific parts of a document. XPath models an XML document as a tree of nodes. An XPath expression is a mechanism for navigating through and selecting nodes from the document. An XPath expression is, in a way, analogous to an SQL query used to select records from a database.

There are various types of nodes, including element nodes, attribute nodes, and text nodes. XPath defines a way to compute a string-value for each type of node.

XPath defines a library of standard functions for working with strings, numbers and boolean expressions.

Examples:

- `child::*` - Selects all children of the root node.
- `//name` - Selects all `<name>` elements and descendants of the current node.
- `/catalog/cd[price>10.80]` - Selects all the `<cd>` elements that have a `<price>` element with a value larger than 10.80.
- `//prolog` - Finds all `<prolog>` elements.
- `//prolog[@platform='mac']` - Finds all `<prolog>` elements that have the `@platform` attribute value set to `mac`.
- `//child::prolog` - Selects all `<prolog>` elements and the child content.
- `/*[count(//accountNumber) > 5]` - Searches for instances where more than 5 `<accountNumber>` elements are found.
- `collection('file:/C:/path/to/folder/*?select=*.xml')/*[not(//prolog)]` - Finds a list of all XML files that do not contain any `<prolog>` elements.

To find out more about XPath, see [http://www.w3.org/TR/xpath](http://www.w3.org/TR/xpath).

Related Information:

- Content Completion in XPath Expressions (on page 894)
- Find/Replace in Multiple Files (on page 436)
- Find/Replace Dialog Box (on page 432)
XPath Toolbar

XPath is a query language for selecting nodes from an XML document. To use XPath expressions effectively, you need a good understanding of the XPath Core Function Library.

XPath Toolbar

Oxygen XML Editor provides an XPath toolbar to let you query XML documents fast and easy using XPath expressions.

Figure 512. XPath Toolbar

The XPath toolbar includes the following features:

XPath version chooser drop-down menu

You can choose the XPath version from the drop-down menu available in the left side of the toolbar. Available options include XPath 1.0, XPath 2.0, XPath 2.0 SA, XPath 3.1, XPath 3.1 SA.

Note:
The XPath 2.0 SA and XPath 3.1 SA options have some limitations. These options only offer information about the beginning part of the matching result. For example, if you search for an element, it will only highlight the start tag.

Warning:
Oxygen XML Editor uses Saxon to execute XPath 3.1 expressions, but implements a part of the 3.1 functions. When using a function that is not implemented, Oxygen XML Editor can return a compilation error.

XPath scope menu

Oxygen XML Editor allows you to define a scope for the XPath operation to be executed. You can choose where the XPath expression will be executed:

- Current file - Currently selected file only.
- Project - All the files in the project.
- Selected project resources - The files selected in the project.
- All opened files - All files that are opened in the application.
- Current DITA Map hierarchy - All resources referenced in the currently selected DITA map that is open in the DITA Maps Manager view (on page 2897).
- Opened archive - Files that are opened in the Archive Browser view (on page 2008).
- Working sets - The selected working sets (on page 3224).
At the bottom of the scope menu the following scope configuration actions are available:

- **Configure XPath working sets** - Allows you to configure and manage collections of files and folders, encapsulated in logical containers called *working sets (on page 3224)*.

- **XPath file filter** - You can filter the files from the selected scope that will have the XPath expression executed. By default, the XPath expression will be executed only on XML or JSON files, but you can also define a set of patterns that will filter out files from the current scope. If you select the *Include archive* option, the XPath expression will be also executed on the files in any archive (including EPUB and DocX) found at the current scope.

**History drop-down list**

The XPath combo box keeps a history of the last 15 expressions that were used so that you can easily choose them again.

**Settings menu**

The following actions are available in this drop-down menu:

- **XPath update on cursor move**
  When selected and you navigate through a document, the XPath expression corresponding to the XML node at the current cursor position is displayed. For JSON documents, it displays the XPath expression for the current property.

- **Evaluate XPath as you type**
  When you select this option, the XPath expression you are composing is evaluated in real time.

---

**Note:**

This option and the automatic validation are disabled when you edit huge documents *(on page 469)* or when the scope is other than *Current file*.

---

**XPath Options**

Opens the Preferences page of the currently selected processing engine.

**Switch to XPath Builder View**

Opens the XPath Builder view *(on page 2002).*

---

**Note:**

During the execution of an XPath expression, the XPath toolbar displays a **Stop** button. Use this button to stop the XPath execution.
When you type expressions longer than 60 characters, a dialog box opens that offers you the possibility to switch to the **XPath Builder view** (on page 2002).

**Related Information:**
- XPath Expression Results View (on page 2005)

---

**XPath Builder View**

The **XPath/XQuery Builder** view allows you to compose complex XPath expressions and execute them over the currently edited XML document. For XPath 2.0 / 3.1, you can use the `doc()` function to specify the source file that will have the expressions executed. When you connect to a database, the expressions are executed over that database. If you are using the **XPath/XQuery Builder** view and the current file is an XSLT document, Oxygen XML Editor executes the expressions over the XML document in the associated scenario.

**Note:**

If an XPath expression is run over a JSON document, it is converted to XML and the XPath is executed over the converted XML document.

If the view is not displayed, it can be opened by selecting it from the **Window > Show View** menu. You can also open it simply by pressing the **Switch to XPath Builder View** button that is located on the **XPath toolbar** (on page 2000).

The upper part of the view contains the following actions:

**XPath version chooser drop-down menu**

A drop-down menu that allows you to select the type of the expression you want to execute. You can choose between:

- XPath 1.0 (Xerces-driven)
- XPath 2.0, XPath 2.0 SA, XPath 3.1, XPath 3.1 SA, Saxon-HE XQuery, Saxon-PE XQuery, or Saxon-EE XQuery (all of them are Saxon-driven)
- Custom connection to XML databases that can execute XQuery expressions

**Note:**

The results returned by XPath 2.0 SA and XPath 3.1 SA have a location limited to the line number of the start element (there are no column information and no end specified).

**Note:**

Oxygen XML Editor uses Saxon to execute XPath 3.1 expressions. Since Saxon implements a part of the 3.1 functions, when using a function that is not implemented, Oxygen XML Editor returns a compilation error.
Execute XPath button

Use this button to start the execution of the XPath or XQuery expression you are editing. The result of the execution is displayed in the Results view (on page 548).

Favorites button

Allows you to save certain expressions that you can later reuse. To add an expression as a favorite, click this button and enter a name for it. The star turns yellow to confirm that the expression was saved. Expand the drop-down menu next to the star button to see all your favorites. Oxygen XML Editor automatically groups favorites in folders named after the method of execution.

History drop-down menu

Keeps a list of the last 15 executed XPath expressions. Use the Clear history action from the bottom of the list to remove them.

Settings drop-down menu

Contains the following three options:

Update on cursor move

When selected and you navigate through a document, the XPath expression corresponding to the XML node at the current cursor position is displayed. For JSON documents, it displays the XPath expression for the current property.

Evaluate as you type

When you select this option, the XPath expression you are composing is evaluated in real time.

Note:

This option and the automatic validation are disabled when you edit huge documents (on page 469) or when the scope is other than Current file.

Options

Opens the Preferences page of the currently selected processing engine.

XPath scope menu

Oxygen XML Editor allows you to define a scope for the XPath operation to be executed. You can choose where the XPath expression will be executed:

- **Current file** - Currently selected file only.
- **Project** - All the files in the project.
- **Selected project resources** - The files selected in the project.
- **All opened files** - All files that are opened in the application.
- **Current DITA Map hierarchy** - All resources referenced in the currently selected DITA map that is open in the DITA Maps Manager view (on page 2897).

- **Opened archive** - Files that are opened in the Archive Browser view (on page 2008).

- **Working sets** - The selected working sets (on page 3224).

At the bottom of the scope menu the following scope configuration actions are available:

- **Configure XPath working sets** - Allows you to configure and manage collections of files and folders, encapsulated in logical containers called working sets (on page 3224).

- **XPath file filter** - You can filter the files from the selected scope that will have the XPath expression executed. By default, the XPath expression will be executed only on XML or JSON files, but you can also define a set of patterns that will filter out files from the current scope. If you select the **Include archive** option, the XPath expression will be also executed on the files in any archive (including EPUB and DocX) found at the current scope.

Figure 513. XPath/XQuery Builder View
While you edit an XPath or XQuery expression, Oxygen XML Editor assists you with the following features:

- **Content Completion Assistant (on page 3217)** - It offers context-dependent proposals and takes into account the cursor position in the document you are editing. The set of functions proposed by the Content Completion Assistant also depends on the engine version. Select the engine version from the drop-down menu available in the toolbar.
- **Syntax Highlighting** - Allows you to identify the components of an expression. To customize the colors of the components of the expression, open the Preferences dialog box (Options > Preferences) (on page 124) and go to Editor > Syntax Highlight (on page 224).
- **Automatic validation of the expression as you type.**
- **Function signature and documentation balloon**, when the cursor is located inside a function.

The usual edit actions (Trim, Copy, Paste, Select All, Cut, Paste, Select All, Cut, Copy, Paste, Select All, Cut) are available in the contextual menu of the top editable part of the view.

**Note:**
When you type invalid syntax, a red serrated line underlines the invalid fragments.

The Results view also includes various toolbar and contextual menu actions. For more information, see Results View (on page 548).
Example:

The following snippets are taken from a DocBook book based on the DocBook XML DTD. The book contains a number of chapters. To return all the chapter nodes of the book, enter //chapter in the XPath expression field and press Enter. This action returns all the chapter nodes of the DocBook book in the Results View. Click a record in the Results View to locate and highlight its corresponding chapter element and all its children nodes in the document you are editing.

To find all example nodes contained in the sect2 nodes of a DocBook XML document, use the following XPath expression: //chapter/sect1/sect2/example. Oxygen XML Editor adds a result in the Results View for each example node found in any sect2 node.

For example, if the result of the above XPath expression is:

```
- /chapter[1]/sect1[3]/sect2[7]/example[1]
```

it means that in the edited file, the example node is located in the first chapter, third section level one, seventh section level 2.
XPath and XML Catalogs

The evaluation of the XPath expression tries to resolve the locations of documents referenced in the expression through *XML Catalogs* (on page 3225). These catalogs are configured in the *XML Catalog preferences* (on page 235) pages and the *XML Parser preferences* (on page 237).

**Example:**

As an example, consider the evaluation of the `collection(URIofCollection)` function (XPath 2.0). To resolve the references from the files returned by the `collection()` function with an *XML catalog*, specify the class name of the catalog-enabled parser for parsing these collection files. The class name is `ro.sync.xml.parser.CatalogEnabledXMLReader`. Specify it as it follows:

```xml
let $docs := collection(iri-to-uri(
  "file:///D:/temp/test/XQuery-catalog/mydocsdir?recurse=yes;select=*.xml;
  parser=ro.sync.xml.parser.CatalogEnabledXMLReader"))
```

**XPath Prefix Mapping**

To define default mappings between prefixes (that you can use in the *XPath toolbar* (on page 2000)) and namespace URIs go to the *XPath preferences page* (on page 258) and enter the mappings in the *Default prefix-namespace mappings* table. The same preferences panel allows you to configure the default namespace used in XPath 2.0 expressions.

**Important:**

If you define a default namespace, Oxygen XML Editor binds this namespace to the first free prefix from the list: `default`, `default1`, `default2`, and so on. For example, if you define the default namespace `xmlns="something"` and the prefix `default` is not associated with another namespace, you can match tags without prefix in an XPath expression typed in the XPath toolbar by using the prefix `default`. To find all the `<level>` elements when you define a default namespace in the root element, use this expression: `//default:level` in the XPath toolbar.
13. Working with Archives

Oxygen XML Editor includes a useful Archive Browser view (on page 2008) that offers the means to work with files directly from various types of archives (for example, opening and saving files directly in archives, or browsing and modifying archive structures). The archive support is available for all ZIP-type archives, including:

- ZIP archives
- EPUB books
- JAR archives (on page 3220)
- Office Open XML (OOXML) files
- Open Document Format (ODF) files
- IDML files (on page 3219)

You can transform, validate, and perform many other operations on files directly from an archive. For instance, you can transform, or validate files directly from OOXML or ODF packages, and the structure and content of the ZIP archives can be opened, edited, and saved, similar to any other ZIP archive browsing tool. Also, when browsing for a URL in various dialog boxes, you can use the Browse for archived file action to browse and select files from a particular archive.

For more information about working with an EPUB archive in Oxygen XML Editor, watch our video demonstration:

https://www.youtube.com/embed/OIGTNQwOCl8

Browsing Archives

Oxygen XML Editor includes a helper view called the Archive Browser that allows you to view the contents and structure of an archive, and it offers a variety of toolbar and contextual menu actions. If the view is not displayed, it can be opened by selecting it from the Window > Show View menu.

To open an archive in the Archive Browser view, use one of the following methods:

- Open an archive from the Project view (on page 402).
- Select an archive in one of the file chooser dialog boxes in Oxygen XML Editor (such as the Open dialog box).
- Drag an archive from a system file explorer and drop it in the Archives Browser view.

When displaying an archive, the Archive Browser view locks the archive file. It is then automatically unlocked when the Archive Browser view is closed.
Tip:
If a file is not recognized by Oxygen XML Editor as a supported archive type, you can add it in the Archive preferences page (on page 290).

Figure 515. Archive Browser

Archive Browser Toolbar Actions

The following actions are available on the Archive Browser toolbar:

- **Open Archive menu**
  Provides access to the Open Archive action that opens a new archive in the browser. If the extension is not known as an archive extension, you will be directed to the Archive preferences page (on page 290) to add a new extension. The submenu keeps a list of recently open archive files and a Clear history action that allows you to delete the list.

- **Close**
  Closes the browsed archive and unlocks the archive file.

- **Validate (available for EPUB archives only)**
  Checks the EPUB archive to see if its content and structure is valid.

- **New folder**
  Creates a folder as child of the selected folder in the browsed archive.

- **New file**
  Creates a file as child of the selected folder in the browsed archive.
Add files

Adds existing files as children of the selected folder in the browsed archive.

*(Note:)*

You can also add files in the archive by dragging them from the file browser or the *Project view (on page 402)* and dropping them in the *Archive Browser view.*

Delete

Deletes the selected resource in the browsed archive.

Open in System Application

Opens the selected resource in the default system application that is associated with that type of file.

Archive Options

Opens the *Archive preferences page (on page 290).*

**Archive Browser Contextual Menu Actions**

The following additional actions are available from the contextual menu for resources in the *Archive Browser view:*

Open

Opens a resource from the archive in the editor.

Extract

Extracts a resource from the archive in a specified folder.

New folder

Creates a folder as child of the selected folder in the browsed archive.

New file

Creates a file as child of the selected folder in the browsed archive.

Add files

Adds existing files as children of the selected folder in the browsed archive.

*(Note:)*

On macOS, the *Add files* action is also available and it allows you to add one file at a time.

Rename

Renames a resource in the archive.
Find/Replace in Files

Opens the Find/Replace in Files dialog box (on page 436) that allows you to search for and replace specific pieces of text inside the archive.

Cut

Cuts the selected archive resource.

Copy

Copies the selected archive resource.

Paste

Pastes a file or folder into the archive.

Delete

Removes a file or folder from archive.

Preview

Previews an image contained in the archive. See the Image Preview (on page 469) topic for more details.

Copy location

Copies the URL location of the selected resource.

Refresh

Refreshes the selected resource.

Properties

Shows the properties of the selected resource.

For more information, watch our video demonstration about working with an EPUB in the Archive Browser view:

https://www.youtube.com/embed/OIGTNQwOCl8

Working with Archive Files

Oxygen XML Editor includes support for working with various types of archives, including the following:

• **EPUB** - An e-book file format that can be used on many types of devices, such as smart phones, tablets, e-readers, or computers.

• **OOXML** - An XML-based file format for representing spreadsheets, charts, presentations, and word processing documents.

• **ODF** - An free and open-source XML-based file format for electronic office documents, such as spreadsheets, charts, presentations, and word processing documents.
When these types of files are opened in the Archive Browser view (on page 2008), their internal components are expanded:

- Document content (XHTML and image files).
- Packaging files.
- Container files.

**Figure 516. EPUB File Displayed in the Archive Browser View**

When an archive is expanded in the Archive Browser view (on page 2008), you can add or delete files that compose the archive structure. All changes made to the structure of an archive are saved immediately. You can open files from within the archive to edit them in the main editing pane and save changes (on page 2013) back to the archive. You can also use the Open in System Application action to open the archive in the default system application that is associated with that type of file.

**EPUB-Specific Validation**

When working with EPUB archives, the Archive Browser (on page 2008) includes a Validate action on the toolbar that checks the EPUB archive to make sure the structure and content are valid. Oxygen XML Editor uses the open-source EpubCheck validator to perform the validation. This validator detects many types of errors, including OCF container structure, OPF and OPS mark-up, as well as internal reference consistency.
For more information about working with an EPUB archive in Oxygen XML Editor, watch our video demonstration:

https://www.youtube.com/embed/OIGTNQwOCI8

Related Information:
The Archive Browser View (on page 2008)
EPUB Document Type (Framework) (on page 1426)

Creating an Archive
To create an archive from scratch, follow these steps:

1. Go to File > New or click New on the main toolbar.
2. Choose your particular type of archive template. For example, select one of the ODF, OOXML, or EPUB templates.
3. Click Create and choose the name and location of the file.
4. Click Save.

A skeleton archive is saved on disk and open in the Archive Browser view (on page 2008).

Tip:
Use toolbar and contextual menu actions to edit, add, and remove resources from the archive. For EPUB archives, you can use the Validate action to verify the integrity of the EPUB archive.

Editing and Saving Files Inside an Archive
You can open files directly from an archive in the Archive Browser view (on page 2008) and then edit them in the main editor pane. To open a file, simply double-click it or select Open from the contextual menu.

When saving the file back to the archive, you are prompted to choose if you want the application to make a backup copy of the archive before saving the new content. If you choose Never ask me again, you will not be asked again to make backup copies. You can re-enable the pop-up message from the Messages preferences page (on page 307).

Migrating Archives to DITA or TEI
Certain types of archives can be converted to DITA or TEI. For example, OOXML (Office Open XML) archive files with the DOCX file extension can be migrated to DITA or TEI.

To migrate DOCX files to DITA or TEI, follow these steps:

1. Open and expand the archive in the Archive Browser (on page 2008).
2. Open the document.xml file contained in the archive.
3. Run one of the following built-in transformation scenarios:
   a. **DOCX DITA** to migrate to DITA.
   b. **DOCX TEI P5** to migrate to TEI.
4. You may need to do some manual reconfiguring to map DOCX styles to DITA or TEI content.

**Tip:**
Oxygen XML Editor also includes a built-in transformation scenario called **ODT TEI P5** for converting ODF archive files with the ODT file extension to TEI and a similar process can be used to migrate ODT files to TEI.
Databases and SharePoint

Oxygen XML Editor provides support for connecting and integrating with various databases and Microsoft SharePoint. This section includes information about the database-related features in Oxygen XML Editor. It explains how to connect with the supported databases, presents the actions that are available for each type, and includes information about SharePoint integration.

Working with Databases

XML is a storage and interchange format for structured data and is supported by all major database systems. Oxygen XML Editor offers the means for managing the interaction with some of the most commonly used databases (both Relational and Native XML databases). Through this interaction, Oxygen XML Editor helps users with browsing, content editing, importing from databases, using XQuery with databases, SQL execution, and generating XML Schema from a database structure.

The types of connections that are supported in Oxygen XML Editor include:

- IBM DB2 (on page 2021)
- Microsoft SQL Server (on page 2025)
- Oracle Database (on page 2029)
- PostgreSQL (on page 2034)
- eXist (on page 2044)
- MarkLogic (on page 2049)
- MySQL (on page 2059)
- Generic JDBC (on page 2061)
- JDBC-ODBC (on page 2062)
- BaseX (on page 2063)
- WebDAV (on page 2068)
- Microsoft SharePoint (on page 2081)
- Berkeley DB XML (Deprecated) (on page 2037)

Related information

Integration with Microsoft SharePoint (on page 2081)

Data Source Explorer View

The Data Source Explorer view displays your database connections. If the view is not displayed, it can be opened by selecting it from the Window > Show View menu.
You can connect to a database simply by expanding the connection node (click the connection). The database structure can be expanded to resource level, or even all the way to column level for tables inside relational databases. Oxygen XML Editor supports multiple simultaneous database connections and the connection tree in the **Data Source Explorer** view provides an easy method for browsing them.

**Figure 517. Data Source Explorer View**

The objects (nodes) that are displayed in the **Data Source Explorer** view depend on the connection type and structure of the database. Various contextual menu actions are available for each hierarchical level and for some connections you can add or move resources in a container by simply dragging them from the **Project view** (on page 402), a file browsing application, or another database.

**Toolbar Actions**

The following actions are available in the toolbar of this view:

- **Filters**
  
  Opens the **Data Sources / Table Filters** preferences page (on page 280), allowing you to decide which table types are displayed in the **Data Source Explorer** view.

- **Configure Database Sources**
  
  Opens the **Data Sources** preferences page (on page 276) where you can configure both data sources and connections.

**Database-Specific Contextual Menu Actions**

Each specific type of database will also include its own specific contextual menu actions in the **Data Source Explorer** view. The actions depend on the type of database, the type of node, or the hierarchical level of the node where the contextual menu is invoked.
For more information on the specific actions that are available, see the topics in this section for each specific type of database.

Related Information:
Data Sources Preferences (on page 276)

Table Explorer View

Relational databases tables in the Data Source Explorer view (on page 2015) can be displayed and edited in the Table Explorer view by selecting the Edit action from the contextual menu of a Table node or by double-clicking one of its fields. To modify the content of a cell, double-click it and start typing. When editing is complete, Oxygen XML Editor attempts to update the database with the new cell content.

Figure 518. Table Explorer View

You can sort the content of a table by one of its columns by clicking its column header.

Note the following:

- The first column is an index (not part of the table structure).
- Every column header contains the field name and its data type.
- The primary key columns are marked with this symbol: ⚖.
- Multiple tables are presented in a tabbed manner.

For performance issues, you can set the maximum number of cells that are displayed in the Table Explorer view (using the Limit the number of cells option in the Data Sources Preferences page (on page 280)). If a table that has more cells than the value set in the options is displayed in the Table Explorer view, a warning dialog box informs you that the table is only partially shown.

You are notified if the value you have entered in a cell is not valid (and thus cannot be updated).
• If the content of the edited cell does not belong to the data type of the column, the cell is marked by a red square and remains in an editing state until a correct value is inserted. For example, in the following figure propID contains LONG values. If a character or string is inserted, the cell will look like this:

**Figure 519. Cell Containing an Invalid Value**

![Image of a cell containing an invalid value]

• If the constraints of the database are not met (for instance, primary key constraints), an information dialog box will appear, notifying you of the reason the database has not been updated. For example, in the table below, trying to set the second record in the primary key propID column to 8, results in a duplicate entry error since that value has already been used in the first record:

**Figure 520. Duplicate Entry for Primary Key**

![Image of a dialog box showing a duplicate entry error]

**Table Explorer Contextual Menu Actions**

Common editing actions (Cut, Copy, Paste, Select All, Undo, Redo) are available in the contextual menu of an edited cell.

The contextual menu, available on every cell in the Table Explorer view, also includes the following actions:

- **Set NULL**

  Sets the content of the cell to null. This action is not available for columns that cannot have a value of null.
<table>
<thead>
<tr>
<th>Action</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Insert row</strong></td>
<td>Inserts an empty row in the table.</td>
</tr>
<tr>
<td><strong>Duplicate row</strong></td>
<td>Makes a copy of the selected row and adds it in the Table Explorer view. Note that the new row will not be inserted in the database table until all conflicts are resolved.</td>
</tr>
<tr>
<td><strong>Commit row</strong></td>
<td>Commits the selected row.</td>
</tr>
<tr>
<td><strong>Delete row</strong></td>
<td>Deletes the selected row.</td>
</tr>
<tr>
<td><strong>Copy</strong></td>
<td>Copies the content of the cell.</td>
</tr>
<tr>
<td><strong>Paste</strong></td>
<td>Pastes copied content into the selected cell.</td>
</tr>
</tbody>
</table>

**Table Explorer Toolbar Actions**

The toolbar of the Table Explorer view also includes the following actions:

<table>
<thead>
<tr>
<th>Action</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Export to XML</strong></td>
<td>Opens the Export Criteria dialog box (a thorough description of this dialog box can be found in the Import from database (on page 2096) chapter).</td>
</tr>
<tr>
<td><strong>Refresh</strong></td>
<td>Performs a refresh for the sub-tree of the selected node.</td>
</tr>
<tr>
<td><strong>Insert row</strong></td>
<td>Inserts an empty row in the table.</td>
</tr>
<tr>
<td><strong>Duplicate row</strong></td>
<td>Makes a copy of the selected row and adds it in the Table Explorer view. Note that the new row will not be inserted in the database table until all conflicts are resolved.</td>
</tr>
<tr>
<td><strong>Commit row</strong></td>
<td>Commits the selected row.</td>
</tr>
<tr>
<td><strong>Delete row</strong></td>
<td>Deletes the selected row.</td>
</tr>
</tbody>
</table>

**Related Information:**

Data Source Explorer View (on page 2015)
Database Connection Support

Oxygen XML Editor offers support for a variety of Relational and Native XML database connections. The database drivers and connections for various types of database are configured in the Data Sources preferences page (on page 276) and once configured, the database connections can be viewed and managed in the Data Source Explorer view (on page 2015). Oxygen XML Editor also includes a Database perspective (on page 350) that helps you to manage databases.

The database support in Oxygen XML Editor offers a variety of capabilities, including:

- Browsing the structure of databases in the Data Source Explorer view (on page 2015).
- Viewing relational tables in the Table Explorer view (on page 2017).
- Executing SQL queries against databases.
- Calling stored procedures with input and output parameters.
- XQuery execution with databases.
- Exporting data from databases to XML.

Relational Database Support

Relational databases use a relational model and are based on tables linked by a common key. Oxygen XML Editor offers support for the most commonly used relational databases, including:

- IBM DB2
- Oracle 11g
- Microsoft SQL Server
- PostgreSQL
- MySQL

Oxygen XML Editor also offers generic support (table browsing and execution of SQL queries) for any JDBC-compliant database (for example, MariaDB).

Native XML Database Support

Native XML databases have an XML-based internal model and their fundamental unit of storage is XML. They use XML as an interface to specify documents as tree structured data that may contain unstructured text, but on disk the data is stored as optimized binary files. This makes query and retrieval processes faster. Oxygen XML Editor offers support for the most commonly used native XML databases, including:

- eXist
- MarkLogic
- Oracle XML DB
- Base X
- Berkeley DB XML (Deprecated)
IBM DB2 Database Connections

Oxygen XML Editor includes support for IBM DB2 database connections. Oxygen XML Editor allows you to browse the structure of an IBM DB2 database in the **Data Source Explorer** view (on page 2015), open tables in the **Table Explorer** view (on page 2017), and perform various operations on the resources in the repository.

**Figure 521. IBM DB2 Database Connection**

---

**Configuring an IBM DB2 Database Connection**

To configure the support for the IBM DB2 database, follow this procedure:

1. Go to the [IBM website](https://www.ibm.com) and in the **DB2 Clients and Development Tools** category select the **DB2 Driver for JDBC and SQLJ** download link. Fill out the download form and download the zip file. Unzip the zip file and use the `db2jcc.jar` and `db2jcc_license_cu.jar` files in Oxygen XML Editor for configuring a DB2 data source (on page 2022).
2. Configure IBM DB2 **Data Source** drivers (on page 2022).
3. Configure an IBM DB2 Server Connection (on page 2023).

4. To view your connection, go to the Data Source Explorer view (on page 2015) (if the view is not displayed, it can be opened by selecting it from the Window > Show View menu) or switch to the Database perspective (on page 3221).

How to Configure IBM DB2 Data Source Drivers

Note:
Available in the Enterprise edition only.

To configure a data source for connecting to an IBM DB2 server, follow these steps:

1. Go to the IBM website and in the DB2 Clients and Development Tools category select the DB2 Driver for JDBC and SQLJ download link. Fill out the download form and download the zip file.
2. Unzip the downloaded archive.
3. Open the Preferences dialog box (Options > Preferences) (on page 124) and go to Data Sources.
4. Click the New button in the Data Sources panel.

The dialog box for configuring a data source is opened.

Figure 522. Data Source Drivers Configuration Dialog Box

5. Enter a unique name for the data source.
6. Select **DB2** in the driver **Type** drop-down menu.

7. Click the **Add Files** button and select the IBM DB2 driver files from the archive that you downloaded and unzipped.

   The IBM DB2 driver files are:
   - `db2jcc.jar`
   - `db2jcc_license_cisuz.jar`
   - `db2jcc_license_cu.jar`

8. Select the most appropriate **Driver class**.

9. Click the **OK** button to finish the data source configuration.

10. Continue on to configure your IBM DB2 connection (on page 2023).

### How to Configure an IBM DB2 Connection

**Note:**

The support to create an IBM DB2 connection is available in the Enterprise edition only.

To configure a connection to an IBM DB2 server, follow these steps:

1. **Open the Preferences dialog box (Options > Preferences)** (on page 124) and go to **Data Sources**.

2. Click the **New** button in the **Connections** panel.

   The dialog box for configuring a database connection is displayed.

**Figure 523. Connection Configuration Dialog Box**
3. Enter a unique name for the connection.
4. Select an *IBM DB2* data source in the *Data Source* drop-down menu.
5. Enter the connection details.
   a. Enter the URL to the installed IBM DB2 engine.
   b. Enter the user name to access the IBM DB2 engine.
   c. Enter the password to access the IBM DB2 engine.
6. Click the *OK* button to finish the connection configuration.
7. To view your connection, go to the *Data Source Explorer view* (on page 2015) (if the view is not displayed, it can be opened by selecting it from the *Window > Show View* menu) or switch to the *Database perspective* (on page 3221).

**IBM DB2 Contextual Menu Actions**

**General Contextual Menu Actions**

For relational databases, the following general actions are available in the contextual menu of the *Data Source Explorer view* (on page 2015), depending on the node where it is invoked:

- **Refresh**
  - Performs a refresh on the selected node.
- **Disconnect (available on ➡️ Connection nodes)**
  - Closes the current database connection. If a table is already open, you are warned to close it before proceeding.
- **❖ Configure Database Sources (available on ➡️ Connection nodes)**
  - Opens the *Data Sources preferences page* (on page 276) where you can configure both data sources and connections.
- **>Edit (available on ➡️ Table nodes)**
  - Opens the selected table in the *Table Explorer view* (on page 2017).
- **❖ Export to XML (available on ➡️ Table nodes)**
  - Opens the *Export Criteria* dialog box (a thorough description of this dialog box can be found in the *Import from Database* (on page 2096) chapter).

**Database-Specific Contextual Menu Actions**

In addition to the general contextual menu actions in the *Data Source Explorer view* (on page 2015), the various nodes in IBM DB2 connections include the following additional contextual menu actions:

- **XML Schema Repository Level Nodes**
  - **Register**
Opens a dialog box for adding a new schema file in the DB XML repository. In this dialog box, you enter a collection name and the necessary schema files. Schema dependencies management can be done by using the Add and Remove buttons.

### Schema Level Nodes

- **Unregister**
  - Removes the selected schema from the XML Schema Repository.

- **View**
  - Opens the selected schema in Oxygen XML Editor.

### Microsoft SQL Server Database Connections

Oxygen XML Editor includes support for Microsoft SQL Server database connections. Oxygen XML Editor allows you to browse the structure of a SQL Server database in the **Data Source Explorer view (on page 2015)**, open tables in the **Table Explorer view (on page 2017)**, and perform various operations on the resources in the repository.

#### Configuring a Microsoft SQL Server Connection

To configure the support for a Microsoft SQL Server database, follow this procedure:

2. Configure MS SQL Server Data Source drivers (on page 2025).
3. Configure a MS SQL Server Connection (on page 2026).
4. To view your connection, go to the **Data Source Explorer view (on page 2015)** (if the view is not displayed, it can be opened by selecting it from the Window > Show View menu) or switch to the **Database perspective (on page 3221)**.

#### How to Configure Microsoft SQL Server Data Source Drivers

**Note:**

Available in the Enterprise edition only.

To configure a data source for connecting to a Microsoft SQL server, follow these steps:

2. Open the **Preferences dialog box (Options > Preferences) (on page 124)** and go to **Data Sources**.
3. Click the + New button in the **Data Sources** panel.
The dialog box for configuring a data source is opened.

Figure 524. Data Source Drivers Configuration Dialog Box

4. Enter a unique name for the data source.
5. Select SQLServer in the driver Type drop-down menu.
6. Click the Add Files button and select the Microsoft SQL Server driver file that you downloaded. The SQL Server driver file is called sqljdbc.jar.
7. Select the most appropriate Driver class.
8. Click the OK button to finish the data source configuration.
9. Continue on to configure your Microsoft SQL Server connection (on page 2026).

How to Configure a Microsoft SQL Server Connection

Note: The support to configure a Microsoft SQL Server connection is available in the Enterprise edition only.

To configure a connection to a Microsoft SQL Server, follow these steps:

1. Open the Preferences dialog box (Options > Preferences) (on page 124) and go to Data Sources.
2. Click the + New button in the Connections panel.

The dialog box for configuring a database connection is displayed.
3. Enter a unique name for the connection.

4. Select the SQL Server data source in the Data Source drop-down menu.

5. Enter the connection details.

   a. Enter the URL of the SQL Server server.
      If you want to connect to the server using Windows integrated authentication, you must add ;integratedSecurity=true to the end of the URL. The URL will look like this:

         jdbc:sqlserver://localhost;instanceName=SQLEXPRESS;integratedSecurity=true

   
   Note:
   For integrated authentication, leave the User and Password fields empty.

   b. Enter the user name for the connection to the SQL Server.

   c. Enter the password for the connection to the SQL Server.

6. Click the OK button to finish the connection configuration.

7. To view your connection, go to the Data Source Explorer view (on page 2015) (if the view is not displayed, it can be opened by selecting it from the Window > Show View menu) or switch to the Database perspective (on page 3221).
Microsoft SQL Server Contextual Menu Actions

General Contextual Menu Actions

For relational databases, the following general actions are available in the contextual menu of the Data Source Explorer view (on page 2015), depending on the node where it is invoked:

- **Refresh**
  Performs a refresh on the selected node.

- **Disconnect (available on Connection nodes)**
  Closes the current database connection. If a table is already open, you are warned to close it before proceeding.

- **Configure Database Sources (available on Connection nodes)**
  Opens the Data Sources preferences page (on page 276) where you can configure both data sources and connections.

- **Edit (available on Table nodes)**
  Opens the selected table in the Table Explorer view (on page 2017).

- **Export to XML (available on Table nodes)**
  Opens the Export Criteria dialog box (a thorough description of this dialog box can be found in the Import from Database (on page 2096) chapter).

Database-Specific Contextual Menu Actions

In addition to the general contextual menu actions in the Data Source Explorer view (on page 2015), the resource level nodes in Microsoft SQL Server connections include the following additional contextual menu action:

- **XML Schema Repository Level Nodes**
  - **Register**
    Opens a dialog box for adding a new schema file in the DB XML repository. In this dialog box, you enter a collection name and the necessary schema files. Schema dependencies management can be done by using the Add and Remove buttons.

- **Schema Level Nodes**
  - **Add**
    Adds a new schema to the XML Schema files.
  - **Unregister**
    Removes the selected schema from the XML Schema Repository.
Oxygen XML Editor 24.1

Oracle Database Connections

The Oracle database is a common relational type of database system. Oxygen XML Editor comes with built-in support for the 11g version of the database system. The Oracle database also includes a Oracle XML DB component that adds native XML support. Oxygen XML Editor allows you to browse Oracle repositories in the Data Source Explorer view (on page 2015), open tables in the Table Explorer view (on page 2017), and perform various operations on the resources in the repository.

![Figure 526. Oracle Database Connection](image)

Related Information:
Using XQuery with Oracle XML DB

Configuring an Oracle 11g Database Connection

To configure the support for a Oracle 11g database, follow this procedure:

2. Configure Oracle 11g Data Source drivers (on page 2030).
3. Configure an Oracle 11g Connection (on page 2031).
4. To view your connection, go to the Data Source Explorer view (on page 2015) (if the view is not displayed, it can be opened by selecting it from the Window > Show View menu) or switch to the Database perspective (on page 3221).
How to Configure Oracle 11g Data Source Drivers

Note:
Available in the Enterprise edition only.

To configure a data source for connecting to an Oracle 11g server, follow these steps:


2. Open the Preferences dialog box (Options > Preferences (on page 124)) and go to Data Sources.

3. Click the + New button in the Data Sources panel.

The dialog box for configuring a data source is opened.

Figure 527. Data Source Drivers Configuration Dialog Box

4. Enter a unique name for the data source.

5. Select Oracle in the driver Type drop-down menu.

6. Click the Add Files button and select the Oracle driver file that you downloaded.

   The Oracle driver file is called ojdbc5.jar.

7. Select the most appropriate Driver class.

8. Click the OK button to finish the data source configuration.

9. Continue on to configure your Oracle connection (on page 2031).
How to Configure an Oracle 11g Connection

Note:
Available in the Enterprise edition only.

To configure a connection to an Oracle 11g server, follow these steps:

1. Open the Preferences dialog box (Options > Preferences) (on page 124) and go to Data Sources.
2. Click the New button in the Connections panel.

The dialog box for configuring a database connection is displayed.

**Figure 528. Connection Configuration Dialog Box**

![Connection Configuration Dialog Box](image)

3. Enter a unique name for the connection.
4. Select the Oracle 11g data source in the Data Source drop-down menu.
5. Enter the connection details.
   a. Enter the URL of the Oracle server.
   b. Enter the user name for the connection to the Oracle server.
   c. Enter the password for the connection to the Oracle server.
6. Click the OK button to finish the connection configuration.
7. To view your connection, go to the Data Source Explorer view (on page 2015) (if the view is not displayed, it can be opened by selecting it from the Window > Show View menu) or switch to the Database perspective (on page 3221).
Oracle Database Contextual Menu Actions

General Contextual Menu Actions

For relational databases, the following general actions are available in the contextual menu of the Data Source Explorer view (on page 2015), depending on the node where it is invoked:

- **Refresh**
  Performs a refresh on the selected node.

- **Disconnect (available on Connection nodes)**
  Closes the current database connection. If a table is already open, you are warned to close it before proceeding.

- **Configure Database Sources (available on Connection nodes)**
  Opens the Data Sources preferences page (on page 276) where you can configure both data sources and connections.

- **Edit (available on Table nodes)**
  Opens the selected table in the Table Explorer view (on page 2017).

- **Export to XML (available on Table nodes)**
  Opens the Export Criteria dialog box (a thorough description of this dialog box can be found in the Import from Database (on page 2096) chapter).

Database-Specific Contextual Menu Actions

In addition to the general contextual menu actions in the Data Source Explorer view (on page 2015), the various nodes in Oracle database connections include the following additional contextual menu actions:

- **XML Schema Repository Level Nodes**
  - **Register**
    Opens a dialog box for adding a new schema file in the XML repository. To add an XML Schema, enter the schema URI and location on your file system. **Local scope** means that the schema is visible only to the user who registers it. **Global scope** means that the schema is public.

  **Note:**
  Registering a schema may involve dropping/creating types. Hence you need type-related privileges such as DROP TYPE, CREATE TYPE, and ALTER TYPE. You need privileges to delete and register the XML schemas involved in the registering process. You need all privileges on XMLType tables that conform to the registered schemas. For XMLType columns, the ALTER TABLE privilege is needed on corresponding tables. If there are
schema-based XMLType tables or columns in other database schemas, you need privileges such as the following:

- CREATE ANY TABLE
- CREATE ANY INDEX
- SELECT ANY TABLE
- UPDATE ANY TABLE
- INSERT ANY TABLE
- DELETE ANY TABLE
- DROP ANY TABLE
- ALTER ANY TABLE
- DROP ANY INDEX

To avoid having to grant all these privileges to the schema owner, Oracle recommends that the registration be performed by a DBA if there are XML schema-based XMLType table or columns in other user database schemas.

XML Repository Level Nodes

Add container

Adds a new child container to the current one.

Add resource

Adds a new resource to the folder.

Container Level Nodes

Add container

Adds a new child container to the current one.

Add resource

Adds a new resource to the folder.

Delete

Deletes the current container.

Properties

Shows various properties of the current container.

Resource Level Nodes

Open

Opens the selected resource in the editor.

Open in System Application
When you use this action, Oxygen XML Editor downloads the selected resource to a local temporary folder and opens the selected resource in the system application that is currently set as the default application associated with that type of resource. You can then edit the resource, save it, and when you switch the focus back to the **Data Source Explorer** view, Oxygen XML Editor will detect that there was a change and will ask if you want to upload the edited resource to the server.

**Rename**

Renames the current resource

**Move**

Moves the current resource to a new container (also available through drag and drop).

**Delete**

Deletes the current container.

**Copy location**

Allows you to copy (to the clipboard) an application-specific URL for the resource that can then be used for various actions, such as opening or transforming the resources.

**Properties**

Shows various properties of the current container.

**Compare**

Compares two selected resources using the **Compare Files** tool (on page 474).

**PostgreSQL Database Connections**

Oxygen XML Editor includes support for PostgreSQL database connections. Oxygen XML Editor allows you to browse the structure of a PostgreSQL database in the **Data Source Explorer** view (on page 2015), open tables in the **Table Explorer** view (on page 2017), and perform various operations on the resources in the repository.
Configuring a PostgreSQL Database Connection

To configure the support for a PostgreSQL database, follow this procedure:

1. Go to [http://jdbc.postgresql.org/download.html](http://jdbc.postgresql.org/download.html) and download the PostgreSQL JDBC driver specific for your server version.
2. Configure PostgreSQL **Data Source** drivers (on page 2035).
3. Configure a PostgreSQL **Connection** (on page 2036).
4. To view your connection, go to the **Data Source Explorer** view (on page 2015) (if the view is not displayed, it can be opened by selecting it from the **Window > Show View** menu) or switch to the **Database perspective** (on page 3221).

How to Configure PostgreSQL Data Source Drivers

To configure a data source for connecting to a PostgreSQL server, follow these steps:

1. Go to [http://jdbc.postgresql.org/download.html](http://jdbc.postgresql.org/download.html) and download the PostgreSQL JDBC3 driver specific for your server version.
2. Open the **Preferences** dialog box (**Options > Preferences**) (on page 124) and go to **Data Sources**.
3. Click the **New** button in the **Data Sources** panel.

   The dialog box for configuring a data source is opened.

4. Enter a unique name for the data source.
5. Select PostgreSQL in the driver Type drop-down list.
6. Click the Add Files button and select the PostgreSQL driver file that you downloaded.
7. Select the most appropriate Driver class.
8. Click the OK button to finish the data source configuration.
9. Continue to configure your PostgreSQL connection (on page 2036).

**How to Configure a PostgreSQL Connection**

To configure a connection to a PostgreSQL server, follow these steps:

1. Open the Preferences dialog box (Options > Preferences) (on page 124) and go to Data Sources.
2. Click the + New button in the Connections panel.

The dialog box for configuring a database connection is displayed.

**Figure 530. Connection Configuration Dialog Box**

![Connection Configuration Dialog Box](image)

3. Enter a unique name for the connection.
4. Select the PostgreSQL data source in the Data Source drop-down menu.
5. Enter the connection details.
   a. Enter the URL of the PostgreSQL server.
   b. Enter the user name for the connection to the PostgreSQL server.
   c. Enter the password for the connection to the PostgreSQL server.
6. Click the OK button to finish the connection configuration.
7. To view your connection, go to the Data Source Explorer view (on page 2015) (if the view is not displayed, it can be opened by selecting it from the Window > Show View menu) or switch to the Database perspective (on page 3221).
PostgreSQL Contextual Menu Actions

General Contextual Menu Actions

For relational databases, the following general actions are available in the contextual menu of the Data Source Explorer view (on page 2015), depending on the node where it is invoked:

- **Refresh**
  Performs a refresh on the selected node.

- **Disconnect (available on Connection nodes)**
  Closes the current database connection. If a table is already open, you are warned to close it before proceeding.

- **Configure Database Sources (available on Connection nodes)**
  Opens the Data Sources preferences page (on page 276) where you can configure both data sources and connections.

- **Edit (available on Table nodes)**
  Opens the selected table in the Table Explorer view (on page 2017).

- **Export to XML (available on Table nodes)**
  Opens the Export Criteria dialog box (a thorough description of this dialog box can be found in the Import from Database (on page 2096) chapter).

Database-Specific Contextual Menu Actions

In addition to the general contextual menu actions in the Data Source Explorer view (on page 2015), the resource level nodes in PostgreSQL connections include the following additional contextual menu action:

- **Resource Level Nodes**

  - **Compare**
    Compares two selected resources using the Compare Files tool (on page 474).

Berkeley DB XML Database Connections (Deprecated)

Oxygen XML Editor includes support for Berkeley DB XML database connections. Oxygen XML Editor allows you to browse the structure of a Berkeley DB XML database in the Data Source Explorer view (on page 2015) and perform various operations on the resources in the repository.

 Oracle Berkeley DB XML is an open source, embeddable XML database with XQuery-based access to documents stored in containers and indexed based on their content. It is built on top of the Oracle Berkeley DB and inherits its features and attributes, along with native XML support. A detailed description can be found at: https://www.oracle.com/database/technologies/related/berkeleydb.html.
Configuring a Berkeley DB XML Database Connection (Deprecated)

Follow this procedure to configure the support for a Berkeley DB XML database:

1. Configure Berkeley DB XML Data Source drivers (on page 2038).
2. Configure a Berkeley DB XML Connection (on page 2039).
3. To view your connection, go to the Data Source Explorer view (on page 2015) (if the view is not displayed, it can be opened by selecting it from the Window > Show View menu) or switch to the Database perspective (on page 3221).

How to Configure Berkeley DB XML Data Source Drivers (Deprecated)

Prerequisite:
For this procedure, you need to already have a Berkeley DB XML database installed on your system.

Oxygen XML Editor supports Berkeley DB XML versions 2.3.10, 2.4.13, 2.4.16 & 2.5.16. To configure a data source for a Berkeley DB XML database, follow these steps:

1. Open the Preferences dialog box (Options > Preferences) (on page 124) and go to Data Sources.
2. Click the + New button in the Data Sources panel.
3. Enter a unique name for the data source.
4. Select Berkeley DBXML from the driver Type drop-down menu.
5. Click the Add Files button to add the Berkeley DB driver files.

The driver files for the Berkeley DB database (and their locations) are as follows:

- db.jar (check for it in [DBXML_DIR]/lib or [DBXML_DIR]/jar)
- dbxml.jar (check for it in [DBXML_DIR]/lib or [DBXML_DIR]/jar)
Where \[DBXML_DIR\] is the Berkeley DB XML database root directory. For example, in Windows it is: C:\Program Files\Oracle\Berkeley DB XML <version>.

6. Click the OK button to finish the data source configuration.
7. Continue on to configure your Berkeley DB XML connection (on page 2039).

How to Configure a Berkeley DB XML Connection (Deprecated)

Oxygen XML Editor supports Berkeley DB XML versions 2.3.10, 2.4.13, 2.4.16 & 2.5.16. To configure a connection to a Berkeley DB XML database, follow these steps:

1. Open the Preferences dialog box (Options > Preferences) (on page 124) and go to Data Sources.
2. Click the + New button in the Connections panel.
3. Enter a unique name for the connection.
4. Select a previously configured Berkeley data source from the Data Source drop-down menu.
5. Enter the connection details.
   a. Set the path to the Berkeley DB XML database directory in the Environment home directory field. Use a directory with write access. DO NOT use the installation directory where Berkeley DB XML is installed if you do not have write access to that directory.
   b. Select the Verbose level: DEBUG, INFO, WARNING, or ERROR.
   c. Optionally, you can select the Join existing environment checkbox.
      If selected, an attempt is made to join an existing environment in the specified home directory and all the original environment settings are preserved. If that fails, try reconfiguring the connection with this option unchecked.
6. Click the OK button to finish the connection configuration.
7. To view your connection, go to the Data Source Explorer view (on page 2015) (if the view is not displayed, it can be opened by selecting it from the Window > Show View menu) or switch to the Database perspective (on page 3221).

Berkeley DB XML Contextual Menu Actions (Deprecated)

While browsing Berkeley DB XML connections in the Data Source Explorer view (on page 2015), the various nodes include the following contextual menu actions:

- **Connection Level Nodes**
  - Configure Database Sources
    Opens the Data Sources preferences page (on page 276) where you can configure both data sources and connections.
  - Disconnect (when connected)
    Stops the connection.
New Collection

Opens a **Container configuration** dialog box that allows you to adds a new container in the repository.

**Figure 532. Container Configuration Dialog Box**

![Container Configuration Dialog Box](image)

This dialog box allows you to configure the following:

- **Name** - The name of the new container.
- **Container type** - At creation time, every container must have a type defined for it. This container type identifies how XML documents are stored in the container. As such, the container type can only be determined at container creation time. You cannot change it when subsequent container opens. You can select one of the following types:
  - **Node container** - XML documents are stored as individual nodes in the container. Each record in the underlying database contains a single leaf node, its attributes and attribute values (if any), and its text nodes (if any). Berkeley DB XML also keeps the information it requires to reassemble the document from the individual nodes stored in the underlying databases. This is the default selection and is the preferred container type.
  - **Whole document container** - The container contains entire documents. The documents are stored without any manipulation of line breaks or whitespace.
- **Allow validation** - If selected, documents will be validated when they are loaded into the container. The default behavior is to not validate documents.
- **Index nodes** - If selected, indices for the container will return nodes rather than documents. The default is to index at the document level. This property has no meaning if the container type is **Whole document container**.
Performs a refresh on the selected node.

Properties

Shows various properties of the current container.

Find/Replace in Files

Opens the Find/Replace in Files dialog box (on page 436) that allows you to find and replace text in multiple files from the connection.

Container Level Nodes

Import Files

Allows you to add a new file on the connection, in the current folder.

Export

Allows you to export the folder on the remote connection to a local folder.

Cut

Removes the current selection and places it in the clipboard.

Paste

Pastes the copied selection.

Rename

Renames the current resource

Delete

Deletes the current container.

Edit indices

Opens a Container Indices dialog box that allows you to configure indices properties for the selected Berkeley container.
This dialog box allows you to configure the following properties:

- **Granularity** - A measure of the level of details of your data in the database.
  - You can select one of the following:
    - **Document level** - Good option for retrieving large documents.
    - **Node level** - Good option for retrieving nodes from within documents.
- **Node** - The name of the node.
- **Namespace** - The index namespace.
- **Index type**:
  - **Uniqueness** - Indicates whether or not the indexed value must be unique within the container.
  - **Path type** - Drop-down menu that allows you to select from the following:
    - **Node** - Indicates that you want to index a single node in the path.
    - **Edge** - Indicates that you want to index the portion of the path where two nodes meet.
  - **Node type** - Drop-down menu that allows you to select from the following:
    - **Element** - An element node in the document content.
    - **Attribute** - An attribute node in the document content.
    - **Metadata** - A node found only in the metadata content of a document.
  - **Key type** - Drop-down menu that allows you to select from the following:
• **equality** - Improves the performances of tests that look for
  nodes with a specific value.

• **presence** - Improves the performances of tests that look for
  the existence of a node regardless of its value.

• **substring** - Improves the performance of tests that look for a
  node whose value contains a given sub-string.

  ◦ **Syntax** - The syntax describes the type of data the index contains and
  is mostly used to determine how indexed values are compared. The
  default value is `string`.

**Refresh**

Performs a refresh on the selected node.

**Properties**

Shows various properties of the current container.

**Find/Replace in Files**

Opens the Find/Replace in Files dialog box (on page 436) that allows you to find
and replace text in multiple files from the connection.

**Resource Level Nodes**

**Open**

Opens the selected resource in the editor.

**Open in System Application**

When you use this action, Oxygen XML Editor downloads the selected resource
to a local temporary folder and opens the selected resource in the system
application that is currently set as the default application associated with that type
of resource. You can then edit the resource, save it, and when you switch the focus
back to the Data Source Explorer view, Oxygen XML Editor will detect that there
was a change and will ask if you want to upload the edited resource to the server.

**Cut**

Removes the current selection and places it in the clipboard.

**Copy location**

Allows you to copy (to the clipboard) an application-specific URL for the resource
that can then be used for various actions, such as opening or transforming the
resources.

**Rename**

 Renames the current resource

**Delete**
Oxygen XML Editor 24.1 | 14 - Databases and SharePoint | 2044

Deletes the current container.

**Refresh**

Performs a refresh on the selected node.

**Properties**

Shows various properties of the current container.

**Find/Replace in Files**

Opens the Find/Replace in Files dialog box (on page 436) that allows you to find and replace text in multiple files from the connection.

**Compare**

Compares two selected resources using the Compare Files tool (on page 474).

Debugging with Berkeley DB XML (Deprecated)

The Berkeley DB XML database added a debugging interface starting with version 2.5. The current version is supported in the Oxygen XML Editor XQuery Debugger. The same restrictions and peculiarities (on page 2056) apply for the Berkeley debugger as for the MarkLogic debugger.

**eXist Database Connections**

**Attention:**

Oxygen XML Editor has been tested to work with the latest stable eXist version (version 6). It might work with previous eXist versions, but they have not been tested and cannot be guaranteed to be compatible.

Oxygen XML Editor includes support for eXist database connections. Oxygen XML Editor allows you to browse the structure of an eXist database in the Data Source Explorer view (on page 2015) and perform various operations on the resources in the repository.
Configuring an eXist Database Connection

There are two ways to configure the support for an eXist database:

- Use the dedicated Create eXist-db XML connection wizard.
- Use the Data Sources preferences page to manually configure your connection.

How to Configure an eXist Connection Using the Built-in Wizard

To configure a connection for an eXist database using the dedicated Create eXist-db XML connection wizard, follow these steps:

1. Open the Preferences dialog box (Options > Preferences) (on page 124), go to Data Sources and click the Create eXist-db XML connection link.
2. Enter your connection details in the connection wizard and click OK.

Important:
To create an eXist connection using this wizard, Oxygen XML Editor expects the exist/webstart/exist.jnlp path to be accessible at the provided Host and Port.

3. To view your connection, go to the Data Source Explorer view (on page 2015) (if the view is not displayed, it can be opened by selecting it from the Window > Show View menu) or switch to the Database perspective (on page 3221).
Attention:
Oxygen XML Editor has been tested to work with the latest stable eXist version (version 6). It might work with previous eXist versions, but they have not been tested and cannot be guaranteed to be compatible.

How to Configure an eXist Connection Manually

Attention:
For this manual procedure, you need to already have an eXist database server installed. Also, Oxygen XML Editor has been tested to work with the latest stable eXist version (version 6). It might work with previous eXist versions, but they have not been tested and cannot be guaranteed to be compatible.

Tip:
There is an easier way to configure an eXist database connection using a built-in wizard. For more information, see How to Configure an eXist Connection Using the Built-in Wizard (on page 2045).

Step 1: Configure eXist Data Source Drivers

Oxygen XML Editor supports eXist database server versions up to and including version 5.0. To configure a data source for an eXist database, follow these steps:

1. Open the Preferences dialog box (Options > Preferences) (on page 124) and go to Data Sources.
2. Click the + New button in the Data Sources panel.
3. Enter a unique name for the data source.
4. Select eXist from the driver Type drop-down menu.
5. Click the Add Files button to add the eXist driver files. The following driver files should be added and they are found in the installation directory of the eXist database server. Make sure you copy the files from the installation of the eXist server where you want to connect from Oxygen XML Editor.
   - The exist.jar file located in the base directory.
   - All JAR files in the lib/core/ directory.
6. Click the OK button to finish the data source configuration.

Step 2: Configure an eXist Connection

To configure a connection to an eXist database, follow these steps:

1. Open the Preferences dialog box (Options > Preferences) (on page 124) and go to Data Sources.
2. Click the + New button in the Connections panel.
3. Enter a unique name for the connection.
4. Select a previously configured eXist data source from the Data Source drop-down menu.
5. Enter the connection details:
a. Set the URI to the installed eXist engine in the XML DB URI field.

b. Set the user name in the User field.

c. Set the password in the Password field.

d. Enter the start collection in the Collection field.

   eXist organizes all documents in hierarchical collections. Collections are like directories. They are used to group related documents together. This text field allows the user to set the default collection name.

6. Click the OK button to finish the connection configuration.

7. To view your connection, go to the Data Source Explorer view (on page 2015) (if the view is not displayed, it can be opened by selecting it from the Window > Show View menu) or switch to the Database perspective (on page 3221).

For more information about running XQuery against an eXist XML database, watch our video demonstration: https://www.youtube.com/embed/Yoc5h1zSddA

**eXist Contextual Menu Actions**

While browsing eXist database connections in the Data Source Explorer view (on page 2015), the various nodes include the following contextual menu actions:

### Connection Level Nodes

- **Configure Database Sources**
  
  Opens the Data Sources preferences page (on page 276) where you can configure both data sources and connections.

- **Disconnect (when connected)**
  
  Stops the connection.

- **Refresh**
  
  Performs a refresh on the selected node.

- **Find/Replace in Files**
  
  Opens the Find/Replace in Files dialog box (on page 436) that allows you to find and replace text in multiple files from the connection.

### Container Level Nodes

- **New File or New Document**
  
  Creates a new file on the connection, in the current folder.

- **New Collection**
  
  Creates a new collection on the connection.

- **Import Folders**
Imports folders on the server.

**Import Files**

Allows you to add a new file on the connection, in the current folder.

**Export**

Allows you to export the folder on the remote connection to a local folder.

**Cut**

Removes the current selection and places it in the clipboard.

**Paste**

Pastes the copied selection.

**Refresh**

Performs a refresh on the selected node.

**Properties**

Shows various properties of the current container.

**Find/Replace in Files**

Opens the Find/Replace in Files dialog box (on page 436) that allows you to find and replace text in multiple files from the connection.

**Resource Level Nodes**

**Open**

Opens the selected resource in the editor.

**Open in System Application**

When you use this action, Oxygen XML Editor downloads the selected resource to a local temporary folder and opens the selected resource in the system application that is currently set as the default application associated with that type of resource. You can then edit the resource, save it, and when you switch the focus back to the **Data Source Explorer** view, Oxygen XML Editor will detect that there was a change and will ask if you want to upload the edited resource to the server.

**Save As**

Allows you to save the selected resource as a file on disk.

**Cut**

Removes the current selection and places it in the clipboard.

**Copy**

Copies the current selection into the clipboard.
Allows you to copy (to the clipboard) an application-specific URL for the resource that can then be used for various actions, such as opening or transforming the resources.

**Rename**

Renames the current resource

**Delete**

Deletes the current container.

**Refresh**

Performs a refresh on the selected node.

**Properties**

Shows various properties of the current container.

**Find/Replace in Files**

Opens the Find/Replace in Files dialog box (on page 436) that allows you to find and replace text in multiple files from the connection.

**Compare**

Compares two selected resources using the Compare Files tool (on page 474).

### MarkLogic Database Connections

Oxygen XML Editor Enterprise edition includes support for MarkLogic database connections. Once you configure a MarkLogic connection (on page 2051), you can use the Data Source Explorer view (on page 2015) to display all the application servers that are configured on the MarkLogic server. You can expand each application server and view all of its configured modules, and the Data Source Explorer view (on page 2015) allows you to open and edit these modules.

**Note:**

To browse modules located in a database, directory properties must be associated with them. These directory properties are generated automatically if the directory creation property of the database is set to automatic. If this property is set to manual or manual-enforced, add the directory properties of the modules manually, using the XQuery function `xdmp:directory-create()`. For example, for two documents with the `/code/modules/main.xqy` and `/code/modules/imports/import.xqy` IDs, run the following query:

```
(xdmp:directory-create('/code/modules/'), xdmp:directory-create('/code/modules/imports/'))
```

For more information about directory properties, go to: [http://blakeley.com/blogofile/2012/03/19/directory-assistance/](http://blakeley.com/blogofile/2012/03/19/directory-assistance/)
MarkLogic and XQuery

MarkLogic connections can be used in conjunction with XQuery scripts to debug and solve problems with XQuery transformations. XQuery modules can also be validated using a MarkLogic server to allow you to spot possible issues without the need of actually executing the XQuery script.

When debugging XQuery files with MarkLogic (on page 2054), you can use the Data Source Explorer view (on page 2015) to open the files from the application server that is involved in the debugging process. By using the Data Source Explorer view (on page 2015), any imported modules are better identified by the MarkLogic server. You can also use step actions and breakpoints (on page 2056) in the modules to help identify problems.

Modules Container

For each Application server (for example: Bill (HTTP port:8060)), you have access to the XQuery modules that are visible to that server. When editing, executing, or debugging XQuery it is recommended to open the XQuery files from this Modules container.

Note:

You can also manage resources for a MarkLogic database through a WebDAV connection, although it is not recommended if you work with XQuery files since imported modules may not be resolved correctly.

Requests Container

Each MarkLogic application server includes a Requests container. In this container, Oxygen XML Editor displays both queries that are stopped for debugging purposes and queries that are still running. To clean up the entire Requests container at the end of your session, right-click it and use the Cancel all requests action (on page 2058).
Configuring a MarkLogic Database Connection

Note that this feature is available in Oxygen XML Editor Enterprise edition only.

Follow this procedure to configure the support for a MarkLogic database connection:

1. Download the MarkLogic driver from MarkLogic Community site.
2. Configure MarkLogic Data Source drivers (on page 2051).
3. Configure a MarkLogic Connection (on page 2052).
4. To view your connection, go to the Data Source Explorer view (on page 2015) (if the view is not displayed, it can be opened by selecting it from the Window > Show View menu) or switch to the Database perspective (on page 3221).

Related Information:

MarkLogic Development in Oxygen XML Editor (on page 2053)

How to Configure MarkLogic Data Source Drivers

Notes:

- Available in the Enterprise edition only.
- Oxygen XML Editor supports MarkLogic version 4.0 or later.

To configure a data source for MarkLogic, follow this procedure:
2. Unzip the downloaded archive.
3. **Open the Preferences dialog box (Options > Preferences)** (on page 124) and go to **Data Sources**.
4. Click the **+** **New** button in the **Data Sources** panel.
5. Enter a unique name for the data source.
6. Select **MarkLogic** from the driver **Type** drop-down list.
7. Click the **Add Files** button and select the MarkLogic driver file from the **lib** folder of the archive that you downloaded and unzipped. The driver file name is `marklogic-xcc-{server_version}.jar`, where `{server_version}` is the MarkLogic server version.
8. Click the **OK** button to finish the data source configuration.
9. Continue on to configure your **MarkLogic Connection** (on page 2052).

### How to Configure a MarkLogic Connection

**Notes:**

- Available in the Enterprise edition only.
- Oxygen XML Editor supports MarkLogic version 4.0 or later.

To configure a connection to a MarkLogic database, follow these steps:

1. **Open the Preferences dialog box (Options > Preferences)** (on page 124) and go to **Data Sources**.
2. Click the **+** **New** button in the **Connections** panel.
3. Enter a unique name for the connection.
4. Select a previously configured MarkLogic data source from the **Data Source** drop-down menu.
5. Enter the connection details.
   
   a. The host name or IP address of the installed MarkLogic engine in the **XDBC Host** field.
      Oxygen XML Editor uses XCC connector to interact with MarkLogic XDBC server and requires the basic authentication schema to be set. Starting with version MarkLogic 4.0 the default authentication method when you create an HTTP or WebDAV Server is **digest**, so make sure to change it to **basic**.
   
   b. Set the port number of the MarkLogic engine in the **Port** field. A MarkLogic XDBC application server must be configured on the server on this port. This XDBC server will be used to process XQuery expressions against the server. Later, if you want to change the XDBC server, instead of editing the configuration just use the **Use it to execute queries** action (on page 2057) from Data Source Explorer.
   
   c. Set the user name to access the MarkLogic engine in the **User** field.
   
   d. Set the password to access the MarkLogic engine in the **Password** field.
e. Optionally, in the WebDAV URL field, set the URL used for browsing the MarkLogic database in the Data Source Explorer view (on page 2015). The Database field specifies the database that will have the XQuery expressions executed. If you set this option to default, the database associated to the application server of the configured port is used.

6. Click the OK button to finish the connection configuration.

7. To view your connection, go to the Data Source Explorer view (on page 2015) (if the view is not displayed, it can be opened by selecting it from the Window > Show View menu) or switch to the Database perspective (on page 3221).

MarkLogic Development in Oxygen XML Editor

The Oxygen XML Editor support for MarkLogic includes features designed for developers, such as debugging XQuery transformations, remote and collaborative debugging, XQuery editing and validation, and an XQuery builder (on page 1039) that helps to improve productivity.

Working with XQuery Files

MarkLogic supports working with XQuery files to create queries over stored XML content. You can open an XQuery file, configure a transformation scenario to match your MarkLogic connection, write the XQuery, and then execute it.

When editing XQuery modules stored on the MarkLogic server, the Outline view (on page 1037) collects and displays all the functions from all imported modules. The Content Completion Assistant (on page 3217) also presents all of these functions along with the latest built-in XQuery functions in accordance with the server version.

When developing queries for MarkLogic, it is best to open the resources from the Data Source Explorer view (on page 2015). When you execute or debug XQuery files opened from this view, imported modules can be resolved better by the MarkLogic server. Another advantage is that validation is automatically performed on the MarkLogic server, including any imported modules.

XQuery Debugging

Oxygen XML Editor allows you to use MarkLogic connections to debug real applications that use XQuery (for example, web applications that trigger XQuery executions). By setting the server in debug mode, you can intercept all the XQuery scripts that run on that server. Oxygen XML Editor connects to the MarkLogic server, shows you the running XQuery scripts, and allows you to debug them. The remote debugging support also allows you to debug collaboratively. Multiple users can participate in the same debugging session. You can start a debugging session and another user can continue it, and vice versa.

Working with Modules

MarkLogic has a concept of two types of XQuery modules, library and main modules. A library module is used to define functions. Library modules cannot be evaluated directly. They are imported, either from other library
modules or from main modules. A main module is used as an entry point that can be executed as an XQuery program. For more information on these types of modules, see XQuery Library Modules and Main Modules.

When working with library modules, you need to create a validation scenario and associate it with the module. In the validation scenario you need to specify a main module as the entry point for validation. The modules need to be deployed on a MarkLogic server because Oxygen XML Editor will request the server to validate the modules.

To validate library modules stored on a MarkLogic server, follow these steps:

1. Configure a MarkLogic database connection (on page 2051).
2. Expand the MarkLogic connection in the Data Source Explorer view (on page 2015) and open the library modules. The main module must also be opened from the Data Source Explorer view (on page 2015).
3. Configure a validation scenario (on page 788) for each library module. Specify the main module in the URL of the file to validate field.

Result: Validation is done on the server that contains the main module. The main module and all other library modules involved in the validation must be saved. Otherwise, the server will validate what was saved on the server, without the uncommitted changes. Also, the Content Completion Assistant (on page 3217) and the Outline view (on page 1037) should now present the functions from all the modules.

Related Information:
- Debugging with MarkLogic (on page 2054)
- Configuring a MarkLogic Database Connection (on page 2051)

**Debugging with MarkLogic**

Oxygen XML Editor includes support for debugging XQuery transformations that are executed against a MarkLogic database.

To use a debugging session against the MarkLogic engine, follow these steps:

1. Configure a MarkLogic data source (on page 2051) and a MarkLogic connection (on page 2052).
2. Make sure that the debugging support is enabled in the MarkLogic server that Oxygen XML Editor accesses. On the server side, debugging must be activated in the XDBC server and in the Task Server section of the server control console (the switch debug allow). If the debugging is not activated, the MarkLogic server reports a DBG-TASKDEBUGALLOW error.

Note:
An XDBC application server must be running to connect to the MarkLogic server and this XDBC server will be used to process XQuery expressions against the server. You can change
3. Open the XQuery file and start the debugging process.

- If you want to debug an XQuery file stored on the MarkLogic server, it is recommended to use the **Data Source Explorer view (on page 2015)** and open the file from the application server that is involved in the debugging process. This improves the resolving of any imported modules.

- The MarkLogic XQuery debugger integrates seamlessly into the **XQuery Debugger perspective (on page 349)**. If you have a MarkLogic validation scenario configured for the XQuery file, you can choose to debug the scenario (on page 2121) directly.

- Otherwise, switch to the **XQuery Debugger perspective (on page 3221)**, open the XQuery file in the editor, and select the MarkLogic connection in the XQuery engine selector from the debug control toolbar (on page 2105).

For general information about how a debugging session is started and controlled, see the **Working with the Debugger (on page 2121)** section.

In a MarkLogic debugging session, you can use step actions and **breakpoints (on page 2125)** to help identify problems. When you add a **breakpoint (on page 2126)** on a line where the debugger never stops, Oxygen XML Editor displays a warning message. These warnings are displayed for **breakpoints** you add either in the main XQuery (which you can open locally or from the server) or for **breakpoints** you add in any XQuery that is opened from the connection that participates in the debugging session. For more information, see **Using Breakpoints for Debugging Queries that Import Modules with MarkLogic (on page 2056)**.

### Remote Debugging with MarkLogic

Oxygen XML Editor allows you to debug remote applications that use XQuery (for example, web applications that trigger XQuery executions). Oxygen XML Editor connects to a MarkLogic server, shows you the running XQuery scripts and allows you to debug them. You can even pause the scripts so that you can start the debugging queries in the exact context of the application. You can also switch a server to debug mode to intercept all XQuery scripts.

Oxygen XML Editor also supports collaborative debugging. This feature allows multiple users to participate in the same debugging session. You can start a debugging session and at a certain point, another user can continue it.

**Important:**

When using the remote debugging feature, the HTTP and the XDBC servers involved in the debugging session must have the same module configuration.

For more information about the XQuery debugger for MarkLogic, watch our video demonstration:
Using Breakpoints for Debugging Queries that Import Modules with MarkLogic

When debugging queries that imports modules stored in the database, it is recommended to place breakpoints in the modules. When starting a new debugging session, make sure that the modules that you will debug are already opened in the editor. This is necessary so that the breakpoints in all the modules will be considered. Also, make sure that there are no other open modules that are not involved in the current debugging session.

To place breakpoints in the modules, use the following procedure:

1. In the Data Source Explorer view, open all the modules from the Modules container of the XDBC application server that performs the debugging.
2. Set breakpoints in the module as needed.
3. Continue debugging the query.

If you get a warning that the breakpoints failed to initialize, try the following solutions:

- Check the Breakpoints view and make sure there are no older breakpoints (set on resources that are not part of the current debugging context).
- Make sure you open the modules from the context of the application server that does the debugging and place breakpoints there.

Peculiarities and Limitations of the MarkLogic Debugger

MarkLogic debugger has the following peculiarities and limitations:

- Debugging support is only available for MarkLogic server versions 4.0 or newer.
- For MarkLogic server versions 4.0 or newer, there are three XQuery syntaxes that are supported: '0.9-ml' (inherited from MarkLogic 3.2), '1.0-ml', and '1.0'.
- All declared variables are presented as strings. The Value column of the Variables view contains the expression from the variable declaration. It can be evaluated by copying the expression with the Copy button.
value action from the contextual menu of the Variables view (on page 2119) and pasting it in the XWatch view (on page 2111).

- There is no support for output to source mapping (on page 2122).
- There is no support for showing the trace (on page 2116).
- You can only set breakpoints (on page 2109) in imported modules in one of the following cases:
  - When you open the module from the context of the application server involved in the debugging, using the Data Source Explorer view (on page 2015).
  - When the debugger automatically opens the modules in the Editor.
- No breakpoints (on page 2125) are set in modules from the same server that are not involved in the current debugging session.
- No support for profiling (on page 2126) when an XQuery transformation is executed in the debugger.

MarkLogic Contextual Menu Actions

While browsing MarkLogic connections in the Data Source Explorer view (on page 2015), the various nodes include the following contextual menu actions:

### Connection Level Nodes

#### Configure Database Sources
Opens the Data Sources preferences page (on page 276) where you can configure both data sources and connections.

#### Disconnect (when connected)
Stops the connection.

#### Refresh
Performs a refresh on the selected node.

#### Find/Replace in Files
Opens the Find/Replace in Files dialog box (on page 436) that allows you to find and replace text in multiple files from the connection.

### Container Level Nodes

#### Enable Debug Mode
Switches the server to a debugging mode. For more information, see MarkLogic debugging sessions (on page 2054).

#### Use it to Execute Queries
The server will be used to process XQuery expressions against it.

#### Refresh
Performs a refresh on the selected node.

### Module or Folder Level Nodes
Export

Allows you to export the folder on the remote connection to a local folder.

**Refresh**

Performs a refresh on the selected node.

Requests Level Nodes

**Refresh**

Performs a refresh on the selected node.

**Cancel all requests**

Cancels all queries that are either running or stopped on the application server. You can use this action to clean up the entire Requests container at the end of your sessions.

Resource Level Nodes

**Open**

Opens the selected resource in the editor.

**Open in System Application**

When you use this action, Oxygen XML Editor downloads the selected resource to a local temporary folder and opens the selected resource in the system application that is currently set as the default application associated with that type of resource. You can then edit the resource, save it, and when you switch the focus back to the Data Source Explorer view, Oxygen XML Editor will detect that there was a change and will ask if you want to upload the edited resource to the server.

**Copy location**

Allows you to copy (to the clipboard) an application-specific URL for the resource that can then be used for various actions, such as opening or transforming the resources.

**Refresh**

Performs a refresh on the selected node.

**Compare**

Compares two selected resources using the Compare Files tool (on page 474).

Related Information:

- Configuring a MarkLogic Database Connection (on page 2051)
- MarkLogic Development in Oxygen XML Editor (on page 2053)
- Debugging with MarkLogic (on page 2054)
MySQL Database Connections

Oxygen XML Editor includes support for MySQL database connections. Oxygen XML Editor allows you to browse the structure of a SQL Server database in the Data Source Explorer view (on page 2015), open tables in the Table Explorer view (on page 2017), and perform various operations on the resources in the repository.

Configuring a MySQL Database Connection

To configure the support for a MySQL database, follow this procedure:

1. Configure MySQL Data Source drivers (on page 2059).
2. Configure a MySQL Connection. (on page 2060)
3. To view your connection, go to the Data Source Explorer view (on page 2015) (if the view is not displayed, it can be opened by selecting it from the Window > Show View menu) or switch to the Database perspective (on page 3221).

How to Configure MySQL Data Source Drivers

To connect to a MySQL server, you need to create a generic JDBC type data source based on the MySQL JDBC driver available on the MySQL website.

To configure this data source, follow these steps:

1. Go to https://www.oxygenxml.com/database_drivers.html and download the appropriate MySQL driver.
2. Open the Preferences dialog box (Options > Preferences) (on page 124) and go to Data Sources.
3. Click the + New button in the Data Sources panel.

   The dialog box for configuring a data source is opened.
4. Enter a unique name for the data source.
5. Select Generic JDBC in the driver Type drop-down list.
6. Click the Add Files button and select the MySQL driver file that you downloaded. The driver file for the MySQL server is called mysql-com.jar.
7. Select the most appropriate Driver class.
8. Click the OK button to finish the data source configuration.
9. Continue on to configure your MySQL connection (on page 2060).

**How to Configure a MySQL Connection**

To configure a connection to a MySQL server, follow these steps:

1. Open the Preferences dialog box (Options > Preferences) (on page 124) and go to Data Sources.
2. Click the + New button in the Connections panel.

   The dialog box for configuring a database connection is displayed.
3. Enter a unique name for the connection.
4. Select the MySQL data source in the Data Source drop-down list.
5. Enter the connection details.
   a. Enter the URL of the MySQL server.
   b. Enter the user name for the connection to the MySQL server.
   c. Enter the password for the connection to the MySQL server.
6. Click the OK button to finish the connection configuration.
7. To view your connection, go to the Data Source Explorer view (on page 2015) (if the view is not displayed, it can be opened by selecting it from the Window > Show View menu) or switch to the Database perspective (on page 3221).

**Generic JDBC Database Connections**

Oxygen XML Editor includes support for Generic JDBC database connections.

**Configuring a Generic JDBC Database Connection**

To configure the support for a generic JDBC database, follow this procedure:

1. Configure Generic JDBC Data Source drivers (on page 2062).
2. Configure a Generic JDBC Connection (on page 2062).
3. To view your connection, go to the Data Source Explorer view (on page 2015) (if the view is not displayed, it can be opened by selecting it from the Window > Show View menu) or switch to the Database perspective (on page 3221).
How to Configure Generic JDBC Data Source Drivers

Starting with version 17, Oxygen XML Editor comes bundled with Java 8, which does not provide built-in access to JDBC-ODBC data sources. To access such sources, you need to find an alternative JDBC-ODBC bridge or use a platform-independent distribution of Oxygen XML Editor along with a Java VM version 7 or 6.

To configure a generic JDBC data source, follow these steps:

1. Open the Preferences dialog box (Options > Preferences) (on page 124) and go to Data Sources.
2. Click the + New button in the Data Sources panel.
3. Enter a unique name for the data source.
4. Select Generic JDBC in the driver Type drop-down list.
5. Add the driver file(s) using the Add Files button.
6. Select the most appropriate Driver class.
7. Click the OK button to finish the data source configuration.
8. Continue on to configure a generic JDBC connection (on page 2062).

How to Configure a Generic JDBC Connection

To configure a connection to a generic JDBC database, follow these steps:

1. Open the Preferences dialog box (Options > Preferences) (on page 124) and go to Data Sources.
2. Click the + New button in the Connections panel.
3. Enter a unique name for the connection.
4. Select the Generic JDBC data source in the Data Source drop-down menu.
5. Enter the connection details.
   a. Enter the URL of the generic JDBC database, with the following format: jdbc: <subprotocol>:<subname>.
   b. Enter the user name for the connection to the generic JDBC database.
   c. Enter the password for the connection to the generic JDBC database.
6. Click the OK button to finish the connection configuration.
7. To view your connection, go to the Data Source Explorer view (on page 2015) (if the view is not displayed, it can be opened by selecting it from the Window > Show View menu) or switch to the Database perspective (on page 3221).

JDBC-ODBC Database Connections

Oxygen XML Editor includes support for JDBC-ODBC database connections.

How to Configure a JDBC-ODBC Connection

Starting with version 17, Oxygen XML Editor comes bundled with Java 8, which does not provide built-in access to JDBC-ODBC data sources. To access such sources, you need to find an alternative JDBC-ODBC bridge or use a platform-independent distribution of Oxygen XML Editor along with a Java VM version 7 or 6.

To configure a connection to an ODBC data source, follow these steps:
1. Open the Preferences dialog box (Options > Preferences) (on page 124) and go to Data Sources.
2. Click the + New button in the Connections panel.

The dialog box for configuring a database connection is displayed.

**Figure 538. Connection Configuration Dialog Box**

3. Enter a unique name for the connection.
4. Select JDBC-ODBC Bridge in the Data Source drop-down list.
5. Enter the connection details.
   a. Enter the URL of the ODBC source.
   b. Enter the user name of the ODBC source.
   c. Enter the password of the ODBC source.
6. Click the OK button to finish the connection configuration.
7. To view your connection, go to the Data Source Explorer view (on page 2015) (if the view is not displayed, it can be opened by selecting it from the Window > Show View menu) or switch to the Database perspective (on page 3221).

**BaseX Database Connections**

Oxygen XML Editor includes support for BaseX database connections using a WebDAV connection. BaseX is a light-weight XML database engine and XQuery processor. Oxygen XML Editor allows you to browse the structure of a BaseX database in the Data Source Explorer view (on page 2015) and perform XQuery executions.
How to Configure a BaseX Connection

To configure a BaseX connection, follow these steps:

1. First of all, make sure the BaseX HTTP Server is started. For details about starting the BaseX HTTP server, go to http://docs.basex.org/wiki/Startup#BaseX_HTTP_Server. The configuration file for the HTTP server is named .basex and is located in the BaseX installation directory. This file helps you to find out which port the HTTP server is using. The default port for BaseX WebDAV is 8984.

2. To ensure that everything is functioning, open a WebDAV URL inside a browser and check to see if it works. For example, the following URL retrieves a document from a database named TEST: http://localhost:8984/webdav/TEST/etc/factbook.xml.

3. Once you are sure that the BaseX WebDAV service is working, you can configure the WebDAV connection in Oxygen XML Editor as described in How to Configure a WebDAV Connection (on page 2068). The WebDAV URL should resemble this: http://{hostname}:{port}/webdav/. If the BaseX server is running on your own machine and it has the default configuration, the data required by the WebDAV connection is:

   ◦ WebDAV URL: http://localhost:8984/webdav
   ◦ User: admin
   ◦ Password: admin

4. Once the WebDAV connection is created, to view your connection, go to the Data Source Explorer view (on page 2015) (if the view is not displayed, it can be opened by selecting it from the Window > Show View menu) or switch to the Database perspective (on page 3221).

BaseX Contextual Menu Actions

While browsing BaseX connections in the Data Source Explorer view (on page 2015), the various nodes include the following contextual menu actions:

Connection Level Nodes

Configure Database Sources

Opens the Data Sources preferences page (on page 276) where you can configure both data sources and connections.

Disconnect (when connected)

Stops the connection.

New Folder

Creates a new folder on the connection.

Import Files

Allows you to add a new file on the connection, in the current folder.

Refresh

Performs a refresh on the selected node.
Find/Replace in Files

Opens the Find/Replace in Files dialog box (on page 436) that allows you to find and replace text in multiple files from the connection.

Folder Level Nodes

New File or New Document

Creates a new file on the connection, in the current folder.

New Folder

Creates a new folder on the connection.

Import Folders

Imports folders on the server.

Import Files

Allows you to add a new file on the connection, in the current folder.

Export

Allows you to export the folder on the remote connection to a local folder.

Cut

Removes the current selection and places it in the clipboard.

Copy

Copies the current selection into the clipboard.

Paste

Pastes the copied selection.

Rename

 Renames the current resource

Delete

Deletes the current container.

Refresh

Performs a refresh on the selected node.

Find/Replace in Files

Opens the Find/Replace in Files dialog box (on page 436) that allows you to find and replace text in multiple files from the connection.

Resource Level Nodes

Open

Opens the selected resource in the editor.
**Open in System Application**

When you use this action, Oxygen XML Editor downloads the selected resource to a local temporary folder and opens the selected resource in the system application that is currently set as the default application associated with that type of resource. You can then edit the resource, save it, and when you switch the focus back to the Data Source Explorer view, Oxygen XML Editor will detect that there was a change and will ask if you want to upload the edited resource to the server.

- **Cut**
  Removes the current selection and places it in the clipboard.

- **Copy**
  Copies the current selection into the clipboard.

- **Copy location**
  Allows you to copy (to the clipboard) an application-specific URL for the resource that can then be used for various actions, such as opening or transforming the resources.

- **Rename**
  Renames the current resource.

- **Delete**
  Deletes the current container.

- **Refresh**
  Performs a refresh on the selected node.

- **Properties**
  Shows various properties of the current container.

- **Find/Replace in Files**
  Opens the Find/Replace in Files dialog box (on page 436) that allows you to find and replace text in multiple files from the connection.

- **Compare**
  Compares two selected resources using the Compare Files tool (on page 474).

**Base X XQJ Connection**

XQuery execution is possible in a BaseX connection through an XQJ connection.

**Important:**
The XQJ connector is only capable of running XQuery 1.0 scripts, therefore XQuery 3.0 and 3.1 scripts are not supported.
BaseX XQJ Data Source

First of all, create an XQJ data source as described in How to Configure an XQJ Data Source (on page 2067). The BaseX XQJ API-specific files that must be added in the configuration dialog box are xqj-api-1.0.jar, xqj2-0.1.0.jar and basex-xqj-1.2.3.jar (the version names of the JAR file may differ). These libraries can be downloaded from xqj.net/basex/basex-xqj-1.2.3.zip. As an alternative, you can also find the libraries in the BaseX installation directory, in the lib sub-directory.

BaseX XQJ Connection

The next step is to create an XQJ connection (on page 2068).

For a default BaseX configuration, the following connection details apply (you can modify them when necessary):

- Port: 1984
- serverName: localhost
- user: admin
- password: admin

XQuery Execution

Now that the XQJ connection is configured, open the XQuery file you want to execute in Oxygen XML Editor and create an XQuery Transformation (on page 1536). In the Transformer drop-down menu, select the name of the XQJ connection you created. Apply the transformation scenario and the XQuery will be executed.

How to Configure an XQJ Data Source

Any transformer that offers an XQJ API implementation can be used when validating XQuery or transforming XML documents. An example of an XQuery engine that implements the XQJ API is Zorba.

1. If your XQJ Implementation is native, make sure the directory containing the native libraries of the engine is added to your system environment variables: to PATH - on Windows, to LD_LIBRARY_PATH - on Linux, or to DYLD_LIBRARY_PATH - on macOS. Restart Oxygen XML Editor after configuring the environment variables.
2. Open the Preferences dialog box (Options > Preferences) (on page 124) and go to Data Sources.
3. Click the + New button in the Data Sources panel.
4. Enter a unique name for the data source.
5. Select XQuery API for Java (XQJ) in the Type combo box.
6. Click the Add button to add XQJ API-specific files.
   You can manage the driver files using the Add, Remove, Detect, and Stop buttons.
   Oxygen XML Editor detects any implementation of javax.xml.xquery.XQDataSource and presents it in the Driver class field.
7. Select the most suited driver in the Driver class combo box.
8. Click the OK button to finish the data source configuration.
9. Continue on to configure the XQJ connection (on page 2068).

How to Configure an XQJ Connection

The steps for configuring an XQJ connection are the following:

1. Open the Preferences dialog box (Options > Preferences) (on page 124) and go to Data Sources.
2. Click the + New button in the Connections panel.
3. Enter a unique name for the connection.
4. Select one of the previously configured XQJ data sources (on page 2067) in the Data Source combo box.
5. Fill-in the connection details.
   - The properties presented in the connection details table are automatically detected depending on the selected data source.
6. Click the OK button to finish the connection configuration.

WebDAV Connections

Oxygen XML Editor includes support for WebDAV server connections. Oxygen XML Editor allows you to browse the structure of a WebDAV connection in the Data Source Explorer view (on page 2015) and perform various operations on the resources in the repository.

How to Configure a WebDAV Connection

By default, Oxygen XML Editor contains built-in data source drivers for WebDAV connections. Based on this data source, you can create a WebDAV connection for browsing and editing data from a database that provides a WebDAV interface. The connection is available in the Data Source Explorer view (on page 2015).

To configure a WebDAV connection, follow these steps:

1. Open the Preferences dialog box (Options > Preferences) (on page 124) and go to Data Sources.
2. Click the + New button in the Connections panel.
3. Enter a unique name for the connection.
4. Select one of the WebDAV data sources in the Data Source drop-down menu.
5. Enter the connection details:
   a. Set the URL to the WebDAV repository in the field WebDAV URL.
   b. Set the user name that is used to access the WebDAV repository in the User field.
   c. Set the password that is used to access the WebDAV repository in the Password field.
6. Click the OK button to finish the connection configuration.
7. To view your connection, go to the Data Source Explorer view (on page 2015) (if the view is not displayed, it can be opened by selecting it from the Window > Show View menu) or switch to the Database perspective (on page 3221).
   For more information about the WebDAV support in Oxygen XML Editor, watch our video demonstration:
WebDAV Contextual Menu Actions

While browsing WebDAV connections in the Data Source Explorer view (on page 2015), the various nodes include the following contextual menu actions:

### Connection Level Nodes

- **Configure Database Sources**
  - Opens the Data Sources preferences page (on page 276) where you can configure both data sources and connections.

- **Disconnect (when connected)**
  - Stops the connection.

- **New Folder**
  - Creates a new folder on the connection.

- **Import Files**
  - Allows you to add a new file on the connection, in the current folder.

- **Refresh**
  - Performs a refresh on the selected node.

- **Find/Replace in Files**
  - Opens the Find/Replace in Files dialog box (on page 436) that allows you to find and replace text in multiple files from the connection.

### Folder Level Nodes

- **New File or New Document**
  - Creates a new file on the connection, in the current folder.

- **New Folder**
  - Creates a new folder on the connection.

- **Import Folders**
  - Imports folders on the server.

- **Import Files**
  - Allows you to add a new file on the connection, in the current folder.

- **Export**
  - Allows you to export the folder on the remote connection to a local folder.

- **Cut**
Removes the current selection and places it in the clipboard.

Copy

Copies the current selection into the clipboard.

Paste

Pastes the copied selection.

Rename

Renames the current resource

Delete

Deletes the current container.

Refresh

Performs a refresh on the selected node.

Find/Replace in Files

Opens the Find/Replace in Files dialog box (on page 436) that allows you to find and replace text in multiple files from the connection.

Resource Level Nodes

Open

Opens the selected resource in the editor.

Open in System Application

When you use this action, Oxygen XML Editor downloads the selected resource to a local temporary folder and opens the selected resource in the system application that is currently set as the default application associated with that type of resource. You can then edit the resource, save it, and when you switch the focus back to the Data Source Explorer view, Oxygen XML Editor will detect that there was a change and will ask if you want to upload the edited resource to the server.

Cut

Removes the current selection and places it in the clipboard.

Copy

Copies the current selection into the clipboard.

Copy location

Allows you to copy (to the clipboard) an application-specific URL for the resource that can then be used for various actions, such as opening or transforming the resources.

Rename

Renames the current resource
**Delete**
Deletes the current container.

**Refresh**
Performs a refresh on the selected node.

**Properties**
Shows various properties of the current container.

**Find/Replace in Files**
Opens the Find/Replace in Files dialog box (on page 436) that allows you to find and replace text in multiple files from the connection.

**Compare**
Compares two selected resources using the Compare Files tool (on page 474).

---

**SQL Execution Support**

The database support in Oxygen XML Editor includes support for writing SQL statements, syntax highlighting, folding (on page 3219), and dragging and dropping from the Data Source Explorer view (on page 2015).

It also includes transformation scenarios for executing the statements, and the results are displayed in the Table Explorer view (on page 2017).

---

**Drag and Drop from Data Source Explorer View**

Dragging operations from the Data Source Explorer view (on page 2015) and dropping them in the SQL Editor allows you to create SQL statements quickly by inserting the names of tables and columns in the SQL statements.

1. Configure a database connection (see the specific procedure for your database server in the Database Connection Support (on page 2020) section).
2. Browse to the table you will use in your statement.
3. Drag the table or a column of the table into the editor where a SQL file is open.

Drag and drop actions are available both on the table and on its fields. A pop-up menu is displayed in the SQL editor.
4. Select the type of statement from the pop-up menu.

Depending on your choice, dragging a table results in one of the following statements being inserted into the document:

- **SELECT `field1`, `field2`, ..., FROM `catalog`.`table`** (for example: `SELECT `DEPT`, `DEPTNAME`, `LOCATION` FROM `camera`.`cameraDesc`)
- **UPDATE `catalog`.`table` SET `field1`=, `field2`=, ...** (for example: `UPDATE `camera`.`cameraDesc` SET `DEPT`=, `DEPTNAME`=, `LOCATION`=)
- **INSERT INTO `catalog`.`table` (`field1`, `field2`, ...) VALUES (, , )** (for example: `INSERT INTO `camera`.`cameraDesc` (`DEPT`, `DEPTNAME`, `LOCATION`) VALUES (, , )
- **DELETE FROM `catalog`.`table`** (for example: `DELETE FROM `camera`.`cameraDesc`)

Depending on your choice, dragging a column results in one of the following statements being inserted into the document:
SELECT `field` FROM `catalog`.`table` (for example: SELECT `DEPT` FROM `camera`.`cameraDesc` )
UPDATE `catalog`.`table` SET `field` = (for example: UPDATE `camera`.`cameraDesc` SET `DEPT` = )
INSERT INTO `catalog`.`table` ( `field1` ) VALUES () (for example: INSERT INTO `camera`.`cameraDesc` ( `DEPT` ) VALUES ()
DELETE FROM `catalog`.`table` (for example: DELETE FROM `camera`.`cameraDesc` WHERE `DEPT` = )

SQL Validation

SQL validation support is offered for IBM DB2. Note that if you choose a connection that does not support SQL validation, you will receive a warning when trying to validate. The SQL document is validated using the connection from the associated transformation scenario.

Executing SQL Statements

The steps for executing an SQL statement on a relational database are as follows:

1. Configure a transformation scenario (on page 1432) using the Configure Transformation Scenario(s) action from the toolbar or the Document > Transformation menu.
   A SQL transformation scenario needs a database connection. You can configure a connection using the Preferences button from the SQL transformation dialog box.
   The dialog box contains the list of existing scenarios that apply to SQL documents.
2. Set parameter values for SQL placeholders using the Parameters button from the SQL transformation dialog box.
   For example, in SELECT * FROM `test`.`department` where DEPT = ? or DEPTNAME = ? the two parameters can be configured for the placeholders (?) in the transformation scenario.

   When the SQL statement is executed, the first placeholder is replaced with the value set for the first parameter in the scenario, the second placeholder is replaced by the second parameter value, and so on.

   **Restriction:**
   When a stored procedure is called in an SQL statement executed on an SQL Server database, mixing inline parameter values with values specified using the Parameters button of the scenario dialog box is not recommended. This is due to a limitation of the SQL Server driver for Java applications. An example of stored procedure that is not recommended: call dbo.Test(22, ?).

3. Execute the SQL scenario by clicking the OK or Apply associated button.

   The result of a SQL transformation is displayed in a view (on page 548) at the bottom of the Oxygen XML Editor window.
4. View more complex return values of the SQL transformation in a separate editor panel.
   A more complex value returned by the SQL query (for example, an XMLTYPE or CLOB value) cannot be displayed entirely in the result table.
   
   a. Right-click the cell containing the complex value.
   
   b. Select the action **Copy cell** from the contextual menu.
      The action copies the value in the clipboard.
   
   c. Paste the value into an appropriate editor.
      For example, you can paste the value in an opened XQuery editor panel of Oxygen XML Editor.

**XQuery and Databases**

XQuery is a native XML query language that is useful for querying XML views of relational data to create XML results. It also provides the mechanism to efficiently and easily extract information from Native XML Databases (NXD) and relational data. The following database systems supported in Oxygen XML Editor offer XQuery support:

- **Native XML Databases**:
  - eXist
  - MarkLogic (validation support available starting with version 5)
  - Berkeley DB XML (Deprecated)

- **Relational Databases**:
  - IBM DB2
  - Microsoft SQL Server (validation support not available)
  - Oracle (validation support not available)

**Build Queries with Drag and Drop from the Data Source Explorer View**

When a query is edited in the XQuery editor, the XPath expressions can be composed quickly by dragging them from the **Data Source Explorer** view (on page 2015) and dropping them into the editor panel.

1. Configure the data source drivers (on page 2020) for the particular relational database in the **Data Sources preferences page** (on page 276).
2. Configure the connection (on page 2020) for the particular relational database in the **Data Sources preferences page** (on page 276).
3. Browse the connection in the **Data Source Explorer view** (on page 2015), expanded to the table or column that you want to insert in the query.
4. Drag the table or column name to the XQuery editor panel.
5. Drop the table or column name where the XPath expression is needed.
An XPath expression that selects the dragged name is inserted in the XQuery document at the cursor position.

**XQuery Validation When Connected to a Database**

With Oxygen XML Editor, you can validate your XQuery documents when connected to a database. When you open an XQuery document from a connection that supports validation (for example, MarkLogic, or eXist), by default Oxygen XML Editor uses this connection for validation. If you open an XQuery file using a MarkLogic connection, the validation resolves imports better.

Related Information:
XQuery Validation (on page 1034)

**XQuery Transformation for Databases**

XQuery is designed to retrieve and interpret XML data from any source, whether it is a database or document. Data is stored in relational databases but it is often required that the data be extracted and transformed as XML when interfacing to other components and services. Also, it is an XPath-based querying language supported by most NXD vendors. To perform a query, you need an XQuery transformation scenario.

1. Configure the data source drivers and the connection (on page 2020) for the particular database.
2. Configure an XQuery transformation scenario.
   a. Click the Configure Transformation Scenario toolbar button or go to menu Document > Transformation > Configure Transformation Scenario.
      The Configure Transformation Scenario dialog box (on page 1548) is opened.
   b. Click the New button toward the bottom of the dialog box.
   c. Select XML Transformation with XQUERY (on page 1480).
      The New Scenario dialog box for configuring an XQuery scenario is opened.
d. Insert the scenario name in the dialog box for editing the scenario.

e. Choose the database connection in the **Transformer** drop-down list.

f. Configure any other parameters as needed.
   
   For an XQuery transformation, the output tab has an option called **Sequence** that allows you to run an XQuery in lazy mode. The amount of data extracted from the database is controlled from the **Size limit on Sequence view** option (on page 253) in the XQuery preferences page. If you choose **Perform FO Processing** in the FO Processor tab, the **Sequence** option is ignored.

   g. Click the **OK** button to finish editing the scenario.

Once the scenario is associated with the XQuery file, the query can include calls to specific XQuery functions that are implemented by that engine. The available functions depend on the target database engine selected in the scenario. For example, for eXist, the **Content Completion Assistant** (on page 3217) lists the functions supported by that database engine. This is useful for only inserting calls to the supported functions (standard XQuery functions or extension ones) into the query.

3. Run the transformation scenario.

To view a more complex value returned by the query that cannot be entirely displayed in the XQuery query result table at the bottom of the Oxygen XML Editor window (for example, an XMLTYPE or CLOB value), do the following:
Right-click that table cell.
Select the **Copy cell** action from the contextual menu to copy the value into the clipboard.
Paste the value wherever you need it (for example, in an open XQuery editor panel of Oxygen XML Editor).

**Related information**

- XML Transformation with XQuery *(on page 1480)*
- XQuery XQJ Transformation *(on page 2077)*

**XQuery XQJ Transformation**

XQuery API for Java (XQJ) refers to the common Java API for the XQuery 1.0 specification. The XQJ API enables you to execute XQuery against an XML data source.

**Important:**
The XQJ connector is only capable of running XQuery 1.0 scripts, therefore XQuery 3.0 and 3.1 scripts are not supported.

Oxygen XML Editor supports any transformer that offers an XQJ API implementation and it be used for validating XQuery or transforming XML documents.

To configure the support for XQJ, do the following:

1. Configure an **XQJ Data Source** *(on page 2067).*
2. Configure an **XQJ Connection** *(on page 2068).*
3. To view your connection, go to the **Data Source Explorer view** *(on page 2015)* (if the view is not displayed, it can be opened by selecting it from the **Window > Show View** menu) or switch to the **Database perspective** *(on page 3221).*

**How to Configure an XQJ Data Source**

Any transformer that offers an XQJ API implementation can be used when validating XQuery or transforming XML documents. An example of an XQuery engine that implements the XQJ API is **Zorba**.

1. If your XQJ implementation is native, make sure the directory containing the native libraries of the engine is added to your system environment variables: to **PATH** - on Windows, to **LD_LIBRARY_PATH** - on Linux, or to **DYLD_LIBRARY_PATH** - on macOS. Restart Oxygen XML Editor after configuring the environment variables.
2. Open the **Preferences** dialog box *(Options > Preferences)* *(on page 124)* and go to **Data Sources**.
3. Click the **+ New** button in the **Data Sources** panel.
4. Enter a unique name for the data source.
5. Select **XQuery API for Java (XQJ)** in the **Type** combo box.
6. Click the **Add** button to add XQJ API-specific files.
You can manage the driver files using the **Add**, **Remove**, **Detect**, and **Stop** buttons. Oxygen XML Editor detects any implementation of `javax.xml.xquery.XQDataSource` and presents it in the **Driver class** field.

7. Select the most suited driver in the **Driver class** combo box.
8. Click the **OK** button to finish the data source configuration.
9. Continue on to configure the XQJ connection *(on page 2068)*.

**How to Configure an XQJ Connection**

The steps for configuring an XQJ connection are the following:

1. Open the **Preferences** dialog box (**Options > Preferences**) *(on page 124)* and go to **Data Sources**.
2. Click the **+ New** button in the **Connections** panel.
3. Enter a unique name for the connection.
4. Select one of the previously configured **XQJ data sources** *(on page 2067)* in the **Data Source** combo box.
5. Fill-in the connection details.
   The properties presented in the connection details table are automatically detected depending on the selected data source.
6. Click the **OK** button to finish the connection configuration.

**XQuery Database Debugging**

Oxygen XML Editor includes a debugging interface that helps you to detect and solve problems with XQuery transformations that are executed against MarkLogic databases.

For more information about the debugging support in Oxygen XML Editor, see **Debugging XSLT Stylesheets and XQuery Documents** *(on page 2102)*.

**Debugging with MarkLogic**

Oxygen XML Editor includes support for debugging XQuery transformations that are executed against a MarkLogic database.

To use a debugging session against the MarkLogic engine, follow these steps:

1. Configure a **MarkLogic data source** *(on page 2051)* and a **MarkLogic connection** *(on page 2052)*.
2. Make sure that the debugging support is enabled in the MarkLogic server that Oxygen XML Editor accesses. On the server side, debugging must be activated in the XDBC server and in the **Task Server** section of the server control console (the switch `debug allow`). If the debugging is not activated, the MarkLogic server reports a **DBG-TASKDEBUGALLOW** error.

**Note:**

An XDBC application server must be running to connect to the MarkLogic server and this XDBC server will be used to process XQuery expressions against the server. You can change
the XDBC application server that Oxygen XML Editor uses to process XQuery expressions by selecting the **Use it to execute queries** action ([on page 2057](#)) from the contextual menu in the **Data Source Explorer** view ([on page 2015](#)).

3. Open the XQuery file and start the debugging process.
   - If you want to debug an XQuery file stored on the MarkLogic server, it is recommended to use the **Data Source Explorer** view ([on page 2015](#)) and open the file from the application server that is involved in the debugging process. This improves the resolving of any imported modules.
   - The MarkLogic XQuery debugger integrates seamlessly into the **XQuery Debugger** perspective ([on page 349](#)). If you have a MarkLogic validation scenario configured for the XQuery file, you can choose to debug the scenario ([on page 2121](#)) directly.
   - Otherwise, switch to the **XQuery Debugger** perspective ([on page 3221](#)), open the XQuery file in the editor, and select the MarkLogic connection in the XQuery engine selector from the debug control toolbar ([on page 2105](#)).

   For general information about how a debugging session is started and controlled, see the **Working with the Debugger** ([on page 2121](#)) section.

In a MarkLogic debugging session, you can use step actions and breakpoints ([on page 2125](#)) to help identify problems. When you add a breakpoint ([on page 2126](#)) on a line where the debugger never stops, Oxygen XML Editor displays a warning message. These warnings are displayed for breakpoints you add either in the main XQuery (which you can open locally or from the server) or for breakpoints you add in any XQuery that is opened from the connection that participates in the debugging session. For more information, see Using Breakpoints for Debugging Queries that Import Modules with MarkLogic ([on page 2056](#)).

**Remote Debugging with MarkLogic**

Oxygen XML Editor allows you to debug remote applications that use XQuery (for example, web applications that trigger XQuery executions). Oxygen XML Editor connects to a MarkLogic server, shows you the running XQuery scripts and allows you to debug them. You can even pause the scripts so that you can start the debugging queries in the exact context of the application. You can also switch a server to debug mode to intercept all XQuery scripts.

Oxygen XML Editor also supports collaborative debugging. This feature allows multiple users to participate in the same debugging session. You can start a debugging session and at a certain point, another user can continue it.

**Important:**

When using the remote debugging feature, the HTTP and the XDBC servers involved in the debugging session must have the same module configuration.

For more information about the XQuery debugger for MarkLogic, watch our video demonstration:
Using Breakpoints for Debugging Queries that Import Modules with MarkLogic

When debugging queries that imports modules stored in the database, it is recommended to place breakpoints in the modules. When starting a new debugging session, make sure that the modules that you will debug are already opened in the editor. This is necessary so that the breakpoints in all the modules will be considered. Also, make sure that there are no other open modules that are not involved in the current debugging session.

To place breakpoints in the modules, use the following procedure:

1. In the Data Source Explorer view, open all the modules from the Modules container of the XDBC application server that performs the debugging.
2. Set breakpoints in the module as needed.
3. Continue debugging the query.

If you get a warning that the breakpoints failed to initialize, try the following solutions:

- Check the Breakpoints view and make sure there are no older breakpoints (set on resources that are not part of the current debugging context).
- Make sure you open the modules from the context of the application server that does the debugging and place breakpoints there.

Peculiarities and Limitations of the MarkLogic Debugger

MarkLogic debugger has the following peculiarities and limitations:

- Debugging support is only available for MarkLogic server versions 4.0 or newer.
- For MarkLogic server versions 4.0 or newer, there are three XQuery syntaxes that are supported: '0.9-ml' (inherited from MarkLogic 3.2), '1.0-ml', and '1.0'.
- All declared variables are presented as strings. The Value column of the Variables view contains the expression from the variable declaration. It can be evaluated by copying the expression with the Copy button.
value action from the contextual menu of the Variables view (on page 2119) and pasting it in the XWatch view (on page 2111).

- There is no support for output to source mapping (on page 2122).
- There is no support for showing the trace (on page 2116).
- You can only set breakpoints (on page 2109) in imported modules in one of the following cases:
  - When you open the module from the context of the application server involved in the debugging, using the Data Source Explorer view (on page 2015).
  - When the debugger automatically opens the modules in the Editor.
- No breakpoints (on page 2125) are set in modules from the same server that are not involved in the current debugging session.
- No support for profiling (on page 2126) when an XQuery transformation is executed in the debugger.

### Debugging with Berkeley DB XML (Deprecated)

The Berkeley DB XML database added a debugging interface starting with version 2.5. The current version is supported in the Oxygen XML Editor XQuery Debugger. The same restrictions and peculiarities (on page 2056) apply for the Berkeley debugger as for the MarkLogic debugger.

### Integration with Microsoft SharePoint

**Restriction:**
The SharePoint integration is only available in the Enterprise edition of Oxygen XML Editor.

Oxygen XML Editor provides support for browsing and managing SharePoint connections in the Data Source Explorer view (on page 2015) and there is also a specialized SharePoint Browser view (on page 2083). You can easily create new resources on the repository, copy and move them using contextual actions or the drag and drop support, or edit and transform the documents in the editor.

There are two types of integrations that are possible:

- **SharePoint Online** - A new implementation that uses the SharePoint REST API v.2 and it supports OAuth credentials (access tokens). If you need authentication, you must use this type of integration.
- **SharePoint** - The older implementation that was implemented using SharePoint Web Services (now deprecated) and it does NOT support OAuth credentials (access tokens).
How to Configure a SharePoint Connection

By default, Oxygen XML Editor contains built-in data source drivers for SharePoint. Use this data source to create a connection to a SharePoint server that will be available in the Data Source Explorer view or SharePoint Browser view (on page 2083).

There are two types of possible SharePoint connections:

- **SharePoint Online** - A new implementation that uses the SharePoint REST API v.2 and it supports OAuth credentials (access tokens). If you need authentication, you must use this type of integration.
- **SharePoint** - The older implementation that was implemented using SharePoint Web Services (now deprecated) and it does NOT support OAuth credentials (access tokens).

**SharePoint Online Connection**

To configure a SharePoint connection, follow these steps:

1. Open the Connection dialog box using one of these methods:
   - Select **New SharePoint Online Connection** from the Settings drop-down menu in the SharePoint Browser view (or using the quick action (on page 2083)).
   - Open the Preferences dialog box (Options > Preferences) (on page 124), go to Data Sources. Select SharePoint Online in the Data Sources pane and in the Connections pane, click the New button.
2. Enter a unique name for the connection.
3. Make sure SharePoint Online is selected in the Data Source combo box.
4. Enter the Tenant URL for your SharePoint repository and click OK.
In the **SharePoint Browser view** *(on page 2083)*, you can select your connection using the **Site** drop-down menu. Then select **Log in with Microsoft account** from the left pane (or user drop-down menu on the right side of the view) to open Microsoft’s log in page in your default browser. Once authenticated, your repository content should be displayed in the view.

**Note:**

If you are still logged in when you close Oxygen XML Editor, the authentication persists the next time the application is started. If, for some reason, the authentication fails to recover the access token, an error is displayed in the **Results** pane at the bottom of the application. If this happens, you need to re-authenticate Oxygen XML Editor by using the **Log in with Microsoft account** action.

**SharePoint Connection (Older Version)**

To configure a SharePoint connection, follow these steps:

1. Open the **Connection** dialog box using one of these methods:
   - Select **New SharePoint Connection** from the **Settings** drop-down menu in the **SharePoint Browser** view (or using the quick action *(on page 2083)*).
   - Open the **Preferences** dialog box (**Options** > **Preferences**) *(on page 124)*, go to Data Sources. Select **SharePoint** in the **Data Sources** pane and in the **Connections** pane, click the **New** button.
2. Enter a unique name for the connection.
3. Make sure **SharePoint** is selected in the **Data Source** combo box.
4. Fill-in the connection details:
   a. Enter the **SharePoint URL** for your SharePoint repository.
   b. Set the server domain in the **Domain** field. If you are using a SharePoint 365 account, leave this field empty.
   c. Set the user name to access the SharePoint repository in the **User** field.
   d. Set the password to access the SharePoint repository in the **Password** field.
5. Click **OK**.

**SharePoint Browser View**

The **SharePoint Browser** view allows you to connect to a SharePoint repository and perform SharePoint-specific actions on the available resources. To display this view, go to **Window** > **Show View** > **SharePoint Browser**.

**Getting Started**

When you first open the view, it includes some quick actions to help you **get connected to your SharePoint repository** *(on page 2082)* (these actions are also available in the **Settings** drop-down menu).
Figure 542. SharePoint Browser View Quick Actions

The SharePoint Browser View Interface

The header stripe of the SharePoint Browser view includes:

**Site drop-down menu**

Use this drop-down to select and connect to an already defined SharePoint connection (on page 2082).

**User drop-down menu**

This drop-down includes:

- **Log in with Microsoft account** (if logged out) - Opens the Microsoft login page in your default browser and authenticates Oxygen XML Editor.
- **Log out** (if logged in) - Logs out of your Microsoft account and the authorization between Oxygen XML Editor and Microsoft is revoked.
- **Help** - Opens the online user guide to a topic relevant to the current context.

**Settings drop-down menu**

This drop-down includes:

- **New SharePoint Online Connection** - Opens the Connection dialog box with the SharePoint Online data source automatically selected.
- **New SharePoint Connection** - Opens the Connection dialog box with the SharePoint (older version) data source automatically selected.
- **Configure Database Sources** - Opens the Database Sources preferences page where you can configure your SharePoint connection.
- **Layout** - Use this option to choose the layout for the view. You can choose between: Automatic, Vertical, and Horizontal.

Once you are connected, the view is separated into two panes. The left pane is a navigation area that presents the SharePoint site structure in a tree-like fashion with the following node types:
When a folder node is selected in the left pane, the right pane displays the contents of the folder (either folders or files).

You can filter and sort the displayed items. To display the available filters of a column, click the filter widget located on the column header. You can apply multiple filters at the same time.

![Figure 543. Column Filter](image)

Checking Documents In and Out

To check out a document from the server, right-click the file and select **Check Out**. You can discard the previous checkout operation, making the file available for editing to other users, by selecting **Discard Check Out**.

To check in a document that has been checked out, right-click the file and select **Check In**. For SharePoint Online connections, you only have the option to enter a comment and click the **Check In** button to process it. For SharePoint (older version), you can also choose the check in type (**Minor Version**, **Major Version**, or **Overwrite**).

Related Information:
- How to Configure a SharePoint Connection *(on page 2082)*

SharePoint Contextual Menu Actions

While browsing SharePoint connections in the **Data Source Explorer view** *(on page 2015)* or the **SharePoint Browser view** *(on page 2083)*, the following contextual menu actions are available, depending on the type of node:

- **Connection Nodes** *(Data Source Explorer view only)*
  - **Configure Database Sources**
    
    Opens the **Data Sources preferences page** *(on page 276)* where you can configure both data sources and connections.
  
  **Log in with Microsoft Account** *(when not connected)*
Opens the Microsoft login page in your default browser and authenticates Oxygen XML Editor.

**Disconnect (when connected)**

Stops the connection.

**Refresh**

Performs a refresh on the selected node.

**Find/Replace in Files**

Opens the Find/Replace in Files dialog box (on page 436) that allows you to find and replace text in multiple files from the connection.

**Site Nodes / Sub-site Nodes**

**Copy location**

Allows you to copy (to the clipboard) an application-specific URL for the resource that can then be used for various actions, such as opening or transforming the resources.

**Refresh**

Performs a refresh on the selected node.

**Find/Replace in Files**

Opens the Find/Replace in Files dialog box (on page 436) that allows you to find and replace text in multiple files from the connection.

**Folder Level Nodes**

**New File or New Document**

Creates a new file on the connection, in the current folder.

**New Folder**

Creates a new folder on the connection.

**Import Folders**

Imports folders on the server.

**Import Files**

Allows you to add a new file on the connection, in the current folder.

**Rename**

Renames the current resource

**Delete**

Deletes the current container.
 Performs a refresh on the selected node.

Find/Replace in Files (Data Source Explorer view only)

Opens the Find/Replace in Files dialog box (on page 436) that allows you to find and replace text in multiple files from the connection.

Resource Level Nodes

Open

Opens the selected resource in the editor.

Open in System Application (Data Source Explorer view only)

When you use this action, Oxygen XML Editor downloads the selected resource to a local temporary folder and opens the selected resource in the system application that is currently set as the default application associated with that type of resource. You can then edit the resource, save it, and when you switch the focus back to the Data Source Explorer view, Oxygen XML Editor will detect that there was a change and will ask if you want to upload the edited resource to the server.

Copy location

Allows you to copy (to the clipboard) an application-specific URL for the resource that can then be used for various actions, such as opening or transforming the resources.

Check Out

Checks out the selected document on the server.

Check In

Checks in the selected document on the server. This action opens the Check In dialog box. For SharePoint Online connections, you only have the option to enter a comment and click the Check In button to process it.

For SharePoint (older version), the following options are available:

- **Minor Version** - Increments the minor version of the file on the server.
- **Major Version** - Increments the major version of the file on the server.
- **Overwrite** - Overwrites the latest version of the file on the server.
- **Comment** - Allows you to add a comment for a file that you check in.

Discard Check Out

Discards the previous checkout operation, making the file available to other users.

Important:

Due to some API restrictions, the Discard Checkout action may not work when SharePoint Online connections are made directly to a sub-site.
Rename

Renames the current resource

Delete

Deletes the current container.

Refresh

Performs a refresh on the selected node.

Compare

Compares two selected resources using the Compare Files tool (on page 474).

Browsing for Remote Files with SharePoint Online

The Open URL dialog box (used for browsing remote files) includes support for connecting to a SharePoint Online server, with controls similar to the SharePoint Browser view (on page 2083). To open this dialog box, go to File > Open URL (or click the Open URL toolbar button), then choose the Browse for remote file option from the drop-down menu.

Figure 544. Open URL Dialog Box for SharePoint Online

The displayed dialog box is composed of the following:

Server Type

Specifies the type of server (SharePoint Online in this case).
Note:
If you select SharePoint On-Premises, the controls are similar to those for WebDav and other servers (on page 387).

Server URL
Specifies the protocol (HTTP, HTTPS or FTP) and the host name or IP of the server.

User drop-down menu
This drop-down includes:

- **Log in with Microsoft account** (if logged out) - Opens the Microsoft login page in your default browser and authenticates Oxygen XML Editor.
- **Log out** (if logged in) - Logs out of your Microsoft account and the authorization between Oxygen XML Editor and Microsoft is revoked.
- **Help** - Opens the online user guide to a topic relevant to the current context.

Browse
Use this button to retrieve the data from the server. Once the authentication completes, the files from the server will be available in the dialog box.

File URL
You can use this combo box to directly specify the URL to be opened or saved. This combo box also displays the current selection when the user changes selection by browsing the tree of folders and files on the server.

MS Azure Active Directory Authentication
It is possible to use your MS Azure Active Directory credentials for SharePoint authentication. To configure the application to use your client ID and client secret, set the following Oxygen system properties (on page 333):

- `com.oxygenxml.azure.active.directory.client.id` - Specifies a custom client ID.
- `com.oxygenxml.azure.active.directory.client.secret` - Specifies a custom client secret.

Your application should allow the following API permissions:

Table 44. API Permissions - Microsoft Graph

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
<th>Admin consent</th>
</tr>
</thead>
<tbody>
<tr>
<td>email</td>
<td>Delegated</td>
<td>View users’ email address</td>
<td>No</td>
</tr>
<tr>
<td>Files.ReadWrite.All</td>
<td>Delegated</td>
<td>Have full access to all files user can access</td>
<td>No</td>
</tr>
</tbody>
</table>
### Table 44. API Permissions - Microsoft Graph (continued)

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
<th>Admin consent</th>
</tr>
</thead>
<tbody>
<tr>
<td>offline_access</td>
<td>Delegated</td>
<td>Maintain access to data you have given it access to</td>
<td>No</td>
</tr>
<tr>
<td>openid</td>
<td>Delegated</td>
<td>Sign users in</td>
<td>No</td>
</tr>
<tr>
<td>profile</td>
<td>Delegated</td>
<td>View users’ basic profile</td>
<td>No</td>
</tr>
<tr>
<td>Sites.ReadWrite.All</td>
<td>Delegated</td>
<td>Edit or delete items in all site collections</td>
<td>No</td>
</tr>
<tr>
<td>User.Read</td>
<td>Delegated</td>
<td>Sign in and read user profile</td>
<td>No</td>
</tr>
</tbody>
</table>

### Table 45. API Permissions - SharePoint

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
<th>Admin consent</th>
</tr>
</thead>
<tbody>
<tr>
<td>AllSites.Read</td>
<td>Delegated</td>
<td>Read items in all site collections</td>
<td>No</td>
</tr>
<tr>
<td>AllSites.Write</td>
<td>Delegated</td>
<td>Read and write items in all site collections</td>
<td>No</td>
</tr>
</tbody>
</table>

Notice:
The Redirect URI should be set to: [http://localhost/oauth/redirect](http://localhost/oauth/redirect)
15. Importing Data

Computer systems and databases contain data in incompatible formats and exchanging data between these systems can be very time consuming. Converting the data to XML can greatly reduce the complexity and create data that can be read by various types of applications.

Oxygen XML Editor offers support for importing text files, MS Excel files, Database Data, and HTML files into XML documents. The XML documents can be further converted into other formats using the Transform features (on page 1432).

Import from Text Files

Oxygen XML Editor includes the possibility of importing text files (.txt or .csv file extensions) as XML documents.

To import a text file into an XML file, follow these steps:

1. Go to File > Import/Convert > Text File to XML. A Select text file dialog box is displayed.
2. Select the URL of the text file (.txt or .csv file extensions).
3. Select the encoding of the text file.
4. Click the Next button. The Import Criteria dialog box is displayed.
5. Configure the settings for the conversion.
   a. Select the **Field delimiter** for the import settings. You can choose between the following: **Comma**, **Semicolon**, **Tab**, **Space**, or **Pipe**.
   b. The **Import settings** section presents the input data in a tabular form. By default, all data items are converted to element content (`` symbol), but this can be overridden by clicking the individual column headers. Clicking a column header once causes the data from this column to be converted to attribute values (`` symbol). Clicking a second time causes the column data to be ignored (`` symbol) when generating the XML file. You can cycle through these three options by continuing to click the column header.
   c. **First row contains field names** - If this option is selected, the default column headers are replaced (where such information is available) by the content of the first row. In other words, the first row is interpreted as containing the field names. The changes are also visible in the preview panel.
d. **Customize** - This button opens a **Presentation Names** dialog box that allows you to edit the name, XML name, and conversion criterion for the root and row elements. The XML names can be edited by double-clicking the desired item and entering the label. The conversion criteria can also be modified by selecting one of the following options in the drop-down menu: `ELEMENT`, `ATTRIBUTE`, or `SKIPPED`.

e. **Import Settings** - Clicking this button opens the **Import preferences page** (on page 266) that allows you to configure more import options.

f. The **XML Import Preview** panel contains an example of what the generated XML document looks like.

g. **Open in editor** - If selected, the new XML document created from the imported text file is opened in the editor.

h. **Save in file** - If selected, the new XML document is saved in the specified path.

6. Click **Import** to generate the XML document.

**Import from MS Excel Files**

Oxygen XML Editor provides several methods for importing MS Excel files into an XML file. The first method is to use the Oxygen XML Editor **Smart Paste mechanism** (on page 613) by simply copying data from Excel and pasting it into a document in **Author** mode (note that this is only supported in DITA, DocBook, TEI, JATS, and XHTML documents). You can also copy data from Excel and paste it into inserted cells in **Grid** mode, but this is a more manual process. If you want to import an entire Excel file, Oxygen XML Editor also offers a configurable import wizard that works with any type of XML document.

**Smart Paste Method in Author Mode**

If you are importing data into DITA, DocBook, TEI, JATS, or XHTML documents, you can open the Excel spreadsheet in your office application, copy its content, and simply paste it into your document in **Author** mode.

The Oxygen XML Editor **Smart Paste mechanism** (on page 613) will convert the pasted content to the equivalent XML markup and considers various pasting solutions to keep the resulting document valid, while preserving the original text styling (such as bold, italics, underline) and formatting (such as lists, tables, paragraphs).

**Grid Mode Method**

The **Grid** mode in Oxygen XML Editor displays all content in an XML document as a structured grid of nested tables and you can work with the cells in those tables much like you would with any spreadsheet application. When importing Excel data into **Grid** mode, you first need to insert new cells in the particular nested table and then you can paste data from Excel the same as you would in any table or spreadsheet.

1. Copy the particular cells from your Excel spreadsheet that you want to import into an XML file.
2. Switch to **Grid** mode in Oxygen XML Editor.
3. Expand the particular nodes and locate the nested table where you want to insert the copied cells.
4. Right-click a particular row or column where you want to insert the data and select **Insert row** or **Insert column**, depending on the structure of the copied cells.

5. Paste the copied cells from the clipboard into the newly inserted cells in **Grid** mode.

6. You may need to make some manual adjustments. For example, if the selection of copied cells contained an empty cell, Oxygen XML Editor might ignore that cell.

**Import Wizard Method**

To use the **Import** wizard to import an Excel file into an XML file, follow these steps:

1. Go to **File > Import/Convert > MS Excel file to XML**.
2. Select the URL of the Excel file. The sheets of the document you are importing are presented in the **Available Sheets** section of this dialog box.
3. Click the **Next** button to proceed to the next stage of the wizard.

![Figure 546. Import Wizard](image-url)
4. Configure the settings for the conversion. This stage of the wizard offers the following options:

**Import settings section**

Presents the input data in a tabular form. By default, all data items are converted to element content (symbol), but this can be overridden by clicking the individual column headers. Clicking a column header once causes the data from this column to be converted to attribute values (symbol). Clicking a second time causes the column data to be ignored (symbol) when generating the XML file. You can cycle through these three options by continuing to click the column header.

**First row contains field names**

If this option is selected, the default column headers are replaced (where such information is available) by the content of the first row. In other words, the first row is interpreted as containing the field names. The changes are also visible in the preview panel.

**Customize**

This button opens a **Presentation Names** dialog box that allows you to edit the name, XML name, and conversion criterion for the root and row elements. The XML names can be edited by double-clicking the desired item and entering the label. The conversion criteria can also be modified by selecting one of the following option in the drop-down menu: **ELEMENT**, **ATTRIBUTE**, or **SKIPPED**.

**Import Settings**

Clicking this button opens the **Import preferences page (on page 266)** that allows you to configure more import options.

**Import formatted data (as displayed in Excel)**

If this option is selected, the imported data retains the Excel data formatting (such as the representation of numeric values or dates). If deselected, the data formatting is not imported.

**XML Import Preview panel**

Contains an example of what the generated XML document will look like.

**Open in editor**

If selected, the new XML document created from the imported file is opened in the editor.

**Save in file**

If selected, the new XML document is saved in the specified path.

5. Click **Import** to generate the XML document.

For more information about exchanging data between Oxygen XML Editor and spreadsheet applications, watch our video demonstration:
Import Database Data as an XML Document

To import the data from a relational database table as an XML document, follow these steps:

1. Go to File > Import/Convert > Database Data to XML to start the Import wizard.

   This opens a Select database table dialog box that lists all the defined database connections:

   **Figure 547. Select Database Table Dialog Box**

   ![Select Database Table Dialog Box](image)

2. Select the connection to the database that contains the appropriate data.

   Only connections configured in relational data sources can be used to import data.

3. If you want to edit, delete, or add a data source or connection, click the Configure Database Sources button.

   The Preferences/Data Sources option page is opened.
4. Click **Connect**.
5. In the list of sources, expand a schema and choose the required table.
6. Click the **Next** button.

The **Import Criteria** dialog box is opened with a default query string in the **SQL Query** pane.

**Figure 548. Import from Database Criteria Dialog Box**

```
SELECT `Country`, `VAT_rate` FROM `oxygenxml`.`EU_VAT_Rates`
```

7. Configure the settings for the conversion.
   a. **SQL Preview** - If this button is pressed, the **Settings** pane displays the labels that are used in the XML document and the first five lines from the database. By default, all data items are converted to element content (\(<\times\) symbol), but this can be overridden by clicking the individual column headers. Clicking a column header once causes the data from this column to be converted to attribute values (\(<\equiv\) symbol). Clicking a second time causes the column data to be ignored (\(<\times\) symbol) when generating the XML file. You can cycle through these three options by continuing to click the column header.
b. **Customize** - This button opens a Presentation Names dialog box that allows you to edit the name, XML name, and conversion criterion for the root and row elements. The XML names can be edited by double-clicking the desired item and entering the label. The conversion criteria can also be modified by selecting one of the following option in the drop-down menu: ELEMENT, ATTRIBUTE, or SKIPPED.

c. **Import Settings** - Clicking this button opens the Import preferences page (on page 266) that allows you to configure more import options.

d. The XML Import Preview panel contains an example of what the generated XML document looks like.

e. **Open in editor** - If selected, the new XML document created from the imported file is opened in the editor.

f. **Save in file** - If selected, the new XML document is saved in the specified path.

g. **Generate XML Schema** - Allows you to specify the path of the generated XML Schema file.

8. Click **Import** to generate the XML document.

**Import from HTML Files**

Oxygen XML Editor offers two methods for importing HTML files into an XML document. The first method is to simply copy data from an HTML document and paste it into a document in Author mode, but this is only supported in DITA, DocBook, TEI, JATS, and XHTML documents. Oxygen XML Editor also offers a configurable import wizard that works with any type of XML document.

**Smart Paste Method**

If you are importing data into DITA, DocBook, TEI, JATS, or XHTML documents, you can open the HTML document in your web browser, copy its content, and paste it into your document in Author mode.

The Oxygen XML Editor Smart Paste mechanism (on page 613) will convert the pasted content to the equivalent XML markup and considers various pasting solutions to keep the resulting document valid, while preserving the original text styling (such as bold, italics, underline) and formatting (such as lists, tables, paragraphs).

**Import Wizard Method**

To use the Import wizard to import from HTML files, follow these steps:

1. Go to **File > Import/Convert > HTML File to XHTML**. The Import HTML to XHTML wizard is displayed.
2. Enter the URL of the HTML document.
3. Select the type of the resulting XHTML document:
   - XHTML5
   - XHTML 1.0 Transitional
   - XHTML 1.0 Strict
4. Click the **OK** button.
Result: The resulting document is an XHTML file containing a DOCTYPE declaration that references the XHTML DTD definition on the Web. The parsed content of the imported file is transformed to XHTML5, XHTML Transitional, or XHTML Strict depending on the option you chose.

Import Content Dynamically

Along with the built-in support for various useful URL protocols (such as HTTP or FTP), Oxygen XML Editor also provides special support for a convert protocol that can be used to chain predefined processors to dynamically import content from various sources.

A dynamic conversion URL chains various processors that can be applied, in sequence, on a target resource and has the following general syntax:

```
convert:/processor=xslt;ss=urn:processors:excel2d.xsl/processor=excel!/urn:files:my.xls
```

The previous example first applies a processor (excel) on a target identified by the identifier (urn:files:sample.xls) and converts the Excel™ resource to XML. The second applied processor (xslt) applies an XSLT stylesheet identified using the identifier (urn:processors:excel2d.xsl) over the resulting content from the first applied processor. These identifiers are all mapped to real resources on disk via an XML catalog that is configured in the application, as in the following example:

```
<catalog xmlns="urn:oasis:names:tc:entity:xmlns:xml:catalog">
  <rewriteURI uriStartString="urn:files:" rewritePrefix="./resources/"/>
  <rewriteURI uriStartString="urn:processors:" rewritePrefix="./processors/"/>
</catalog>
```

The target resource part of the conversion URL must always follow the !/ pattern. It can be any of the following:

- An absolute URL that points to a resource.
- An identifier that will be resolved to an actual resource via the XML Catalog (on page 3225) support in the application. In the example above, the urn:files:sample.xls target resource is resolved via the XML catalog.
- A relative location. This location can only be resolved to an actual resource URL when the application has enough information about the location where the URL is referenced.

For example, for a DITA map (on page 3218) with a `topicref` such as:

```
<topicref href="convert://.../processor=excel!/resources/sample.xls"/>
```

the resources/sample.xls path will be resolved relative to the DITA map location.

This type of URL can be opened in the application by using the Open URL action from the File menu. It can also be referenced from existing XML resources via xi:include or as a topic reference from a DITA map.

A GitHub project that contains various dynamic conversion samples for producing DITA content from various sources (and then publishing it) can be found here: https://github.com/oxygenxml/dita-glass.
Conversion Processors

A set of predefined conversion processors is provided in Oxygen XML Editor. Each processor has its own parameters that can be set to control the behavior of the conversion process. All parameters that are resolved to resources are passed through the XML catalog mapping.

The following predefined conversion processors are included:

- **xslt Processor** - Converts an XML input using the Saxon EE XSLT processor. The `ss` parameter indicates the stylesheet resource to be loaded. All other specified parameters will be set as parameters to the XSLT transformation.

  ```xml
  convert:/processor=xslt;ss=urn:processors:convert.xsl;p1=v1!/urn:files:sample.xml
  ```

- **xquery Processor** - Converts an XML input using the Saxon EE XQuery processor. The `ss` parameter indicates the XQuery script to be loaded. All other specified parameters will be set as parameters to the XSLT transformation.

  ```xml
  convert:/processor=xquery;ss=urn:processors:convert.xquery;p1=v1!/urn:files:sample.xml
  ```

- **excel Processor** - Converts an Excel™ input to an XML format that can later be converted by other piped processors. It has a single parameter `sn`, which indicates the name of the sheet that needs to be converted. If this parameter is missing, the XML will contain the combined content of all sheets included in the Excel™ document.

  ```xml
  convert:/processor=excel;sn=test!/urn:files:sample.xls
  ```

- **java Processor** - Converts an input to another format by applying a specific Java method. The `jars` parameter is a comma-separated list of JAR (on page 3220) libraries, or folders that libraries will be loaded from. The `ccn` parameter is the fully qualified name of the conversion class that will be instantiated. The conversion class needs to have a method with the following signature:

  ```java
  public void convert(String systemID, String originalSourceSystemID, 
  InputStream is, OutputStream os, LinkedHashMap<String, String> properties)
  throws IOException
  ```

  ```xml
  convert:/processor=java;jars=libs;ccn=test.JavaToXML!/urn:files:java/WSEditorBase.java
  ```

- **js Processor** - Converts an input to another format by applying a JavaScript method. The `js` parameter indicates the script that will be used. The `fn` parameter is the name of the method that will be called from the script. The method must take a string as an argument and return a string. If any of the parameters are missing, an error is thrown and the conversion stops.

  ```xml
  convert:/processor=js;js=urn:processors:md.js;fn=convertExternal!/urn:files:sample.md
  ```

- **json Processor** - Converts a JSON input to XML. It has no parameters.

  ```xml
  convert:/processor=json!/urn:files:personal.json
  ```

- **xhtml Processor** - Converts HTML content to well-formed XHTML. It has no parameters.

  ```xml
  convert:/processor=xhtml!/urn:files:test.html
  ```
• **wrap Processor** - Wraps content in a tag name making it well-formed XML. The `rn` parameter indicates the name of the root tag to use. By default, it is `wrapper`. The `encoding` parameter specifies the encoding that should be used to read the content. By default, it is `UTF8`. As an example, this processor can be used if you want to process a comma-separated values file with an XSLT stylesheet to produce XML content. The CSV file is first wrapped as well-formed XML, which is then processed with an `xslt` processor.

```
convert:/processor=wrap!/urn:files:test.csv
```

• **cache Processor** - Caches the converted content obtained from the original document to a temporary file. The cache will be used on subsequent uses of the same URL, thus increasing the speed for the application returning the converted content. If the original URL points to the local disk, the cache will be automatically invalidated when the original file content gets modified. Otherwise, if the original URL points to a remote resource, the cache will need to be invalidated by reloading ([Reload (F5)] from the toolbar) the URL content that is opened in the editor.

```
convert:/processor=cache/processor=xslt;...!/urn:files:test.csv
```

### Reverse Conversion Processors

All processors defined above can also be used for saving content back to the target resource if they are defined in the URL as reverse processors. Reverse processors are evaluated right to left. These reverse processors allow round-tripping content to and from the target resource.

As an example, the following URL converts HTML to DITA when the URL is opened using the `h2d.xsl` stylesheet and converts DITA to HTML when the content is saved in the application using the `d2h.xsl` stylesheet.

```
convert:/processor=xslt;ss=h2d.xsl/rprocessor=xslt;ss=d2h.xsl!/urn:files:sample.html
```

⚠️ **Important:**

If you are publishing a DITA map that has such conversion URL references inside, you need to edit the transformation scenario and set the value of the parameter `fix.external refs.com.oxygenxml` to `true`. This will instruct Oxygen XML Editor to resolve such references during a special pre-processing stage. Depending on the conversion, you may also require additional libraries to be added using the Libraries button in the Advanced tab of the transformation scenario.

**Related Information:**

[https://github.com/oxygenxml/dita-glass](https://github.com/oxygenxml/dita-glass)
16.

Debugging XSLT Stylesheets and XQuery Documents

Oxygen XML Editor includes a powerful debugging interface that helps you to detect and solve problems with XSLT and XQuery transformations.

**XSLT Debugger Perspective**

The XSLT Debugger perspective (on page 3221) allows you to detect problems in an XSLT transformation by executing the process step by step. To switch the focus to this perspective, select the XSLT Debugger button in the top-right corner of the interface or Window > Open perspective > XSLT Debugger.

**XQuery Debugger Perspective**

The XQuery Debugger perspective (on page 3221) allows you to detect problems in an XQuery transformation process by executing the process step by step in a controlled environment and inspecting the information provided in the special views. To switch the focus to this perspective, select the XQuery Debugger button in the top-right corner of the interface or Window > Open perspective > XQuery Debugger.

**XSLT/XQuery Debugging Overview**

The XSLT Debugger and XQuery Debugger perspectives (on page 3221) allows you to test and debug XSLT 1.0 / 2.0 / 3.0 stylesheets and XQuery 1.0 / 3.0 documents including complex XPath 2.0 / 3.0 expressions. The interface presents simultaneous views of the source XML document, the XSLT/XQuery document and the result document. As you go step by step through the XSLT/XQuery document the corresponding output is generated step by step, and the corresponding position in the XML file is highlighted. At the same time, special views provide various types of debugging information and events useful to understand the transformation process.

The following set of features allow you to test and solve XSLT/XQuery problems:

- Support for XSLT 1.0 stylesheets (using Saxon 6.5.5 and Xalan XSLT engines), XSLT 2.0 / 3.0 stylesheets and XPath 2.0 / 3.0 expressions that are included in the stylesheets (using Saxon 10.6 XSLT engine) and XQuery 1.0 / 3.0 (using Saxon 10.6 XQuery engine).
- Stepping capabilities: step in, step over, step out, run, run to cursor, run to end, pause, stop.
- Output to source mapping between every line of output and the instruction element / source context that generated it.
- Breakpoints (on page 2125) on both source and XSLT/XQuery documents.
- Call stack on both source and XSLT/XQuery documents.
- Trace history on both source and XSLT/XQuery documents.
- Support for XPath expression evaluation during debugging.
- Step into imported/included stylesheets as well as included source entities.
• Available templates and hits count.
• Variables view.
• Dynamic output generation.

For even more information, watch our video demonstration:

https://www.youtube.com/embed/m9d8c4V-LJw

Debugger Layout

The XML and XSL files are displayed in Text mode (on page 516). The other modes (Author mode (on page 354), Grid mode (on page 354)) are available only in the Editor perspective (on page 344).

The XSLT/XQuery Debugger perspective (on page 3221) contains the following components:

• Source Document View (XML) - Displays and allows the editing of XML files (documents).
• XSLT/XQuery Document View (XSLT/XQuery) - Displays and allows the editing of XSL files (stylesheets) or XQuery documents.
• Output View - Displays the output that results from inputting a document (XML) and a stylesheet (XSL) or XQuery document in the transformer. The transformation result is written dynamically while the transformation is processed (using the Run button on the Control toolbar (on page 2107)). Several actions are available in the contextual menu for this view, including Find/Replace, Copy, and Format and Indent. There are two types of output views: a Text view (with XML syntax highlights) and XHTML view. For large outputs, the XHTML view can be disabled (see Debugger Settings (on page 256)).
• Control Toolbar (on page 2105) - Contains a variety of actions to help you configure and control the debugging process.
• Information Views (on page 2108) - The information views at the bottom of the editor display various types of information to help you understand the transformation process.

Tip:
The Output view and the various other information views are dockable (on page 3218) so that you can configure the workspace according to your preferences.
XML documents and XSL stylesheets or XQuery documents that were opened in the Editor perspective are automatically sorted into the first two panes. When multiple files of each type are opened, the individual documents and stylesheets are separated using the familiar tab management system that you are used to in the Editor perspective. Selecting a tab brings the document or stylesheet into focus and enables editing without the need to go back to the Editor perspective.

In Debugger mode, the normal editor toolbar is not available. However, functions are still accessible from the Document menu and the contextual menus.

Bookmarks (on page 519) are replaced in the Debugger perspective by breakpoints (on page 2125).

During debugging, the current execution node is highlighted in both document (XML) and XSLT/XQuery views.

Related Information:
Steps in a Typical Debugging Process (on page 2121)
Identify the XSLT / XQuery Expression that Generated Particular Output (on page 2122)
Supported Processors for XSLT / XQuery Debugging (on page 2132)
Performance Profiling of XSLT Stylesheets and XQuery Documents (on page 2126)
Control Toolbar

The **Control** toolbar contains all the actions that you need to configure and control the debugging process. The following actions are described as they appear in the toolbar from left to right.

![Control Toolbar](image)

**XML source selector**

The current selection represents the source document used as input by the transformation engine. The selection list contains all open files (XML files being emphasized). This option allows you to use other file types also as source documents. In an XQuery debugging session this selection field can be set to the default value **NONE**, because usually XQuery documents do not require an input source.

**XSL / XQuery selector**

The current selection represents the stylesheet or XQuery document to be used by the transformation engine. The selection list contains all open files (XSLT / XQuery files being emphasized).

**Link with editor**

When selected, the XML and XSLT/XQuery selectors display the names of the files open in the central editor panels. This button is toggled off by default.

**Output selector**

The selection represents the output file specified in the associated transformation scenario. You can specify the path by using the text field, the [Insert Editor Variables](on page 323) button, or the [Browse](button).

**Configure parameters**

Opens a dialog box that allows you to configure the XSLT / XQuery parameters to be used by the transformation.

**Edit extensions**

Allows you to add and remove the Java classes and JARS (on page 3220) used as XSLT extensions.

**Turn on/off profiling**

 Enables / Disables current transformation profiling.

**Enable XHTML output**

 Enables the rendering of the output in the [XHTML output view](on page 2103) during the transformation process. For performance issues, disable XHTML output when working with very
large files. Note that only XHTML conformant documents can be rendered by this view. To view the output result of other formats, such as HTML, save the Text output area to a file and use an external browser for viewing.

When starting a debug session from the Editor perspective (on page 3221) by using the Debug Scenario action, the state of this toolbar button reflects the state of the Show as XHTML output option from the scenario.

راتيج Turn on/off output to source mapping

Enables or disables the output to source mapping between every line of output and the instruction element / source context that generated it.

利 Debugger preferences

Quick link to Debugger preferences page (on page 256).

XSLT / XQuery engine selector

Lists the processors available for debugging XSLT and XQuery transformations (on page 2132).

.EntityManager / XQuery engine advanced options

If Saxon HE/PE/EE is selected, you can click this button to open the Advanced Saxon Transformation Options page (on page 1468).

管理条例 Step into

Starts the debugging process and runs until the next instruction is encountered.

管理条例 Step over

Run until the current instruction and its sub-instructions are over. Usually this will advance to the next sibling instruction.

Figure 551. Step over

管理条例 Step out

Run until the parent of the current instruction is over. Usually this will advance to the next sibling of the parent instruction.
Run Shift + F5

Starts the debugging process. The execution of the process is paused when a breakpoint (on page 2109) is encountered or the transformation ends.

Run to cursor

Starts the debugging process and runs until one of the following conditions occur: the line of cursor is reached, a valid breakpoint (on page 2125) is reached or the execution ends.

Run to end

Runs the transformation until the end, without taking into account enabled breakpoints (on page 2125), if any.

Pause

Request to pause the current transformation as soon as possible.

Stop

Request to stop the current transformation without completing its execution.

Show current execution nodes

Reveals the current debugger context showing both the current instruction and the current node in the XML source. Possible displayed states:

- Entering (▲) or leaving (▼) an XML execution node.
- Entering (▲) or leaving (▼) an XSL execution node.
- Entering (▲) or leaving (▼) an XPath execution node.

Note:

When you set a MarkLogic server as a processor, the Show current execution nodes button is named Refresh current session context from server. Click this button to refresh the information in all the views.
Note:
For some of the XSLT processors (Saxon-HE/PE/EE) the debugger could be configured to step into the XPath expressions affecting the behavior of the following debugger actions: Step into, Step over or Step Out.

Related Information:
Advanced Saxon HE/PE/EE XQuery Transformation Options (on page 1483)

Debugging Information Views

The information views at the bottom of the editor is comprised of two panes that are used to display various types of information used to understand the transformation process. For each information type there is a corresponding tab. While running a transformation, relevant events are displayed in the various information views. This enables the developer to obtain a clear view of the transformation progress. By using the debug controls, developers can easily isolate parts of stylesheet. Therefore, they may be more easily understood and modified.

The information types include the following:

Left side information views

- **Breakpoints** view (on page 2109)
- **XWatch** view (on page 2111)
- **Context** view (on page 2110)
- **Messages** view (on page 2112) (XSLT only)
- **Variables** view (on page 2119)
- **Invocation Tree** view (on page 2128)

Right side information views

- **Stack** view (on page 2113)
- **Output Mapping Stack** view (on page 2114)
- **Trace** view (on page 2116)
- **Templates** view (on page 2117) (XSLT only)
- **Nodes/Values Set** view (on page 2118)
- **Hotspots** view (on page 2129)

Tip:
The information views are dockable (on page 3218) so that you can configure the workspace according to your preferences.
Breakpoints View

The Breakpoints view lists all breakpoints (on page 2125) that are set on open documents. If the view is not displayed, it can be opened by selecting it from the Window > Show View menu. Breakpoints can be inserted (on page 2126) in the XML source document or the XSLT/XQuery document in debugging sessions.

Once you insert a breakpoint, it is automatically added to the list in the Breakpoints view and you can edit its associated condition. A breakpoint can have an associated break condition that represents an XPath expression evaluated in the current debugger context. For them to be processed, their evaluation result should be a boolean value. A breakpoint with an associated condition only stops the execution of the Debugger if the breakpoint condition is evaluated as true.

Note: The breakpoints list is not deleted at the end of a transformation (it is preserved between debugging sessions).

The Breakpoints view contains the following columns:

- **Enabled** - If selected, the current condition is evaluated and taken into account.
- **Resource** - Resource file and number of the line where the breakpoint is set. The Entire path of resource file is available as tooltip.
- **Condition** - XSLT/XQuery expression to be evaluated during debugging. The expression will be evaluated at every debug step.

Clicking a record highlights the breakpoint line in the document.

The following actions are available in the contextual menu of the table:

- **Go to**
  
  Moves the cursor to the source of the breakpoint.

- **Run to Breakpoint**
Runs the debugger up to the point of this particular breakpoint and ignores the others (regardless of whether they were previously enabled or disabled).

**Enable**

Enables the breakpoint.

**Disable**

Disables the breakpoint. A disabled breakpoint will not be evaluated by the Debugger.

**Add**

Allows you to add a new breakpoint and breakpoint condition.

**Edit**

Allows you to edit an existing breakpoint.

**Remove**

Deletes the selected breakpoint.

**Enable all**

Enables all breakpoints.

**Disable all**

Disables all breakpoints.

**Remove all**

Removes all breakpoints.

Related Information:
Using Breakpoints (on page 2125)

**Context View**

The context node is valid only for XSLT debugging sessions and is a source node corresponding to the XSL expression that is evaluated. It is also called the context of execution. The context node implicitly changes as the processor hits various steps (at the point where XPath expressions are evaluated). This node has the same value as evaluating "." (dot) XPath expression in XWatch view (on page 2111). The value of the context node is presented as a tree in the Context view. If the view is not displayed, it can be opened by selecting it from the Window > Show View menu.
The context nodes are presented in a tree-like fashion. Nodes from a defined namespace bound to a prefix are displayed using the qualified name. If the namespace is not bound to a prefix, the namespace URI is presented before the node name. The value of the selected attribute or node is displayed in the right side panel. The **Context** view also presents the current mode of the XSLT processor if this mode differs from the default one.

The title bar displays the current element index and the number of elements that compose the current context (this information is not available if you choose Xalan or Saxon 6 as processing engine).

**XPath Watch (XWatch) View**

The **XWatch** view shows XPath expressions evaluated during the debugging process. If the view is not displayed, it can be opened by selecting it from the **Window > Show View** menu.

Expressions are evaluated dynamically as the processor changes its source context. When you type an XPath expression in the **Expression** column, Oxygen XML Editor supports you with syntax highlight and **content completion assistance** ([on page 894](#)).

**Figure 555. XPath Watch View**
Table 46. XWatch columns

<table>
<thead>
<tr>
<th>Column</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Expression</td>
<td>XPath expression to be evaluated (XPath 1.0 or 2.0 / 3.0 compliant).</td>
</tr>
<tr>
<td>Value</td>
<td>Result of XPath expression evaluation. Value has a type (see the possible values (on page 2119) in the Variables View (on page 2119)) section. For Node Set results, the number of nodes in the set is shown in parenthesis.</td>
</tr>
</tbody>
</table>

**Important:**

Notes about working with the XWatch view:

- Expressions that reference variable names are not evaluated.
- The expression list is not deleted at the end of the transformation (it is preserved between debugging sessions).
- To insert a new expression, click the first empty line of the Expression column and start typing. As an alternative, right-click and select the Add action. Press Enter on the cell to add and evaluate.
- To delete an expression, click its Expression column and delete its content. As an alternative, right-click and select the Remove action. Press Enter on the cell to commit changes.
- If the expression result type is a Node Set, click it (Value column) and its value is displayed in the Nodes/Values Set view (on page 2118).
- The Copy, Add, Remove and Remove All actions are available in every row's contextual menu.

**Messages View**

Using an xsl:message instruction is one way to signal special situations encountered during transformation as well as a raw way of doing the debugging. The Messages view is available only for XSLT debugging sessions and shows all xsl:message calls executed by the XSLT processor during transformation. If the view is not displayed, it can be opened by selecting it from the Window > Show View menu.

**Figure 556. Messages View**

<table>
<thead>
<tr>
<th>Message</th>
<th>Terminate</th>
<th>Resource</th>
</tr>
</thead>
<tbody>
<tr>
<td>Message 1</td>
<td>no</td>
<td>personal.xsl [line: 8]</td>
</tr>
<tr>
<td>Message 2</td>
<td>no</td>
<td>personal.xsl [line: 12]</td>
</tr>
<tr>
<td>Message 3</td>
<td>no</td>
<td>personal.xsl [line: 29]</td>
</tr>
</tbody>
</table>
Table 47. Messages columns

<table>
<thead>
<tr>
<th>Column</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Message</td>
<td>Message content.</td>
</tr>
<tr>
<td>Terminate</td>
<td>Signals whether or not the processor terminates the transformation once it encounters the message (yes/no respectively).</td>
</tr>
<tr>
<td>Resource</td>
<td>Resource file where <em>xsl:message</em> instruction is defined and the message line number. The complete path of the resource is available as tooltip.</td>
</tr>
</tbody>
</table>

The following actions are available in the contextual menu:

**Go to**
- Highlight the XSL fragment that generated the message.

**Copy**
- Copies to clipboard message details (system ID, severity info, description, start location, terminate state).

**Clear all**
- Removes all messages from the view.

**Important:**
- Clicking a record from the table highlights the *xsl:message* declaration line.
- Message table values can be sorted by clicking the corresponding column header. Clicking the column header switches the sorting order between: ascending, descending, no sort.

**Stack View**

The **Stack** view shows the current execution stack of both source and XSLT/XQuery nodes. If the view is not displayed, it can be opened by selecting it from the **Window > Show View** menu.

During the transformation, two stacks are managed. One for source nodes being processed and the other for XSLT/XQuery nodes being processed. Oxygen XML Editor shows both node types in one common stack. The source (XML) nodes are preceded by a red color icon while XSLT/XQuery nodes are preceded by a green color icon. The advantage of this approach is that you can always see the source scope on which an XSLT/XQuery instruction is executed (the last red color node on the stack). The stack is oriented upside down.
Figure 557. Stack View

The contextual menu contains one action: **Go to**, which moves the selection in the editor panel to the line containing the XSLT element that is displayed on the selected line from the view.

**Table 48. Stack Columns**

<table>
<thead>
<tr>
<th>Column</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>#</td>
<td>Order number, represents the depth of the node (0 is the stack base).</td>
</tr>
<tr>
<td>XML/XSL/XQuery Node</td>
<td>Node from source or stylesheet document currently being processed. One partic­ular stack node is the document root, noted as <strong>#document</strong>.</td>
</tr>
<tr>
<td>Attributes</td>
<td>Attributes of the node (a list of <em>id</em>=&quot;value&quot; pairs).</td>
</tr>
<tr>
<td>Resource</td>
<td>Resource file where the node is located. The entire path is available as tooltip.</td>
</tr>
</tbody>
</table>

**Important:**

Remarks:

- Clicking a record from the stack highlights that node's location inside resource.
- Using Saxon, the stylesheet elements are qualified with XSL proxy, while using Xalan you only see their names. (example: `xsl:template` using Saxon and `template` using Xalan).
- Only the Saxon processor shows element attributes.
- The Xalan processor shows also the built-in rules.

**Output Mapping Stack View**

The **Output Mapping Stack** view displays context data (on page 2122) and presents the XSLT templates/ XQuery elements that generated specific areas of the output. If the view is not displayed, it can be opened by selecting it from the **Window > Show View** menu.
The Go to action of the contextual menu takes you to the line that contains the XSLT element displayed in the Output Mapping Stack view.

Table 49. Output Mapping Stack Columns

<table>
<thead>
<tr>
<th>Column</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>#</td>
<td>The order number in the stack of XSLT templates/XQuery elements. Number 0 corresponds to the bottom of the stack in the status of the XSLT/XQuery processor. The highest number corresponds to the top of the stack.</td>
</tr>
<tr>
<td>XSL/XQuery Node</td>
<td>The name of an XSLT template/XQuery element that participated in the generation of the selected output area.</td>
</tr>
<tr>
<td>Attributes</td>
<td>The attributes of the XSLT template/XQuery node.</td>
</tr>
<tr>
<td>Resource</td>
<td>The name of the file containing the XSLT template/XQuery element.</td>
</tr>
</tbody>
</table>

Important:

Remarks:

- Clicking a record highlights that XSLT template definition/XQuery element inside the resource (XSLT stylesheet file/XQuery file).
- Saxon only shows the applied XSLT templates having at least one hit from the processor. Xalan shows all defined XSLT templates, with or without hits.
- The table can be sorted by clicking the corresponding column header. When clicking a column header the sorting order switches between: ascending, descending, no sort.
- Xalan shows also the built-in XSLT rules.
Trace View

Usually, the XSLT/XQuery processors signal the following events during transformation:

- Entering a source (XML) node.
- Leaving a source (XML) node.
- Entering an XSLT/XQuery node.
- Leaving an XSLT/XQuery node.

The Trace view catches all of these events, so you can see how the process evolved. If the view is not displayed, it can be opened by selecting it from the Window > Show View menu.

The red icon lines denote source nodes while the green icon lines denote XSLT/XQuery nodes. It is possible to save the element trace in a structured XML document (using the Export to XML action in the contextual menu). Thus, you have the possibility of comparing the trace results from multiple debug sessions.

Figure 559. Trace History View

The contextual menu contains the following actions:

Go to

Moves the selection in the editor panel to the line containing the XSLT element or XML element that is displayed on the selected line from the view;

Export to XML

Saves the entire trace list in XML format.
Table 50. Trace History Columns

<table>
<thead>
<tr>
<th>Column</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Depth</td>
<td>Shows you how deep the node is nested in the XML or stylesheet structure. The bigger the number, the more nested the node is. A depth 0 node is the document root.</td>
</tr>
<tr>
<td>XML/XSLT/XQuery Node</td>
<td>Represents the node from the processed source or stylesheet document. One particular node is the document root, noted as #document. Every node is preceded by an arrow that represents what action was performed on it (entering or leaving the node).</td>
</tr>
<tr>
<td>Attributes</td>
<td>Attributes of the node (a list of id=&quot;value&quot; pairs).</td>
</tr>
<tr>
<td>Resource</td>
<td>Resource file where the node is located. The complete path of the resource file is provided as tooltip.</td>
</tr>
</tbody>
</table>

**Important:**

Remarks:

- Clicking a record highlights that node's location inside the resource.
- Only the Saxon processor shows the element attributes.
- The Xalan processor shows also the built-in rules.

**Templates View**

The xsl:template is the basic element for stylesheets transformation. The Templates view is only available during XSLT debugging sessions and shows all xsl:template instructions used by the transformation. If the view is not displayed, it can be opened by selecting it from the Window > Show View menu.

Being able to see the number of hits for each of the templates allows you to get an idea of the stylesheet coverage by template rules with respect to the input source.

**Figure 560. Templates view**

The contextual menu contains one action: **Go to**, which moves the selection in the editor panel to the line that contains the XSLT template displayed on the selected line from the view.
### Table 51. Templates columns

<table>
<thead>
<tr>
<th>Column</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Match</td>
<td>The <code>match</code> attribute of the <code>xsl:template</code>.</td>
</tr>
<tr>
<td>Hits</td>
<td>The number of hits for the <code>xsl:template</code>. Shows how many times the XSLT processor used this particular template.</td>
</tr>
<tr>
<td>Priority</td>
<td>The template priority as established by XSLT processor.</td>
</tr>
<tr>
<td>Mode</td>
<td>The <code>mode</code> attribute of the <code>xsl:template</code>.</td>
</tr>
<tr>
<td>Name</td>
<td>The <code>name</code> attribute of the <code>xsl:template</code>.</td>
</tr>
<tr>
<td>Resource</td>
<td>The resource file where the template is located. The complete path of the resource file is available as tooltip.</td>
</tr>
</tbody>
</table>

**Important:**

Remarks:

- Clicking a record highlights that template definition inside the resource.
- Saxon only shows the applied templates having at least one hit from the processor. Xalan shows all defined templates, with or without hits.
- Template table values can be sorted by clicking the corresponding column header. When clicking a column header the sorting order switches between: ascending, descending, no sort.
- Xalan shows also the built-in rules.

### Nodes/Values Set View

The **Nodes/Values Set view** is always used in relation with the **Variables view** (on page 2119) and **XWatch view** (on page 2111). If the view is not displayed, it can be opened by selecting it from the **Window > Show View** menu. It shows an XSLT node set value in a tree form. This view is updated as a response to the following events:

- You click a variable that has a node set value in the **Variables** (on page 2119) or **XWatch view** (on page 2111).
- You click a tree fragment in the **Variables** (on page 2119) or **XWatch view** (on page 2111).
- You click an XPath expression evaluated to a node set in the **Variables** (on page 2119) or **XWatch view** (on page 2111).
The nodes / values set is presented in a tree-like fashion. The total number of items is presented in the title bar. Nodes from a defined namespace bound to a prefix are displayed using the qualified name. If the namespace is not bound to a prefix, the namespace URI is presented before the node name. The value of the selected attribute or node is displayed in the right side panel.

**Important:**

Remarks:

- For longer values in the right side panel, the interface displays it with an ellipsis (...) at the end. A more detailed value is available as a tooltip when hovering over it.
- Clicking a record highlights the location of that node in the source or stylesheet view.

**Variables View**

The **Variables** view displays variables and parameters (local and global), along with their values. If the view is not displayed, it can be opened by selecting it from the **Window > Show View** menu.

Variables and parameters play an important role during an XSLT/XQuery transformation. Oxygen XML Editor uses the following icons to differentiate variables and parameters:

- **V** - Global variable.
- **{V}** - Local variable.
- **P** - Global parameter.
- **{P}** - Local parameter.

The following value types are available:

- **Boolean**
- **String**
- **Date** - XSLT 2.0 / 3.0 only.
- **Number**
• Set
• Object
• Fragment - Tree fragment.
• Any
• Undefined - The value was not yet set, or it is not accessible.

Note:

When Saxon 6.5 is used, if the value is unavailable, then the following message is displayed in the Value field: "The variable value is unavailable".

When Saxon 9 is used:

◦ If the variable is not used, the Value field displays "The variable is declared but never used".
◦ If the variable value cannot be evaluated, the Value field displays "The variable value is unavailable".

• Document
• Element
• Attribute
• ProcessingInstruction
• Comment
• Text
• Namespace
• Evaluating - Value under evaluation.
• Not Known - Unknown types.

Figure 562. Variables View

Table 52. Variables Columns

<table>
<thead>
<tr>
<th>Column</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Name of variable / parameter.</td>
</tr>
</tbody>
</table>
The value of a variable (the **Value** column) can be copied to the clipboard for pasting it to other editor areas with the **Copy value** action from the contextual menu. This is useful if you have long and complex values that cannot be easily remembered just by looking at them once.

**Important:**

Remarks:

- Local variables and parameters are the first entries presented in the table.
- Clicking a record highlights the variable definition line.
- Variable values could differ depending on the transformation engine used or stylesheet version set.
- If the value of the variable is a node set or a tree fragment, clicking it causes the **Node Set view** to be shown with the corresponding set of values.
- Variable table values can be sorted by clicking the corresponding column header. Clicking the column header switches between the orders: ascending, descending, no sort.

### Multiple Output Documents in XSLT 2.0 and XSLT 3.0

For XSLT 2.0 and XSLT 3.0 stylesheets that store the output in multiple files by using the **xsl:result-document** instruction, the content of the file created in this way is displayed in an output view after the transformation is finished. There is one view for each **xsl:result-document** instruction so that the output is not mixed while still being presented in multiple views.

### Steps in a Typical Debugging Process

Depending on your situation and needs, the debugging process might be more complex, but the following procedure is an example of a typical debugging process:

1. Open the source XML document and the XSLT/XQuery document.
2. If you are in the **Editor perspective** (on page 3221), switch to the XSLT Debugger or XQuery Debugger perspective (on page 3221) with one of the following actions:
   - Select **Window > Open perspective > XSLT Debugger/XQuery Debugger** or the **XSLT Debugger/XQuery Debugger** button in the top-right corner of the interface.
   - Select **Document > XML Document > Debug scenario** or use the **Debug scenario** action on the toolbar. This action initializes the Debugger perspective (on page 3221) with the
parameters of the transformation scenario. Any modification applied to the scenario parameters (the transformer engine, XSLT parameters, transformer extensions, etc.) will be saved back in the scenario when exiting from the Debugger perspective.

3. Select the source XML document in the XML source selector of the Control toolbar (on page 2105). In the case of XQuery debugging, if your XQuery document has no implicit source, set the source selector value to NONE.

4. Select the XSLT/XQuery document in the XSL/XQuery selector of the Control toolbar (on page 2105).

5. Set XSLT/XQuery parameters using the Configure parameters button on the Control toolbar (on page 2105).

6. Set one or more breakpoints (on page 2125).

7. Step through the stylesheet using the following buttons available on the Control toolbar (on page 2106):
   - Step into
   - Step over
   - Step out
   - Run
   - Run to cursor
   - Run to end
   - Pause
   - Stop

8. Examine the data in the information views to find the bug in the transformation process.
   For more information about fixing bugs in the transformation, see: Identify the XSLT / XQuery Expression that Generated Particular Output (on page 2122).

Related Information:
Identify the XSLT / XQuery Expression that Generated Particular Output (on page 2122)

Identify the XSLT / XQuery Expression that Generated Particular Output

To quickly spot the XSLT templates or XQuery expressions with problems, it is important to know what XSLT template in the XSLT stylesheet (or XQuery expression in the XQuery document) and what element in the source XML document generated a specified area in the output.

Some of the debugging capabilities (for example, Step in) can be used for this purpose. Using Step in, you can see how output is generated and link it with the XSLT/XQuery element being executed in the current source context. However, this can become difficult on complex XSLT stylesheets or XQuery documents that generate a large output.

You can click particular text in the Output view or XHTML output view and the editor will select the XML source context and the XSLT template/XQuery element that generated that text. Also, inspecting the whole
stack of XSLT templates/XQuery elements that determined the state of the XSLT/XQuery processor at the moment of generating the specified output area speeds up the debugging process.

This is an example of a typical procedure for identifying an expression that generated particular output:

1. Switch to the XSLT Debugger or XQuery Debugger perspective (on page 3221) with one of the following actions:
   ◦ Select Window > Open perspective > XSLT Debugger/XQuery Debugger or the XSLT Debugger/XQuery Debugger button in the top-right corner of the interface.
   ◦ Select Document > XML Document > Debug scenario or use the Debug scenario action on the toolbar. This action initializes the Debugger perspective (on page 3221) with the parameters of the transformation scenario. Any modification applied to the scenario parameters (the transformer engine, XSLT parameters, transformer extensions, etc.) will be saved back in the scenario when exiting from the Debugger perspective.

2. Select the source XML document in the XML source selector of the Control toolbar (on page 2105). In the case of XQuery debugging, if your XQuery document has no implicit source, set the source selector value to NONE.

3. Select the XSLT/XQuery document in the XSL/XQuery selector of the Control toolbar (on page 2105).

4. Select the appropriate engine in the XSLT/XQuery engine selector of the Control toolbar (on page 2106).

5. Set XSLT/XQuery parameters using the Configure parameters button on the Control toolbar (on page 2105).

6. Apply the XSLT stylesheet or XQuery transformation using the Run to end button that is available on the Control toolbar (on page 2107).

7. Inspect the mapping by clicking a section of the output in either the Text tab or XHTML tab of the Output view (on page 348).
Figure 563. XHTML Output to Source Mapping
This action will highlight the XSLT / XQuery element and the XML source context. This XSLT template/XQuery element that is highlighted in the XSLT/XQuery editor represents only the top of the stack of XSLT templates/XQuery elements that determined the state of the XSLT/XQuery processor at the moment of generating the clicked output section. In the case of complex transformations, inspecting the whole stack of XSLT templates/XQuery elements speeds up the debugging process. This stack is available in the Output Mapping Stack view (on page 2114).

Related Information:
- Output Mapping Stack View (on page 2114)
- Trace View (on page 2116)
- Templates View (on page 2117)

Using Breakpoints

The Oxygen XML Editor XSLT/XQuery Debugger allows you to interrupt XSLT/XQuery processing to gather information about variables and processor execution at particular points. To ensure breakpoints are persistent between work sessions, they are saved at project level. You can set a maximum of 100 breakpoints per project.
Inserting Breakpoints

To insert a **breakpoint**, follow these steps:

1. Click the line where you want to insert the **breakpoint** in the XML source document or the XSLT/XQuery document. **Breakpoints** are automatically created on the ending line of a start tag, even if you click a different line.
2. Click the vertical stripe on the left side of the editor panel or use **Shift+F7**.

**Result:**

Once you insert a **breakpoint**, it is automatically added to the list in the **Breakpoints** view and you can edit its associated **condition**. A **breakpoint** can have an associated break condition that represents an XPath expression evaluated in the current debugger context. For them to be processed, their evaluation result should be a boolean value. A **breakpoint** with an associated condition only stops the execution of the Debugger if the **breakpoint condition** is evaluated as **true**.

![Figure 565. Example: Breakpoints](image)

Removing Breakpoints

To remove a **breakpoint**, click its icon (🔴) in the vertical stripe on the left side of the editor panel or right-click the **breakpoint** and select **Remove** or **Remove all**.

Related Information:

Breakpoints View *(on page 2109)*

Performance Profiling of XSLT Stylesheets and XQuery Documents

Whether you are trying to identify a performance issue that is causing your production XSLT/XQuery transformation to not meet customer expectations or you are trying to proactively identify issues prior to deploying your XSLT/XQuery transformation, using the XSLT/XQuery profiler feature is essential to helping you save time and ultimately ensure a better performing, more scalable XSLT/XQuery transformation.

The XSLT/XQuery profiling feature can use any available XSLT/XQuery processor that can be used for debugging and it is available from the debugging **perspective** *(on page 3221)*.
Enabling the Profiler

Enabling and disabling the profiler is controlled by the Profiler button from the debugger Control toolbar (on page 2105). The XSLT/XQuery profiler is off by default. This option is not available during a debugger session so you need to set it before starting the transformation. For information about a common debugging procedure, see Steps in a Typical Debugging Process (on page 2121).

Profiling Information Views

Immediately after enabling the profiler, two new information views are added to the current debugger information views (on page 2108):

- **Invocation tree** view (on page 2128) on left side
- **Hotspots** view (on page 2129) on right side

Profiling data is available only after the transformation ends successfully.

On the left side (Invocation tree view (on page 2128)), you can examine how style instructions are processed. This result view is also named call-tree, as it represents the order of style processing. The profiling result shows the duration time for each of the style-instruction including the time needed for its called children.

On the right side (Hotspots view (on page 2129)), you can immediately spot the time the processor spent in each instruction. As an instruction usually calls other instructions, the used time of the called instruction is extracted from the duration time of the caller (the hotspot only presents the inherent time of the instruction).

Source Backmapping

In either the **Invocation tree** (on page 2128) or **Hotspots** view (on page 2129), you can use the backmapping feature to find the XSLT stylesheet or XQuery expression definition. Clicking the selected item causes Oxygen XML Editor to highlight the XSLT stylesheet or XQuery expression source line where the instruction is defined.

---

Figure 566. Source Backmapping
Saving and Customizing Profiling Data

The profiling data can be saved (exported) into XML and HTML format. In either the Invocation tree (on page 2128) or Hotspots view (on page 2129), right-click anywhere in the view and select Export to XML or Export to HTML. The HTML report can be customized based upon the profiling raw data. When you select Export to HTML, Oxygen XML Editor will save it as XML and apply an XSLT stylesheet to render the report as XML. You can customize these stylesheets to suit your needs. By default, they are located in: [OXYGEN_INSTALL_DIR]/frameworks/profiler/.

Other Profiling Notes

- If you want to change the XSLT/XQuery profiler settings (on page 257), use the contextual menu and choose the corresponding View settings entry.
- Profiling exhaustive transformations may run into an OutOfMemory error due to the large amount of information being collected. If this is the case, you can close unused projects when running the profiling or use high values for Java VM options `-Xms` and `-Xmx`. If this does not help you can shorten your source XML file and try again.
- For more information about the XSLT/XQuery Profiler, watch our video demonstration:

https://www.youtube.com/embed/4ftHschjLqA

Invocation Tree View

The Invocation Tree view shows a top-down call tree that represents how XSLT instructions or XQuery expressions are processed. If the view is not displayed, it can be opened by selecting it from the Window > Show View menu.

Figure 567. Invocation Tree View

The entries in the invocation tree include a few possible icons that indicate the following:

- - Points to a call whose inherent time is insignificant compared to its total time.
- - Points to a call whose inherent time is significant compared to its total time (greater than 1/3rd of its total time).
Every entry in the invocation tree includes textual information that depends on the XSLT/XQuery profiler settings (on page 257):

- A percentage number of the total time that is calculated with respect to either the root of the tree or the calling instruction.
- A total time measurement in milliseconds or microseconds. This is the total execution time that includes calls into other instructions.
- A percentage number of the inherent time that is calculated with respect to either the root of the tree or the calling instruction.
- An inherent time measurement in milliseconds or microseconds. This is the inherent execution time of the instruction.
- An invocation count that shows how often the instruction has been invoked on this call-path.
- An instruction name that contains also the attributes description.

The Invocation Tree view also includes the following contextual menu actions:

Export to HTML

Selecting this option will save the profiling data as XML and then apply an XSLT stylesheet to render the report as HTML. These stylesheets are included in the subfolder: \[OXYGEN_INSTALL_DIR]/frameworks/profiler/. You can use them to customize your own report based on the profiling raw data.

Export to XML

Use this option to save the profiling data as an XML file in a specified location.

View settings

Opens the XSLT/XQuery Profiler preferences page (on page 257) that allows you to configure various profiling settings.

Hotspots View

The Hotspots view displays a list of all instruction calls that lie above the threshold defined in the XSLT/XQuery profiler settings (on page 257). If the view is not displayed, it can be opened by selecting it from the Window > Show View menu.
By opening a hotspot instruction entry, the tree of back-traces leading to that instruction call are calculated and shown.

Every hotspot is described by the values from the following columns:

- **Instruction** - The name of the instruction.
- **Time** - The inherent time in milliseconds or microseconds of how much time has been spent in the hotspot, along with a bar whose length is proportional to this value. All calls into this instruction are summed up regardless of the particular call sequence.
- **Hits** - The invocation count of the hotspot entry.

If you click the ⚠ handle on the left side of a hotspot, a tree of back-traces will be shown.

Every entry in the backtrace tree has textual information attached to it that depends on the XSLT/XQuery profiler settings (on page 257):

- A percentage number that is calculated with respect to either the total time or the called instruction.
- A time measured in milliseconds or microseconds of how much time has been contributed to the parent hotspot on this call-path.
- An invocation count that shows how often the hotspot has been invoked on this call-path.

**Note:**

This is not the number of invocations of this instruction.

- An instruction name that also contains its attributes.

The **Hotspots** view also includes the following contextual menu actions:

**Export to HTML**

Selecting this option will save the profiling data as XML and then apply an XSLT stylesheet to render the report as HTML. These stylesheets are included in the subfolder:
Oxygen XML Editor 24.1 | 16 - Debugging XSLT Stylesheets and XQuery Documents | 2131

[OXYGEN_INSTALL_DIR]/frameworks/profiler/. You can use them to customize your own report based on the profiling raw data.

Export to XML

Use this option to save the profiling data as an XML file in a specified location.

View settings

Opens the XSLT/XQuery Profiler preferences page (on page 257) that allows you to configure various profiling settings.

Debugging XSLT that Call Java Extensions

It is possible to debug an XSLT that calls Java extensions. This is achieved through a transformation scenario where the Java extensions are specified, and the debugging can be done based upon the same scenario.

To debug XSLT with Java extensions, follow this procedure:

1. Create an XSLT transformation scenario (on page 1510) for your XSLT document (select Configure Transformation Scenario(s) action from the toolbar, then click New, and select XSLT transformation).
2. In the New scenario dialog box, click the Extensions button (in the XSLT tab), specify the Java extensions (JAR libraries) that are needed, and click OK.
3. Once you are finished configuring the transformation scenario, click OK, then select Save and close.
4. Use the Debug scenario action on the toolbar and the debugging will be based upon the same transformation scenario you just configured and saved.

Tip:

You could achieve this during a typical debugging process (on page 2121) by specifying the Java extensions using the Edit extensions button on the debugger control toolbar (on page 2105).

Related Information:
Validating XSLT Stylesheets that Call Java Extensions (on page 889)

Debugging Java Extensions

The XSLT/XQuery debugger does not step into Java classes that are configured as XSLT/XQuery extensions of the transformation. To step into Java classes, inspect variable values, and set breakpoints (on page 2125) in Java methods, you can set up a Java debug configuration in an IDE (such as the Eclipse SDK) as described in the following steps:
1. Create a debug configuration.
   a. Set at least 256 MB as heap memory for the Java virtual machine (recommended 1024 MB) by setting the `-Xmx` parameter in the debug configuration (for example, `-Xmx1024m`).
   
   b. Make sure the `[OXYGEN_INSTALL_DIR]/lib/oxygen.jar` file and your Java extension classes are on the Java classpath.
      The Java extension classes should be the same classes that were set as an extension (on page 2105) of the XSLT/XQuery transformation in the debugging perspective (on page 3221).
   
   c. Set the class `ro.sync.exml.Oxygen` as the main Java class of the configuration.
      The main Java class `ro.sync.exml.Oxygen` is located in the `oxygen.jar` file.

2. Start the debug configuration.
   Now you can set breakpoints and inspect Java variables as in any Java debugging process executed in the selected IDE (Eclipse SDK, and so on.).

**Supported Processors for XSLT / XQuery Debugging**

The following built-in XSLT processors are integrated in the debugger and can be selected in the Control Toolbar (on page 2105):

- **Saxon 10.6 HE (Home Edition)** - a limited version of the Saxon 9 processor, capable of running XSLT 1.0, XSLT 2.0 / 3.0 basic and XQuery 1.0 transformations, available in both the XSLT debugger and the XQuery one,
- **Saxon 10.6 PE (Professional Edition)** - capable of running XSLT 1.0 transformations, XSLT 2.0 basic ones and XQuery 1.0 ones, available in both the XSLT debugger and the XQuery one,
- **Saxon 10.6 EE (Enterprise Edition)** - a schema-aware processor, capable of running XSLT 1.0 transformations, XSLT 2.0 / 3.0 basic ones, XSLT 2.0 / 3.0 schema-aware ones and XQuery 1.0 / 3.0 ones, available in both the XSLT debugger and the XQuery debugger,
- **Saxon 6.5.5** - capable of running only XSLT 1.0 transformations, available only in the XSLT debugger,
- **Xalan 2.7.2** - capable of running only XSLT 1.0 transformations, available only in the XSLT debugger.
17.
Framework and Author Mode Customization

This section contains information and tutorials about customizing the authoring experience through custom frameworks and customizing the Author editing mode through CSS styling or API extensions.

Creating and Configuring Custom Frameworks

Oxygen XML Editor includes built-in, configured frameworks (on page 3219) for DocBook, DITA, TEI, XHTML, and JATS, but you can also create your own customization to handle other types of documents. A common use-case is wanting to customize the interface to accommodate the needs of your authoring team.

Fully configuring a framework usually involves customizing CSS stylesheets, XML schemas, GUI components (menu actions, toolbars, inline components, content completion proposals, and more), configuring other more general settings, then bundling the framework to share with your team. The CSS and GUI components are used to customize the interface, while other general settings can be configured to accommodate custom document templates, XML catalogs, transformation scenarios, and more.

Advanced users who are familiar with API development can also create custom Author mode operations (on page 2177) for a particular framework.

This section includes information about numerous possibilities for creating and customizing a framework, and how to share your customization with others.

Tip:
A sample framework customization package is available that you can dabble with and use to help you get started. It can be downloaded from: https://www.oxygenxml.com/php/get_oxygen_sample_framework.php. The package includes a sample CSS file, XSL file, schema files, document templates, an XML catalog file, custom icons, and other resources.

Creating a Framework through the Configuration Dialog

The easiest way to create a custom framework (on page 3219) (document type) is by extending an existing built-in framework, such as DITA or DocBook, and then making modifications to it. You can then easily share the custom framework (on page 2288) with your team.

To create a custom framework by extending an existing one, follow these steps:
1. In a location where you have full write access, create a folder structure similar to this:
   `custom_frameworks/dita-extension`.

2. Open the Preferences dialog box (Options > Preferences) (on page 124) and go to Document Type Association > Locations (on page 139). Add the path to your `custom_frameworks` folder in the Additional frameworks directories list and click OK or Apply to save your changes.

3. Go to the Document Type Association preferences page (on page 138) and select an existing framework configuration (for example, DITA) and use the Extend button to create an extension for it.

   **Step Result:** This opens the Document Type Configuration dialog box (on page 140) where you can define the set of rules and settings for your custom framework.

4. Give the extension an appropriate name, select External for the Storage option, click the browsing button ( ) to specify the location of the custom directory you created in step 1.

5. Continue to configure the extension using the tabs on the bottom half of the dialog box. For details about each of those tabs, see the child topics in the Document Type Configuration dialog box (on page 140) section. For even more information about customizing the extended framework, see the various topics and tutorials in the Creating and Configuring Custom Frameworks (on page 2133) section. Make sure that you save any resources you reference in your framework configuration (CSS files, new document templates, schemas used for validation, catalogs, etc.) in your custom framework directory you created in step 1.

6. Click OK to close the configuration dialog box and then OK or Apply to save your changes.

   **Results:** You now have a fully functional framework that can be shared with others (on page 2288).

---

**Related Information:**

Sharing a Framework (on page 2288)

---

**Creating a Framework Using an Extension Script**

A custom framework (on page 3219) (document type) can be created using a special XML descriptor file, either from scratch or by extending an existing built-in framework (such as DITA or DocBook) and then making modifications to it. You can then easily share the custom framework (on page 2288) with your team.

The easiest way to create such a descriptor is to use the New document wizard (on page 368) and choose the Extend Framework Script or Create Framework Script template.

---

**Tip:**

To see a visual, detailed look at how to create frameworks with an extension script, watch our webinar: Creating Frameworks Using an Extension Script (some samples are also available on that events page).
Creating a Custom Framework Starting from an Existing Framework

To create a custom framework by extending an existing one, follow these steps:

1. In a location where you have full write access, create a folder structure similar to this:
   `custom_frameworks/dita-extension`.
2. Open the Preferences dialog box (Options > Preferences) (on page 124) and go to Document Type Association > Locations (on page 139). Add the path to your `custom_frameworks` folder in the Additional frameworks directories list and click OK or Apply to save your changes and close the dialog box.
3. Click the New button on the toolbar and select the Extend Framework Script template. Save it inside the previously configured framework path (e.g. `custom_frameworks/dita-extension`).
4. Set the `@base` attribute on the script element to the value of the name of the extended framework (e.g. DITA).

   **Note:**
   Removing the `@base` attribute will create a framework from scratch.

5. Edit the script as described in Framework Extension Script File (on page 2136).

To test your customization, open a document that matches the newly created framework and inspect how your settings apply or go to Options > Preferences > Document Type Association and inspect the resulting framework structure.

   **Note:**
   If you want to use the framework in an older Oxygen XML Editor version that does not have support for these scripts, you can compile the script to obtain the *.framework file by using the Compile Framework Extension script action from the contextual menu or by running the scripts/compileFrameworkScript.bat external tool.

Creating a Custom Framework Without a Base Framework

To create a custom framework without starting from an existing one, follow these steps:

1. In a location where you have full write access, create a folder structure similar to this:
   `custom_frameworks/dita-extension`.
2. Open the Preferences dialog box (Options > Preferences) (on page 124) and go to Document Type Association > Locations (on page 139). Add the path to your `custom_frameworks` folder in the Additional frameworks directories list and click OK or Apply to save your changes and close the dialog box.
3. Click the New button on the toolbar and select the Create Framework Script template. Save it inside the previously configured framework path, `custom_frameworks/dita-extension`.
4. Edit the script as described in Framework Extension Script File (on page 2136).
To test your customization, open a document that matches the newly created framework and inspect how your settings apply or go to Options > Preferences > Document Type Association and inspect the newly generated framework structure.

**Note:**
If you want to use the framework in an older Oxygen XML Editor version that does not have support for these scripts, you can compile the script to obtain the `.framework` file by using the **Compile Framework Extension script** action from the contextual menu or by running the scripts/compileFrameworkScript.bat external tool.

**Related information**
Sharing a Framework (on page 2288)

**Framework Extension Script File**

The framework extension file is used to describe a new framework configuration. Optionally, you can extend an existing built-in framework configuration (such as DITA or DocBook) and then make additions and changes to it.

The easiest way to create such a file is to use the **New document wizard (on page 368)** and choose the **Extend Framework Script** or **Create Framework Script** template.

The following examples assume that the newly created framework extends a built-in one.

**Basic Information**

Once you have created a new script file, you need to:

- Specify the name of the framework using the `<name>` element. Optionally, you can also add a description using the `<description>` element.
- If you want to extend an existing framework (such as DITA or DocBook), specify the name of the extended framework using the `@base` attribute on the `<script>` element.
- The `<priority>` element might be needed to instruct Oxygen XML Editor to use this new framework instead of the one being extended or another framework that matches the same document.

**Example: Extending the Built-in DITA Framework**

```
<script xmlns="http://www.oxygenxml.com/ns/framework/extend"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://www.oxygenxml.com/ns/framework/extend
        base="DITA">
  <name>My custom DITA framework</name>
  <description>A custom framework based on the built-in DITA framework</description>
```
-changing the association rules

Oxygen XML Editor identifies the type of a document when the document matches at least one of the association rules.

Example: Instructing the Built-in Associations to Inherit None to Add Your Own

```xml
<associationRules inherit="none">
  <addRule rootElementLocalName="concept"/>
  <addRule fileName="test.xml"/>
</associationRules>
```

-setting the initial editing mode

You can set the default editing mode that is loaded when a document is opened for the first time. For example, if the files are usually edited in the Author mode, you can set it as the initial page. For more details on the possible values, see Document Type Configuration Dialog Box - Initial Edit Mode (on page 141).

Example: Setting the Initial Editing Mode to Author

```xml
<initialPage>Author</initialPage>
```

-changing the classpath

The Classpath tab displays a list of folders and JAR (on page 3220) libraries that hold implementations for API extensions, implementations for custom Author mode operations, various resources (such as stylesheets), and framework (on page 3219) translation files. Oxygen XML Editor loads the resources looking in the folders in the order they appear in the list.

Example: Customizing and Extending the Classpath Inherited From the Base Framework

```xml
<classpath inherit="all">
  <!-- Contribute this resource before the ones inherited from the base framework because Oxygen loads the resources looking in the folders in the order they appear in the list. -->
  <addEntry path="${framework}/resources_2x" position="before"/>

  <removeEntry path="${baseFramework}/refactoring"/>
</classpath>
```

Note:

When removing entries with framework editor variables, take into consideration how they were added in the base framework.
• An entry present in the base framework with the path ${framework}/file can be removed using an identical path or by using the ${baseFramework} variable: ${baseFramework}/file.

• An entry present in the base framework with the path ${frameworkDir}/file can be removed using an identical path or by using the ${baseFrameworkDir} variable: ${baseFrameworkDir}/file.

Sharing a Plugin Class Loader

If your framework uses the same JAVA extensions/classes as a plugin (on page 3222), it is recommended that they share the same classloader. This way, the common classes are loaded by only one Class Loader and they will both use the same static objects and have the ability to cast objects between one another.

<classpath parentClassLoaderID="com.oxygenxml.git.plugin"/>

Changing XML Catalogs

For cases where you need to reference the location of a schema file from a remote web location and an internet connection may not be available, an XML Catalog (on page 3225) may be used to map the web location to a local file system entry.

Example: Customizing and Extending the XML Catalogs Inherited From the Base Framework

<xmlCatalogs inherit="all">
  <!-- Contribute this resource before the ones inherited from the base framework because Oxygen loads the resources looking in the folders in the order they appear in the list. -->
  <addEntry path="${framework}/catalog.xml" position="before"/>
  <removeEntry path="${baseFramework}/oldCatalog.xml"/>
</xmlCatalogs>

Changing the Document Templates

You can create your own custom document templates or remove templates inherited from the base framework.

Example: Customizing and Extending the XML Catalogs Inherited From the Base Framework

<documentTemplates inherit="all">
  <!-- Contribute this resource before the ones inherited from the base framework to make them appear first in the list. -->
  <addEntry path="${framework}/newTemplates" position="before"/>
</documentTemplates>
Adding New Transformation Scenarios and Removing Existing Ones

You can create new transformation scenarios and export them in one of the following locations:

- The Transformation Scenarios View (on page 1555).
- The Configure Transformation Scenario(s) Dialog Box (on page 1548).
- The Transformation Tab (on page 165).

Example: Importing New Transformation Scenarios

The @href attribute from the <addScenarios> element is used to point to the location of the scenarios export file. Relative paths are resolved relative to the script's location. The ${framework} editor variable also resolves to the script's location. You can also remove any scenario inherited from the base framework or set the default scenario (the one used when another specific scenario is not specified).

```xml
<transformationScenarios>
  <addScenarios href="scenarioExport.scenarios"/>
  <removeScenario name="DITA HTML5"/>
  <defaultScenarios>
    <name>DITA</name>
    <name>XML</name>
  </defaultScenarios>
</transformationScenarios>
```

Adding New Validation Scenarios and Removing Existing Ones

You can create new validation scenarios and export them in one of the following locations:

- The Configure Validation Scenario Dialog Box (on page 788).
- The Validation Tab (on page 166).

Example: Importing New Validation Scenarios

The @href attribute from the <addScenarios> element is used to point to the location of the scenarios export file. Relative paths are resolved relative to the script's location. The ${framework} editor variable also resolves to the script's location. You can also remove any scenario inherited from the base framework or set the default scenario (the one used when another specific scenario is not specified).

```xml
<validationScenarios>
  <addScenarios href="validationScenarioExport.scenarios"/>
  <removeScenario name="DITA"/>
  <defaultScenarios>
    <name>DITA Validation</name>
  </defaultScenarios>
</validationScenarios>
```
Customizing the Author Mode Through New CSS Files

The Author mode layout is driven by CSS rules. To customize it, you need to create new CSS files and add them in the new framework.

Example: Using Larger Fonts in Titles

```xml
<author>
  <css>
    <removeCss path="${framework}/base.css"/>
    <!--
    Adding CSS after the ones in the base gives the opportunity to
    override rules from previous CSSs.
    -->
    <addCss path="${framework}/titles.css" position="after"/>
  </css>
</author>
```

The ${framework}/titles.css file contains a rule like this:

```css
* [class~='topic/title'] {  
  font-size: larger;
}
```

Example: Creating an Alternate CSS That Activates When the User Selects it in the Styles Menu

```xml
<author>
  <css>
    <addCss path="${framework}/pink.css" title="Pink titles" alternate="true"/>
  </css>
</author>
```

The ${framework}/pink.css file contains a rule like this:

```css
* [class~='topic/title'] {  
  color: #FF1493;
}
```

Defining Author Actions for the New Framework

Create external author actions (on page 2149), save them in a specific subdirectory of your particular framework directory (on page 2152), and they will be loaded automatically.
Removing Author Actions from the Base Framework

Suppose that the base framework configuration defines some author actions (on page 2148) that are added in the main menu (on page 157), contextual menu (on page 157), toolbar (on page 160), or content completion window (on page 160). If you do not want to inherit one of these actions in the new framework and you also want to remove it from all the GUI elements, you can use the `<removeAction>` element:

```xml
<author>
  <authorActions>
    <removeAction id="action.to.remove"/>
  </authorActions>
</author>
```

Note:
If the new framework has an external author action (on page 2149) with the same ID as one of the actions specified in a `<removeAction>` element, the action will not be removed from the GUI elements (menus, toolbars, content completion window).

Author Toolbar Configuration

The Author mode-specific toolbars for the new framework can be customized by:

- Adding or removing actions from toolbars.
- Changing toolbar groups by adding or removing actions.
- Creating new toolbars and action groups.

Example: Customizing the Toolbar

```xml
<author>
  <toolbars>
    <toolbar>
      <!-- Remove an action inherited from the base framework. -->
      <removeAction id="bold"/>
      <!-- Insert an action into an existing group -->
      <group name="${i18n(link)}">
        <addAction id="insert.note"/>
      </group>
      <!-- Add actions, separators and new groups-->
      <separator/>
      <addAction id="insert.note"/>
      <group name="New group">
        <!-- Additional groups and actions can be added here -->
      </group>
    </toolbar>
  </toolbars>
</author>
```
Example: Creating a New Toolbar

A new toolbar is created if the `@name` attribute does not match a toolbar inherited from the base.

```xml
<author>
  <toolbars>
    <toolbar name="Extra Toolbar">
      <!-- Add actions, separators and new groups-->
      <separator/>
      <addAction id="insert(note")/>

      <group name="New group">
        <addAction id="insert(note")/>
        <addAction id="insert.table")/>
      </group>
    </toolbar>
  </toolbars>
</author>
```

Example: Adding an Action in the Toolbar at a Specific Location

You can insert items (actions or groups) relative to other items already present in the toolbar because they were inherited from the base framework configuration. The `@anchor` attribute specifies either the ID of an action or the name of a group already present in the toolbar and the `@position` attribute specifies whether the new item should be added before or after it.

Note:
If the `@anchor` attribute is missing, the entries will be added either first or last, according to `@position` value.
Author Menu and Contextual Menu Configuration

The Author mode-specific menus for the new framework can be customized by:

- Adding or removing actions and submenus.
- Changing existing submenus by adding or removing actions.

Example: Customizing the Contextual Menu

```xml
<contextualMenu>
   <!-- Add new actions and submenu -->
   <separator/>
   <addAction id="insert.table"/>
   <submenu name="Other actions">
      <addAction id="insert.note"/>
   </submenu>

   <!-- Contribute to an existing submenu -->
   <submenu name="${i18n(section)}">
      <addAction id="paragraph"/>
   </submenu>

   <!-- Remove a submenu inherited from the base framework. -->
   <removeSubmenu name="${i18n(link)}"/>
</contextualMenu>
```

Note:
The framework main menu is configured similarly, inside a <menu> container.

Tip:
You can inspect the names of the submenus inherited from the base framework in the Contextual Menu Subtab (on page 157) and Menu Subtab (on page 157).

Example: Adding an Action in the Contextual Menu at a Specific Location
You can insert new actions and submenus relative to other actions and submenus already present in the menu because they were inherited from the base framework configuration. The `@anchor` attribute specifies the ID of an Author mode action or a name of a submenu already present in the menu and the `@position` attribute specifies whether the new action should be added before or after it.

Note:
If the `@anchor` attribute is missing, the entries will be added either first or last, according to `@position` value.

```xml
<contextualMenu>
  <addAction id="insert.note" anchor="edit.image.map" position="before"/>

  <submenu name="Table menu" anchor="${i18n(insert)}" position="after">
    <addAction id="insert.table"/>
  </submenu>
</contextualMenu>
```

Configuring the Content Completion in Author Mode

You can replace content completion entries obtained from the associated schema with Author mode actions (on page 2148).

In the `<authorActions>` container, you can specify the Author mode actions to be contributed. Optionally, you can mark them as a replacement for an existing schema proposal with the `@replacedElement` attribute.

The `<schemaProposals>` element allows you to remove proposals detected from the associated schema through the `<removeProposal>` element. If some proposals were removed in the base framework configuration and you want them re-added, you can do so through the `<addProposal>` element.

Example: Customizing the Content Completion Assistant

```xml
<contentCompletion>
  <authorActions>
    <addAction id="insert.note" replacedElement="note" inCCWindow="true"/>
  </authorActions>

  <schemaProposals>
    <removeProposal renderName="table"/>
    <!-- The base framework removed the "list" element proposal. We want it back... -->
    <addProposal renderName="list"/>
  </schemaProposals>
</contentCompletion>
```
Using Framework Extension Points

The Extensions tab specifies implementations of Java interfaces used to provide advanced functionality to the document type. Libraries that contain the implementations must be present in the classpath of your framework (on page 2137). The Javadoc available at https://www.oxygenxml.com/InstData/Editor/SDK/javadoc/ contains details about how each API implementation functions.

Example: Setting a Custom Extensions Bundle

```xml
<extensionPoints>
  <extension
    name="extensionsBundleClassName"
    value="ro.sync.ecss.extensions.dita.map.DITAMapExtensionsBundle"/>
</extensionPoints>
```

Reusing Parts of the Script Using XInclude

Elements in the script can be specified in dedicated files that can then be referenced using XInclude in the script.

Example: Using XInclude to Reference Elements in the Script

```xml
<script xmlns="http://www.oxygenxml.com/ns/framework/extend">
  <name>New framework</name>
  <xi:include href="classpath.xml" xmlns:xi="http://www.w3.org/2001/XInclude"/>
</script>
```

Where the referenced classpath.xml has this content:

```xml
<classpath xmlns="http://www.oxygenxml.com/ns/framework/extend">
  <addEntry path="test.jar"/>
</classpath>
```

Customizing the Author Mode Editing Experience for a Framework

You can customize the editing experience in Author mode for you and any other user who shares the same framework. This includes the ability to configure actions, menus, toolbars, icons, structure insertion shortcuts, and content completion proposals specifically for a particular framework (on page 2177) (document type). Advanced users who are familiar with API development can also create custom Author mode operations (on page 2177) for a particular framework.

Configuring and Managing Multiple CSS Styles for a Framework

Oxygen XML Editor provides a Styles drop-down menu on the toolbar that allows you to select one main (non-alternate) CSS style (on page 3220) and multiple alternate CSS styles (on page 3216). This makes it easy to change the look of the document as it appears in Author mode.
An example of a common use case is when content authors want to use custom styling within a document. You can select a main CSS stylesheet that styles the whole document and then apply alternate styles, as layers, to specific parts of the document.

**Note:**
When altering a CSS file configured as a stylesheet for the current document framework, you can quickly check its effects in the Author mode by using the Reload document action that is available on the toolbar.

### Managing the CSS Styles

The main (on page 3220) and alternate (on page 3216) styles that are listed in the Styles drop-down menu can be controlled in the Document Type configuration dialog box (on page 140). To access it, follow these steps:

1. Open the Preferences dialog box (on page 124).
2. Go to Document Type Association.
3. Select the appropriate document type and click the Edit button.

**Important:**
If you do not have access rights to the folder where the framework (on page 3219) files are stored, you can either elevate read/write permissions on that framework folder or you can extend the framework and customize the CSS stylesheets in the extension. If you want to share the customized extension with the rest of your team, see Sharing the Extended Framework (on page 2288).

The CSS styles (CSS files) associated with the particular document type are listed in the CSS subtab (on page 146) of the Author tab.
You can Add, Edit, or Delete styles from this dialog box to manage the main (on page 3220) and alternate (on page 3216) styles associated to the particular document type. You can also change the order of the styles by using the Move Up and Move Down buttons. This also changes the order that they appear in the Styles drop-down menu. The alternate styles are combined with the main CSS sequentially, in the order that they appear in this list. Therefore, if the same style rules are included in multiple CSS files, the alternate style that is listed last in this list takes precedence, since it is the last one to be combined (applied as a layer).

The URI column shows the path of each CSS file. The names listed in the Styles drop-down menu match the values in the Title column. The value in the Alternate column determines whether it is a main or alternate CSS. If the value is no, it is a main CSS. If the value is yes, it is an alternate CSS and the style can be combined with a main CSS or other alternate styles when using the Styles drop-down menu.

Note:
To group alternate styles into categories (submenus), use a vertical bar character ( | ) in the Title column. You can use multiple vertical bars for multiple submenus. The text before each vertical bar will be rendered as the name of a submenu entry in the Styles drop-down menu, while the text after the final vertical bar is rendered as the name of the style inside the submenu.
Example: Suppose that you want to add two alternate stylesheets in separate submenus, with the Title column set to My Styles|User Assistance|Hints and My Styles|User Actions|Inline Actions, respectively.

<table>
<thead>
<tr>
<th>URI</th>
<th>Title</th>
<th>Alternate</th>
</tr>
</thead>
<tbody>
<tr>
<td>$(framework)/css/...</td>
<td>My Styles</td>
<td>User Assistance</td>
</tr>
<tr>
<td>$(framework)/css/...</td>
<td>My Styles</td>
<td>User Actions</td>
</tr>
</tbody>
</table>

Oxygen XML Editor will add a My Styles submenu with two submenus (User Assistance that contains the Hints style, and User Actions that contains the Inline Actions style) in the Styles drop-down menu.

The Enable multiple selection of alternate CSSs checkbox (on page 146) at the bottom of the pane must be selected for the alternate CSS styles (on page 3216) to be combined. They are applied like layers and you can activate any number of them. If this option is not selected, the alternate styles are treated like main CSS styles (on page 3220) and you can only select one at a time. By default, this option is selected. There are also a few options that allow you to specify how to handle the CSS if there are CSS styles specified in the document. You can choose to ignore or merge them.

The following rules apply for merging CSS styles:

- CSS files with the same title will be merged.
- CSS files without a title will contribute to all others.
- They are merged sequentially, in the order that they appear in the list.

Related information
Changing the Look of Documents in Author Mode Using the Styles Menu (on page 590)
CSS Subtab (on page 146)

Creating and Customizing Author Mode Actions for a Framework

There are several possibilities for creating new Author mode actions:

- You can create new actions for a framework or edit existing ones using the Actions subtab of the Document Type configuration dialog box (on page 140). In this case, the actions are stored internally in the *.framework file.
- You can export existing actions from the Document Type configuration dialog box (on page 140) into individual XML files or use a built-in template to create a new XML file that defines a single action. In
this case, the actions are stored externally as separate XML files. The benefits of using this approach are explained in the Creating or Editing Actions Using an Individual XML File for Each Action (on page 2149) section below.

Creating or Editing Actions Using the Document Type Configuration Dialog Box

To add or configure Author mode actions for a framework (on page 3219) (document type) using the Document Type configuration dialog box (on page 140), follow this procedure:

1. Open the Preferences dialog box (Options > Preferences) (on page 124), go to Document Types Association, and select the framework.
2. Select your framework and click the Edit button (or you can use the Duplicate or Extend button to create an extension of the framework (on page 2133)).
3. In the resulting Document Type configuration dialog box (on page 140), go to the Author tab, then the Actions subtab.
4. To create a new action, click the New button. To edit an existing action, select the action and click the Edit button.

   **Step Result:** In either case, this opens the Action configuration dialog box (on page 148) where you can configure numerous aspects of the action.

5. Once you are finished, click OK several times to exit the configuration dialog box.

   **Result:** Your changes are stored in the *.framework file for your particular framework.

Creating or Editing Actions Using an Individual XML File for Each Action

It is possible to work with Author mode actions outside the Document Type configuration dialog box (on page 140) and store them externally from the *.framework file. You can either export existing actions or use a template to create a new action from scratch. The benefits of using this approach are:

- You can share, copy, or reuse each individual action across multiple projects or frameworks.
- It is easier to develop and test action configurations. After configuring the XML file that defines an action, you can test its functionality by opening a document from your particular framework and invoking the action to see if it works as expected. If you did not get the desired result, you can simply repeat the process until you are happy with the result without having to navigate through the framework configuration dialog box.

**Exporting Actions**

To export existing Author mode actions into individual XML files, follow this procedure:

1. Open the Preferences dialog box (Options > Preferences) (on page 124), go to Document Types Association, and select the framework.
2. Select your framework and click the Edit button (or you can use the Duplicate or Extend button to create an extension of the framework (on page 2133)).
3. In the resulting **Document Type configuration dialog box** *(on page 140)*, go to the **Author** tab, then the **Actions** subtab.

4. **[Important]** Make sure the **Storage option** *(on page 141)* in the top part of the dialog box is set to **External** and the external location must be a subdirectory of your current framework directory *(see the Notes About the Storage Path section *(on page 2152)*).  

5. Select the action (or multiple actions) you want to export, right-click, and use the **Export** action *(this action is also located at the bottom of the table of actions).*  

**Step Result:** If you choose to export a single action, a resulting dialog box will allow you to select the destination path for the new XML file that contains the configuration details of the action. If you export multiple actions, they will automatically be saved as individual XML files inside a newly created folder *(it will have `_externalAuthorActions` at the end of the folder name)* inside your current framework directory.

6. **[Important]** Click **OK** several times to confirm your changes and exit the **Preferences** dialog box. The files will not be created until you exit this dialog box.  

**Step Result:** Each exported action is extracted from the framework configuration file and exported as an individual XML file.

7. To configure and test a particular action, you can open its corresponding XML file in Oxygen XML Editor, make changes, save the file, then open a document from your framework, test the action and repeat until you get the desired result.

**Note:** You can add or edit the action files outside of Oxygen XML Editor, but you will need to restart the application each time to reload the changes.

### Creating New Actions

To create a new **Author** mode action outside the framework configuration dialog box, follow these steps:

1. Open the **New document wizard** *(on page 368)*, search for a template called **Author Action**, and choose a storage path and file name. Remember that ultimately, it must be saved in a subdirectory of your particular framework directory *(see the Notes About the Storage Path section *(on page 2152)*).  

**Step Result:** The resulting XML file contains some hints and it is an example of an action configuration that will insert a new paragraph.

2. Configure the action as needed and save your changes.

**Note:** You can use XInclude to reuse different fragments *(such as XPath expressions or configured operations between actions).*
**Example: Reusing and XPath Expression**

```xml
<a:operations>
  <a:operation id="InsertFragmentOperation">
    <a:xpathCondition>
      <xi:include href="expression.txt" xmlns:xi="http://www.w3.org/2001/XInclude" parse="text"/>
    </a:xpathCondition>
    <a:arguments>
      <a:argument name="insertLocation">
        <xi:include href="expression.txt" xmlns:xi="http://www.w3.org/2001/XInclude" parse="text"/>
      </a:argument>
    </a:arguments>
  </a:operation>
</a:operations>
```

Where the content of the `expression.txt` file is `self::para`.

**Example: Reusing an Entire Operation**

```xml
<a:operations xmlns:a="http://www.oxygenxml.com/ns/author/external-action">
  <xi:include href="operation.xml" xpointer="element(/1/1)" xmlns:xi="http://www.w3.org/2001/XInclude"/>
</a:operations>
```

Where the content of `operation.xml` is:

```xml
<a:operations xmlns:a="http://www.oxygenxml.com/ns/author/external-action">
  <a:operation id="ToggleSurroundWithElementOperation">
    <a:xpathCondition>ancestor-or-self::p</a:xpathCondition>
    <a:arguments>
      <a:argument name="element"/><i/></a:argument>
    </a:arguments>
  </a:operation>
</a:operations>
```

**Step Result:** At this point, the action has been created but it needs to be added to the UI (in a toolbar or menu).

3. Add the new action to a UI component. For example, to add it in a toolbar, open the Document Type configuration dialog box (on page 140), go to the Author tab, then the Toolbar subtab (on page 160), and add the action.
4. To test the action, you can open a document from your framework and test the action. If you don't get the desired result, open the action file, make changes, then test them again. Repeat until you get the desired result.

**Note:**
You can add or edit the action files outside of Oxygen XML Editor, but you will need to restart the application each time to reload the changes.

**Notes About the Storage Path**

As mentioned above, it is imperative that the action configuration files be stored in a specific subdirectory of your particular framework directory.

There are two possible naming conventions for this subdirectory:

- `{framework_directory}/externalAuthorActions` - If there are multiple framework subdirectories inside `{framework_directory}`, using this path structure will make the actions available to all of them.
- `{framework_directory}/{framework_file_name}.externalAuthorActions` - Using this path structure will make the actions only available in the framework stored inside the `{framework_file_name}.framework` file.

**Note:**
When exporting actions from the UI *(on page 2149)*, this is the directory structure that is used.

**Action Configuration Tips**

- If an action is configured to insert a fragment that contains entities, you need to wrap it in `CDATA` markup.
- For a list of default operation, see Built-in Author Mode Operations *(on page 2152).*

**Resources**

For more information about customizing Author mode actions, see the following resources:

- Webinar: **Improving the Authoring Experience Through Custom Actions.**
- Webinar: **Working with DITA in Oxygen - Customizing the Editing Experience.**

**Related information**

Framework Configuration Dialog Box: Actions Subtab *(on page 147)*

**Built-in Author Mode Operations**

This topic lists the default operations for the Author mode.
ChangeAttributeOperation

This operation allows you to add/modify/remove an attribute. You can use this operation in your own custom Author mode action to modify the value for a certain attribute on a specific XML element. The arguments of the operation are:

- **name**
  - The attribute local name.

- **namespace**
  - The attribute namespace.

- **elementLocation**
  - The XPath location that identifies the element.

- **value**
  - The new value for the attribute. If empty or null the attribute will be removed.

- **editAttribute**
  - If an in-place editor exists for this attribute, it will automatically activate the in-place editor and start editing.

- **removeIfEmpty**
  - The possible values are `true` and `false`. True means that the attribute should be removed if an empty value is provided. The default behavior is to remove it.

ChangeAttributesOperation

This operation allows you to add/modify/remove multiple attributes. You can use this operation in your own custom Author mode action to modify the value for one or more attributes for one or more XML elements. The arguments of the operation are:

- **elementLocations**
  - The XPath location that identifies the elements whose attributes will be affected. If not defined, the element at the cursor location will be used.

- **attributeNames**
  - The names of the attributes to add, modify, or remove, separated by the new-line character (`\n`). The values can be local names or Clark notations.

- **values**
  - The new attributes values, each on a new line, separated by the new-line character (`\n`). An empty value will remove the attribute if `removeIfEmpty` is set to `true`.

- **removeIfEmpty**
  - The possible values are `true` (default) and `false`. True means that the attribute will be removed if an empty value is provided.
**ChangePseudoClassesOperation**

Operation that sets a list of pseudo-class values to nodes identified by an XPath expression. It can also remove a list of values from nodes identified by an XPath expression. The operation accepts the following parameters:

- **setLocations**
  An XPath expression indicating a list of nodes that will have the specified list of pseudo-classes set. If it is not defined, then the element at the cursor position will be used.

- **setPseudoClassNames**
  A space-separated list of pseudo-class names that will be set on the matched nodes.

- **removeLocations**
  An XPath expression indicating a list of nodes that will have the specified list of pseudo-classes removed. If it is not defined, then the element at the cursor position will be used.

- **removePseudoClassNames**
  A space-separated list of pseudo-class names that will be removed from the matched nodes.

- **includeAllNodes**
  The possible values are `yes` and `no`. If set to `yes`, comments, CDATA, and text nodes are included when evaluating XPath expressions. If set to `no`, they are ignored.

**DeleteElementOperation**

Deletes the node indicated by the `elementLocation` parameter XPath expression. If missing, the operation will delete the node at the cursor location.

**DeleteElementsOperation**

Deletes the nodes indicated by the `elementLocations` parameter XPath expression. If missing, the operation will delete the node at the cursor location.

**ExecuteCommandLineOperation**

This operation allows you to start a process executing a given command line. It has the following arguments:

- **name**
  The name of the operation (or name of the console panel that corresponds to the process run by an action built over this operation).

- **workingDirectory**
  The path to the directory where the command line is executed. The default value is "." (current directory).

- **cmdLine**
  The command line to be executed (accepts editor variables (on page 2168)).
showConsole

If set to true, the console panel will be displayed in Oxygen XML Editor. The default value is false.

wait

If set to true, the command line will wait for the operation to finish. The default value is false.

ExecuteMultipleActionsOperation

This operation allows the execution of a sequence of actions, defined as a list of action IDs. The actions must be defined by the corresponding framework, or one of the common actions for all frameworks supplied by Oxygen XML Editor.

actionIDs

The action IDs list that will be executed in sequence, the list must be a string sequence containing the IDs separated by commas or new lines.

ExecuteMultipleWebappCompatibleActionsOperation

An implementation of an operation that runs a sequence of Oxygen XML Web Author-compatible actions, defined as a list of IDs.

ExecuteTransformationScenariosOperation

This operation allows running one or more transformation scenarios defined in the current document type association (on page 3218), in the project options, or in the global options. A use case would be to add a toolbar button that triggers publishing to various output formats. The argument of the operation is:

scenarioNames

The list of scenario names that will be executed, separated by new lines.

ExecuteValidationScenariosOperation

This operation allows running one or more validation scenarios defined in the current document type association (on page 3218), in the project options, or in the global options. The single argument for the operation is:

scenarioNames

The list of scenario names that will be executed, separated by new lines.

InsertEquationOperation

Inserts a fragment containing a MathML equation at the cursor offset. The argument of this operation is:

fragment

The XML fragment containing the MathML content that should be inserted.
InsertFragmentOperation

Inserts an XML fragment at the current cursor position. The selection, if there is one, remains unchanged. The fragment will be inserted in the current context of the cursor position meaning that if the current XML document uses some namespace declarations then the inserted fragment must use the same declarations. The namespace declarations of the inserted fragment will be adapted to the existing namespace declarations of the XML document. For more details about its list of parameters, see Arguments of InsertFragmentOperation (on page 2174).

InsertOrReplaceFragmentOperation

Similar to InsertFragmentOperation (on page 2156), except it removes the selected content before inserting the fragment. Also, the insertPosition parameter has another possible value: Replace. If this value is used, the operation deletes the node selected by the XPath expression denoted by the insertLocation parameter. For more details about its list of parameters, see Arguments of InsertFragmentOperation (on page 2174).

InsertOrReplaceTextOperation

Inserts a text at current position removing the selected content, if any. The argument of this operation is:

   text

   The text section to insert.

InsertXIncludeOperation

Insert an XInclude element at the cursor offset. Opens a dialog box that allows you to browse and select content to be included in your document and automatically generates the corresponding XInclude instruction.

JSOperation

Allows you to call the Java API from custom JavaScript content. For some sample JSOperation implementations, see https://github.com/oxygenxml/javascript-sample-operations.

Notice:
For the Oxygen XML Web Author, this operation cannot be invoked using the JavaScript API.

This operation accepts the following parameter:

   script

   The JavaScript content to execute. It must have a function called doOperation(), which can use the predefined authorAccess variable. The authorAccess variable has access to the entire ro.sync.ecss.extensions.api.AuthorAccess Java API.

   The following example is a script that retrieves the current value of the type attribute on the current element, allows the end-user to edit its new value and sets the new value in the document:
function doOperation()
{
    // The current node is either entirely selected...
    currentNode = authorAccess.getEditorAccess().getFullySelectedNode();
    if(currentNode == null){
        // or the cursor is placed in it
        caretOffset = authorAccess.getEditorAccess().getCaretOffset();
        currentNode = authorAccess.getDocumentController().getNodeAtOffset(caretOffset);
    }
    // Get current value of the @type attribute
    currentTypeValue = "";
    currentTypeValueAttr = currentNode.getAttribute("type");
    if(currentTypeValueAttr != null){
        currentTypeValue = currentTypeValueAttr.getValue();
    }
    // Ask user for new value for attribute.
    newTypeValue = javax.swing.JOptionPane.showInputDialog("Input @type value", currentTypeValue);
    if(newTypeValue != null){
        // Create and set the new attribute value for the @type attribute.
        attrValue = new Packages.ro.sync.ecss.extensions.api.node.AttrValue(newTypeValue);
        authorAccess.getDocumentController().setAttribute("type", attrValue, currentNode);
    }
}

Tip:
You can call functions defined inside a script called commons.js from your custom script content so that you can use that external script file as a library of functions. Note that this commons.js file must be placed in the root of the framework directory (for example, \OXYGEN_INSTALL_DIR\frameworks\dita\commons.js) because that is the only location where Oxygen XML Editor will look for it.

MoveCaretOperation

Flexible operation for moving the cursor within a document and it is also capable of performing a selection. The operation accepts the following arguments:

**xpathLocation**

An XPath expression that identifies the node relative to where the cursor will be moved. If the expression identifies more than one node, only the first one will be taken into account.
position

The position relative to the node obtained from the XPath expression where the cursor will be moved. When also choosing to perform a selection, you can use the following possible values:

- **Before** - Places the cursor at the beginning of the selection.
- **Inside, at the beginning** - Places the cursor at the beginning of the selection.
- **After** - Places the cursor at the end of the selection.
- **Inside, at the end** - Places the cursor at the end of the selection.

selection

Specifies if the operation should select the element obtained from the XPath expression, its content, or nothing at all. The possible values of the argument are: None, Element, and Content.

MoveElementOperation

Flexible operation for moving an XML element to another location from the same document. XPath expressions are used to identify the source element and the target location. The operation takes the following parameters:

**sourceLocation**

XPath expression that identifies the content to be moved.

**deleteLocation**

XPath expression that identifies the node to be removed. This parameter is optional. If missing, the sourceLocation parameter will also identify the node to be deleted.

**surroundFragment**

A string representation of an XML fragment. The moved node will be wrapped in this string before moving it in the destination.

**targetLocation**

XPath expression that identifies the location where the node must be moved to.

**insertPosition**

Argument that indicates the insert position.

**moveOnlySourceContentNodes**

When set to `true`, only the content of the source element is moved.

**processTrackedChangesForXpathLocations**

When nodes are located via an XPath expression and the nodes are deleted with Change Tracking (on page 3224) enabled, they are considered as being present by default (thus, the change tracking is ignored). If you set this argument to `true` and change tracking is enabled, deleted nodes will be ignored when the XPath locations are computed (thus, the change tracking is NOT ignored).
alwaysPreserveTrackedChangesInMovedContent

When set to `true`, tracked changes are included when a copied fragment is inserted in a document, regardless of the current state of the `Track Changes` feature.

OpenInSystemAppOperation

Opens a resource in the system application that is associated with the resource in the operating system. The arguments of this operation are:

resourcePath

An XPath expression that, when executed, returns the path of the resource to be opened. Editor variables (on page 323) are expanded in the value of this parameter, before the expression is executed.

isUnparsedEntity

Possible values are `true` or `false`. If the value is `true`, the value of the `resourcePath` argument is treated as the name of an unparsed entity.

ReloadContentOperation

Reloads the content of the editor by re-reading the information from the URL used to open it. It accepts the following argument:

markAsNotModified

The possible values are `true` and `false`. After reloading the editor, the content may appear as modified and in some cases where the content is already present on the file server, you would not want the user to save it again. You can set this flag to `true` to prevent the editor from showing the content as modified.

RemovePseudoClassOperation

An operation that removes a pseudo-class from an element. Accepts the following parameters:

name

Name of the pseudo-class to be removed.

includeAllNodes

The possible values are `yes` and `no`. If set to `yes`, comments, CDATA, and text nodes are included when evaluating XPath expressions. If set to `no`, they are ignored.

elementLocation

The XPath location that identifies the element. If it is not defined, then the element at the cursor position is used. It can also identify multiple elements, in which case the pseudo class will be removed from all of them.

Example:
Suppose that there is a pseudo-class called `myClass` on the element `paragraph` and there are CSS styles matching the pseudo-class.

```
paragraph:myClass{
  font-size:2em;
  color:red;
}
paragraph{
  color:blue;
}
```

In the previous example, by removing the pseudo-class, the layout of the `paragraph` is rebuilt by matching the other rules (in this case, the foreground color of the `paragraph` element will become blue.

**RenameElementOperation**

This operation allows you to rename all occurrences of the elements identified by an XPath expression. The operation requires two parameters:

`elementName`

The new element name.

`elementLocation`

The XPath expression that identifies the element occurrences to be renamed. If this parameter is missing, the operation renames the element at current cursor position.

**ReplaceElementContentOperation**

An operation that replaces the content of the element at the cursor location (or fully selected element). The operation accepts the following parameters:

`fragment`

Specifies the fragment that will be inserted as the element content.

`elementLocation`

An XPath expression that identifies the element. If it is not defined, then the element at the cursor position is used.

**SetPseudoClassOperation**

An operation that sets a pseudo-class to an element. The operation accepts the following parameters:

`elementLocation`

An XPath expression that identifies the element that will have the pseudo-class set. If it is not defined, then the element at the cursor position is used.
name
The pseudo-class local name.

includeAllNodes
The possible values are yes and no. If set to yes, comments, CDATA, and text nodes are included when evaluating XPath expressions. If set to no, they are ignored.

ShowElementDocumentationOperation
Opens the associated specification HTML page for the current element. The operation accepts as parameter a URL pattern that points to the HTML page containing the documentation.

SurroundWithFragmentOperation
Surrounds the selected content with a text fragment. Since the fragment can have multiple nodes, the surrounded content will be always placed in the first leaf element. If there is no selection, the operation will simply insert the fragment at the cursor position. For more details about the list of parameters go to: Arguments of SurroundWithFragmentOperation (on page 2176).

SurroundWithTextOperation
This operation has two arguments (two text values) that will be inserted before and after the selected content. If there is no selected content, the two sections will be inserted at the cursor position. The arguments of the operation are:

header
The text that is placed before the selection.

footer
The text that is placed after the selection.

TogglePseudoClassOperation
An implementation of an operation to toggle on/off the pseudo-class of an element. Accepts the following parameters:

name
Name of the pseudo-class to be toggled on/off.

includeAllNodes
The possible values are yes and no. If set to yes, comments, CDATA, and text nodes are included when evaluating XPath expressions. If set to no, they are ignored.

elementLocation
The XPath location that identifies one or more elements that will have the pseudo class toggled. If it is not defined, then the element at the cursor position is used.
Example:

```xml
paragraph:myClass{
  color:red;
}
paragraph{
  color:blue;
}
```

By default, the paragraph content is rendered in blue. Suppose that you have a `TogglePseudoClassOperation` configured for the `myClass` pseudo-class. Invoking it the first time will set the `myClass` pseudo-class and the paragraph will be rendered in red. Invoking the operation again, will remove the pseudo-class and the visible result will be a blue rendered `paragraph` element.

**ToggleSurroundWithElementOperation**

This operation allows wrapping and unwrapping content in a specific wrapper element that can have certain attributes specified on it. It is useful to implement toggle actions such as highlighting text as bold, italic, or underline. The operation supports processing multiple selection intervals, such as multiple cells within a table column selection. The arguments of the operation are:

- `element`
  The element to wrap or unwrap content.

- `schemaAware`
  This argument applies only on the surround with element operation and controls whether or not the insertion is valid, based upon the schema. If the insertion is not valid, then wrapping action will be broken up into smaller intervals until the wrapping action is valid. For example, if you try to wrap a `paragraph` element with a `bold` element, it would not be valid, so the operation will wrap the text inside the paragraph instead, since it would be valid at that position.

**ToggleCommentOperation**

This operation allows for commenting or un-commenting the selected content. It does not have any arguments. If the selection is text, the operation wraps the selection in a comment. If the selection is a comment, the operation removes the comment.

**UnwrapTagsOperation**

This operation allows removing the element tags either from the current element or for an element identified with an XPath location. The argument of the operation is:

- `unwrapElementLocation`
  An XPath expression that identifies the element to unwrap. If it is not defined, the element at the cursor position is unwrapped.
XQueryUpdateOperation

Allows you to execute an XQuery Update script directly over content in Author mode.

Notice:
This operation is not applicable to the Oxygen XML Author Component or the Oxygen XML Web Author.

It supports the following arguments:

**script**

The XQuery Update script to be executed. The value can either be an XQuery script or a URL that points to the XQuery Update script. You can use the `${framework}` (on page 330) or `${frameworkDir}` (on page 330) editor variables to refer the scripts from the framework directory.

The script will be executed in the context of the node at the cursor position. If the script declares the following variable, it will also receive the selected nodes (assuming that entire nodes are selected):

```xml
declare variable $oxyxq:selection external;
```

In the example below (on page 2163), you can see how this argument is used.

**externalParams**

A string that can assign multiple key-value pairs separated by a comma or a new line character.

For example, if an XQuery script declares two external parameters like this:

```xml
declare variable $param1 external;
declare variable $param2 external;
```

You can pass custom values for each parameter by setting the `externalParams` to `param1=value1,param2=value2`.

**expandXincludeReferences**

Makes all XInclude elements transparent to the XQuery transformer. When the XInclude references are transparent, the XQueryUpdateOperation can use the referenced elements for further processing in the current document, but it cannot change their values in the original document. The default value is false, which means the XInclude elements are not transparent.

An example of an XQuery Update Script that converts paragraphs to list items:

```xml
declare namespace oxyxq = "http://www.oxygenxml.com/ns/xqu";
(: This variable will be linked to the selected nodes assuming that there are actually fully selected nodes. For example this selection will return null:
<p>{SEL_START}text{SEL_END} in para</p>
but this will give two "p" elements:
```
If a multiple selection exists it will also be processed and forwarded.
Again, only fully selected nodes will be passed.

declare variable $oxyxq:selection external;

(: We will process either the selection or the context node :) 

let $toProcess := if (empty($oxyxq:selection)) then
   (.)
else
   ($oxyxq:selection)

return if (not(empty($toProcess))) then
   {
   (: Create the list :) 
   let $ul :=
   <ul>
   {
      for $sel in $toProcess
         return
         <li>{$sel}</li>
   }
   </ul>

   return
   
   (: Delete the processed nodes :) 
   for $sel in $toProcess
      return
      delete node $sel,
   (: Inserts the constructed list :) 
   insert node $ul
      before $toProcess[1]
   }

else
   ()

XSLTOperation and XQueryOperation

Applies an XSLT or XQuery script on a source element and then replaces or inserts the result in a specified target element.
Notice:
For Oxygen XML Web Author, these operations cannot be invoked using the JavaScript API.

These operations accept the following parameters:

**sourceLocation**

An XPath expression indicating the element that the script will be applied on. If it is not defined, then the element at the cursor position will be used.

There may be situations where you want to look at an ancestor of the current element and make decisions in the script based on that. To do this, you can set the `sourceLocation` to point to an ancestor node then use the `oxy:current-element()` function to access the current element, as in the following example:

```xml
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="2.0"
    xmlns:oxygen="http://www.oxygenxml.com/ns/author/xpath-extension-functions"
    exclude-result-prefixes="oxy">
    <xsl:template match="/">
        <xsl:apply-templates select="oxy:current-element()"/>
    </xsl:template>

    <xsl:template match="para">
        <!-- And the context is again inside the current element, but we can use information from the entire XML -->
        <xsl:variable name="keyImage" select="//imagedata[@fileref='images/lake.jpeg']/ancestor::inlinemediaobject/@xml:id/string()"/>
        <xref linkend="{$keyImage}" role="key_include" xmlns="http://docbook.org/ns/docbook"/>
    </xsl:template>
</xsl:stylesheet>
```

**targetLocation**

An XPath expression indicating the insert location for the result of the transformation. If it is not defined then the insert location will be at the cursor location.

**script**

The script content (XSLT or XQuery). The base system ID for this will be the `framework` file, so any include/import reference will be resolved relative to the `.framework` file that contains this action definition.

For example, for the following script, the imported `xslt_operation.xsl` needs to be located in the current `framework` directory.
You can also use a path for an included or imported reference. When using a path, the following apply:

- A relative path is resolved to the framework directory.
- The \$\{framework\} editor variable (on page 330) can also be used to reference resources from the framework directory.
- The path is passed through the catalog mappings. It helps to use an absolute URL (for instance, http://www.oxygenxml.com/fr/testy.xsl) and map it in the catalog.xml file from the framework directory to a resource from the framework.

**action**

The insert action relative to the node determined by the target XPath expression. It can be: Replace, At cursor position, Before, After, Inside as first child or Inside as last child.

**caretPosition**

The position of the cursor after the action is executed. It can be: Preserve, Before, Start, First editable position, End, or After. If this parameter is not set, you can still indicate the position of the cursor by using the \$\{caret\} editor variable (on page 329) in the inserted content.

**expandEditorVariables**

Parameter controlling the expansion of editor variables (on page 323) returned by the script processing. Expansion is enabled by default.

**suspendTrackChanges**

It has 2 possible values (true and false). The default value is false. When set to true, the Track Changes (on page 3224) feature is deactivated. When using this argument, after the action is finished, the state of the Track Changes feature is restored to its initial value.

**externalParams**

A string that can assign multiple key-value pairs separated by a comma or a new line character.

For example, if an XQuery script declares two external parameters like this:

```xml
declare variable $param1 external;
declare variable $param2 external;
```

You can pass custom values for each parameter by setting the externalParams to `param1=value1,param2=value2`.

**XSLTOperation Example:** Sort a list with respect to the language declared on the root element:
Suppose you want an action that will sort a list with respect to the language declared on the root element and you have an XML file like this:

```xml
<article xml:lang="en">
  <ul>
    <li>B</li>
    <li>C</li>
    <li>A</li>
  </ul>
</article>
```

The **XSLTOperation** needs to be configured as follows:

- **sourceLocation** is set to `/*` so that the script has access to the root element and its children.
- **targetLocation** is left untouched (assuming that the action is active only when the cursor is inside the list).

The XSLT script would look like this:

```xml
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
                 xmlns:oxy="http://www.oxygenxml.com/ns/author/xpath-extension-functions"
                 xmlns:xs="http://www.w3.org/2001/XMLSchema"
                 exclude-result-prefixes="xs oxy" version="2.0">

  <xsl:template match="/*">
    <!-- sourceLocation parameter was set to /* to have a larger context. 
       We can pinpoint the element that contained the caret 
       using the oxy:current-element() function. 
    -->
    <xsl:apply-templates select="oxy:current-element()"/>
  </xsl:template>

  <xsl:template match="ul">
    <!-- Because the sourceLocation parameter was set to /* we now have access to 
         the root element and its attributes. -->
    <xsl:variable name="lang" select="/*/@xml:lang"/>
    <xsl:variable name="collationURI">
      <xsl:value-of select="concat('http://www.w3.org/2013/collation/UCA?lang=', $lang)"/>
    </xsl:variable>
    <xsl:copy>
      <xsl:copy-of select="@*"/>
    </xsl:copy>
  </xsl:template>

</xsl:stylesheet>
```
Using Entities and Xincludes with the XSLTOperation

- **Entities** will be resolved in the result.
- **Xincludes** are also resolved in the result, and you can alter the XML obtained after the resolving using the XSLT/XQuery script of the operation, but you cannot alter the included document itself.

Editor Variables in Author Mode Operations

**Author** mode operations can include parameters that contain the following editor variables (*on page 323*):

- **${caret}** - The position where the cursor is located. This variable can be used in a code template, in **Author** mode operations, or in a **selection** plugin.

**Note:**
The **${caret}** editor variable is available only for parameters that take XML content as values. It is replaced with the **${UNIQUE_CARET_MARKER_FOR_AUTHOR}** macro. The default Author operations process this macro and position the cursor at the designated offset.

**Note:**
The **${caret}** editor variable can be used for setting a fixed cursor position inside an XML fragment. To set the cursor position depending on the fragment inserted in the document, you can use **AuthorDocumentFilter** and inside the **insertFragment(AuthorDocumentFilterBypass, int, AuthorDocumentFragment)** method, use the **AuthorDocumentFragment.setSuggestedRelativeCaretOffset(int)** API on the given fragment.
• ${selection} - The currently selected text content in the currently edited document. This variable can be used in a code template, in Author mode operations, or in a selection plugin.

• ${ask('message', type, ('real_value1':'rendered_value1'; 'real_value2':'rendered_value2'; ...), 'default_value', @id)} - To prompt for values at runtime, use the ask('message', type, ('real_value1':'rendered_value1'; 'real_value2':'rendered_value2'; ...), 'default_value') editor variable.

You can set the following parameters:

- 'message' - The displayed message. Note the quotes that enclose the message.
- 'default-value' - Optional parameter. Provides a default value.
- @id - Optional parameter. Used for identifying the variable to reuse the answer using the ${answer(@id)} editor variable.
- type - Optional parameter (defaults to generic), with one of the following values:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Format: ${ask('message', type, 'default')}</th>
<th>Description: The input is considered to be generic text that requires no special handling.</th>
</tr>
</thead>
<tbody>
<tr>
<td>generic (default)</td>
<td>Format: ${ask('message', generic, 'default')}</td>
<td>Description: The input is considered to be generic text that requires no special handling.</td>
</tr>
</tbody>
</table>

**Example:**

- ${ask('Hello world!')} - The dialog box has a Hello world! message displayed.
- ${ask('Hello world!', generic, 'Hello again!')} - The dialog box has a Hello world! message displayed and the value displayed in the input box is 'Hello again!'.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Format: ${ask('message', url, 'default_value')}</th>
<th>Description: Input is considered a URL. Oxygen XML Editor checks that the provided URL is valid.</th>
</tr>
</thead>
<tbody>
<tr>
<td>url</td>
<td>Format: ${ask('message', url, 'default_value')}</td>
<td>Description: Input is considered a URL. Oxygen XML Editor checks that the provided URL is valid.</td>
</tr>
</tbody>
</table>

**Example:**

### Note:

The title of the dialog box will be determined by the type of parameter and as follows:

- For url and relative_url parameters, the title will be the name of the parameter and the value of the 'message'.
- For the other parameters listed below, the title will be the name of that respective parameter.
- If no parameter is used, the title will be "Input".
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Format: ${ask('message', parameter, 'default')}</th>
</tr>
</thead>
<tbody>
<tr>
<td>relative_url</td>
<td>Description: Input is considered a URL. Oxygen XML Editor tries to make the URL relative to that of the document you are editing.</td>
</tr>
<tr>
<td>password</td>
<td>Description: The input is hidden with bullet characters.</td>
</tr>
<tr>
<td>combobox</td>
<td>Description: Displays a dialog box that offers a drop-down menu. The drop-down menu is populated with the given rendered_value values. Choosing such a value will return its associated value (real_value).</td>
</tr>
</tbody>
</table>

**Note:**

If the $ask editor variable is expanded in content that is not yet saved (such as an untitled file, whose path cannot be determined), then Oxygen XML Editor will transform it into an absolute URL.

Example:

${ask('File location', relative_url, 'C:/example.txt')} - The dialog box has the name 'File location'. The URL inserted in the input box is made relative to the currently edited document location.

Example:

- ${ask('Input password', password)} - The displayed dialog box has the name 'Input password' and the input is hidden with bullet symbols.
- ${ask('Input password', password, 'abcd')} - The displayed dialog box has the name 'Input password' and the input hidden with bullet symbols. The input field already contains the default abcd value.

**Note:**

The list of 'real_value':'rendered_value' pairs can be computed using ${xpath_eval()}. 

- ${ask('Input URL', url)} - The displayed dialog box has the name Input URL. The expected input type is URL.
- ${ask('Input URL', url, 'http://www.example.com')} - The displayed dialog box has the name Input URL. The expected input type is URL. The input field displays the default value http://www.example.com.
Note:
The 'default' parameter specifies the default-selected value and can match either a key or a value.

Example:

▪ \$\{ask('Operating System', combobox, ('win': 'Microsoft Windows', 'macos': 'macOS', 'lnx': 'Linux/UNIX'), 'macos')\} - The dialog box has the name 'Operating System'. The drop-down menu displays the three given operating systems. The associated value will be returned based upon your selection.

Note:
In this example, the default value is indicated by the osx key. However, the same result could be obtained if the default value is indicated by macOS, as in the following example: \$\{ask('Operating System', combobox, ('win': 'Microsoft Windows', 'macos': 'macOS', 'lnx': 'Linux/UNIX'), 'macOS')\}

▪ \$\{ask('Mobile OS', combobox, ('ios': 'iOS', 'and': 'Android'), 'Android')\}

▪ \$\{ask('Mobile OS', combobox, ($\{xpath_eval($for$pair$in$(['ios', 'iOS'], ['and', 'Android'])\} return $pair?1 || $pair?2 || ';')$), 'ios')\}

editable_combobox

Format: \$\{ask('message', editable_combobox, ('real_value1': 'rendered_value1';...;'real_valueN': 'rendered_valueN'), 'default')\}

Description: Displays a dialog box that offers a drop-down menu with editable elements. The drop-down menu is populated with the given rendered_value values. Choosing such a value will return its associated real value (real_value) or the value inserted when you edit a list entry.

Note:
The list of real_value:rendered_value pairs can be computed using \$\{xpath_eval()\}.

Note:
The 'default' parameter specifies the default-selected value and can match either a key or a value.
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Example:</th>
</tr>
</thead>
<tbody>
<tr>
<td>▪ <code>${ask('Operating System', editable_combobox, ('win':'Microsoft Windows';'macos':'macOS';'lnx':'Linux/UNIX'), 'macos')}</code> - The dialog box has the name 'Operating System'. The drop-down menu displays the three given operating systems and also allows you to edit the entry. The associated value will be returned based upon your selection or the text you input.</td>
<td></td>
</tr>
</tbody>
</table>

| radio | Format: `${ask('message', radio, (‘real_value1’:‘rendered_value1’;…;‘real_valueN’:‘rendered_valueN’), ‘default’)}` |
| Description: Displays a dialog box that offers a series of radio buttons. Each radio button displays a 'rendered_value' and will return an associated real_value. |
| Note: The list of 'real_value':’rendered_value’ pairs can be computed using ${xpath_eval()}. |
| Note: The 'default' parameter specifies the default-selected value and can match either a key or a value. |
| Example: |
| ▪ `${ask('Operating System', radio, (‘win’:‘Microsoft Windows’;‘macos’:‘macOS’;‘lnx’:‘Linux/UNIX’), ‘macos’)}` - The dialog box has the name 'Operating System'. The radio button group allows you to choose between the three operating systems. |
| Note: In this example, macOS is the default-selected value and if selected, it would return macos for the output. |
• `{timeStamp}` - The timestamp, which is the current time in Unix format. For example, it can be used to save transformation results in multiple output files on each transformation.

• `{uuid}` - Universally unique identifier, a unique sequence of 32 hexadecimal digits generated by the Java UUID class.

• `{id}` - Application-level unique identifier. It is a short sequence of 10-12 letters and digits that is not guaranteed to be universally unique.

• `{cfnn}` - Current file name without the extension and parent folder. The current file is the one currently open and selected.

• `{cfne}` - Current file name with extension. The current file is the one currently open and selected.

• `{cf}` - Current file as file path, that is the absolute file path of the currently edited document.

• `{cfd}` - Current file folder as file path, that is the path of the currently edited document up to the name of the parent folder.

• `{frameworksDir}` - The path (as file path) of the frameworks directory. When used to define references inside a framework configuration, it expands to the parent folder of that specific framework folder. Otherwise, it expands to the main frameworks folder defined in the Document Type Association > Locations preferences page.

• `{pd}` - The file path to the folder that contains the current project file (.xpr).

• `{oxygenInstallDir}` - Oxygen XML Editor installation folder as file path.

• `{homeDir}` - The path (as file path) of the user home folder.

• `{pn}` - Current project name.

• `{env(VAR_NAME)}` - Value of the VAR_NAME environment variable. The environment variables are managed by the operating system. If you are looking for Java System Properties, use the `{system(var.name)}` editor variable.

• `{system(var.name)}` - Value of the var.name Java System Property. The Java system properties can be specified in the command-line arguments of the Java runtime as -Dvar.name=var.value. If you are looking for operating system environment variables, use the `{env(VAR_NAME)}` editor variable instead.

• `{date(pattern)}` - Current date. The allowed patterns are equivalent to the ones in the Java SimpleDateFormat class. Example: yyyy-MM-dd.

Note:
This editor variable supports both the xs:date and xs:dateTime parameters. For details about xs:date, go to: http://www.w3.org/TR/xmlschema-2/#date. For details about xs:dateTime, go to: http://www.w3.org/TR/xmlschema-2/#dateTime.

How to Find More Information About the Arguments of an Operation

If you need to find more information about the arguments of an operation, there are several places where this information is available:

• In the API documentation for the particular operation.

• By invoking the getArguments() method on the operation.

• In the source code of the operation.

• In Oxygen XML Editor:
1. Go to Options > Preferences > Document Type Association, select a document type and click the New, Edit, Duplicate, or Extend button (on page 138).

2. Go to the Author tab and then the Actions subtab. At the bottom of this subtab, click + New to open the Action dialog box (on page 148).

3. Locate the Operation field and click the Choose button on the right side. This will open a dialog box that displays the default operations.

4. Double-click the operation (or select it and click OK).

   The arguments for the operation will now be displayed in the Action dialog box (on page 148).

Arguments of InsertFragmentOperation

**fragment**

This argument has a textual value. This value is parsed by Oxygen XML Editor as it was already in the document at the cursor position. You can use entity references declared in the document and it is namespace aware. The fragment may have multiple roots.

You can even use namespace prefixes that are not declared in the inserted fragment, if they are declared in the document where the insertion is done. For the sake of clarity, you should always prefix and declare namespaces in the inserted fragment!

If the fragment contains namespace declarations that are identical to those found in the document, the namespace declaration attributes will be removed from elements contained by the inserted fragment.

There are two possible scenarios:

- **Prefixes that are not bound explicitly**

For instance, the fragment:

```xml
<x:item id="dty2"/>
&ent;
<x:item id="dty3"/>
```

Can be correctly inserted in the document: ('|' marks the insertion point):

```xml
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE x:root [ 
  <!ENTITY ent "entity">
]>

<x:root xmlns:x="nsp">
  |
</x:root>
```
Result:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE x:root [ 
  <!ENTITY ent "entity"> ]>
<x:root xmlns:x="nsp">
  <x:item id="dty2"/>
  &ent;
  <x:item id="dty3"/>
</x:root>
```

• **Default namespaces**

If there is a default namespace declared in the document and the document fragment (on page 3218) does not declare a namespace, the elements from the fragment are considered to be in no namespace.

For instance, the fragment:

```xml
  <item id="dty2"/>
  <item id="dty3"/>
```

Inserted in the document:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<root xmlns="nsp">
  <item xmlns="" id="dty2"/>
  <item xmlns="" id="dty3"/>
</root>
```

Gives the result document:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<root xmlns="nsp">
  <item xmlns="" id="dty2"/>
  <item xmlns="" id="dty3"/>
</root>
```

**insertLocation**

An XPath expression that is relative to the current node. It selects the reference node for the fragment insertion. When missing, the fragment will be inserted at the cursor position.

**insertPosition**

Specifies where the insertion is made relative to the reference node selected by the insertLocation. It can be one of the following constants:
**Inside as first child** (default value) - The fragment is inserted as first child of the reference node.
**Inside as last child** - The fragment is inserted as the last child of the reference node.
**After** - The fragment is inserted after the reference node.
**Before** - The fragment is inserted before the reference node.

**goToNextEditablePosition**

After inserting the fragment, the first editable position is detected and the cursor is placed at that location. It handles any in-place editors used to edit attributes. It will be ignored if the fragment specifies a cursor position using the `{$caret}` editor variable (on page 329). The possible values of this action are **true** and **false**.

**schemaAware**

This argument applies only on the surround with element operation and controls whether or not the insertion is valid, based upon the schema. If the insertion is not valid, then wrapping action will be broken up into smaller intervals until the wrapping action is valid. For example, if you try to wrap a **paragraph** element with a **bold** element, it would not be valid, so the operation will wrap the text inside the paragraph instead, since it would be valid at that position.

**insertEvenIfInvalid**

The possible values of this argument are **true** and **false**. If **true**, the content that would make the document invalid is accepted. If **false** and the insertion is not valid, the operation will not be executed and an error message will be displayed.

**Arguments of SurroundWithFragmentOperation**

**fragment**

The XML fragment that will surround the selection. For example, consider the fragment:

```
<F>
  <A></A>
  <B>
    <C></C>
  </B>
</F>
```

and the document:

```
<doc>
  <X></X>
  <Y></Y>
  <Z></Z>
</doc>
```
Considering the selected content to be surrounded is the sequence of elements \( x \) and \( y \), then the result is:

\[
<doc>
  <F>
    <A>
      <X></X>
      <Y></Y>
    </A>
    <B>
      <C></C>
    </B>
  </F>
  <Z></Z>
</doc>
\]

Since the element \( A \) was the first leaf in the fragment, it received the selected content. The fragment was then inserted in the place of the selection.

**Note:**

If the first leaf is not the desired location for the surrounded fragment, you can use \( \text{ro.sync.ecss.extensions.commons.operations.InsertOrReplaceFragmentOperation} \) and set the following arguments:

- **fragment**
  
  The XML fragment that will surround the selection. Use the \( \$\{selection\} \) editor variable in the location you want to place the surrounded fragment.

- **schemaAware**
  
  Set it to \( \text{false} \) to avoid moving the fragment if it is not valid at the given location.

**schemaAware**

This argument applies only on the surround with element operation and controls whether or not the insertion is valid, based upon the schema. If the insertion is not valid, then wrapping action will be broken up into smaller intervals until the wrapping action is valid. For example, if you try to wrap a \( \text{paragraph} \) element with a \( \text{bold} \) element, it would not be valid, so the operation will wrap the text inside the paragraph instead, since it would be valid at that position.

**Adding a Custom Operation to an Existing Framework**

This task explains how to add a custom **Author** mode operation to an existing **framework (on page 3219)** (document type).
1. Set up a sample project by following the instructions for installing the SDK.

**Tip:**
The SDK contains a sample framework project called `oxygen-sample-framework`.

2. A variety of classes in the `simple.documentation.framework.operations` package implement the `ro.sync.ecss.extensions.api.AuthorOperation` interface. Depending on your use-case, modify one of these classes.

3. Pack the operation class inside a Java JAR library.

4. Copy the JAR library to your framework directory (for example, `{OXYGEN_INSTALL_DIR}/frameworks/[FRAMEWORK_DIR]`).

5. Open the Preferences dialog box (Options > Preferences) (on page 124), go to Document Type Association, and edit the document type (you need write access to the `{OXYGEN_INSTALL_DIR}`) to open the Document Type configuration dialog box (on page 140).
   a. In the Classpath tab, add a reference to your JAR library (for example, `${framework}/customAction.jar`).
   b. Go to the Author tab, then go to the Actions subtab.
   c. Click the New button and use the Action dialog box (on page 148) to create a new action that uses your custom operation.
   d. Mount the action to the toolbars or menus. You can also define a shortcut key.

6. Share the modifications (on page 2288) with your colleagues. The files that should be shared are your `customAction.jar` library and the `.framework` configuration file from the `{OXYGEN_INSTALL_DIR}/frameworks/[FRAMEWORK_DIR]` directory.

**Related Information:**

*AuthorOperation API*

**Example: Configuring the Insert Section Action for a Framework**

This topic describes the procedure for defining the Insert Section action for a custom framework (on page 3219). It is assumed that the icon files, `$Section16.gif` for the menu item and `$Section20.gif` for the toolbar, are already available. Although you could use the same icon size for both the menu and toolbar, usually the icons from the toolbars are larger than the ones found in the menus. These files should be placed in your custom framework directory (`{OXYGEN_INSTALL_DIR}\frameworks\{CUSTOM_FRAMEWORK_DIR}\`).
1. Set the **ID** field to **insert_section**. This is a unique action identifier.
2. Set the **Name** field to **Insert Section**. This will be the action's name, displayed as a tooltip when the action is placed in the toolbar, or as the menu item name.
3. Set the **Menu access key** to **i**. On Windows, the menu items can be accessed using **Alt+letter** keys combination, when the menu is visible. The **letter** is visually represented by underlining the first letter from the menu item name having the same value.
4. Add a **Description**.
5. Set the **Large icon (20x20)** field to **${framework}/Section20.gif**. A good practice is to store the image files inside the **framework** directory and use **editor variable** (on page 323) **${framework}** to make the image relative to the **framework** location.
   If the images are bundled in a **JAR (on page 3220)** archive together with some Java operations implementation, for instance, it might be convenient for you to reference the images not by the file name, but by their relative path location in the class-path.
   If the image file **Section20.gif** is located in the **images** directory inside the **JAR** archive, you can reference it by using **/images/Section20.gif**. The **JAR** file must be added into the **Classpath** list.
6. Set the **Small icon (16x16)** field to **${framework}/Section16.gif**.
7. Click the text field next to **Shortcut key** and set it to **Ctrl+Shift+S (Meta+Shift+S on macOS)**. This will be the key combination to trigger the action using the keyboard only.
   The shortcut is enabled only by **adding the action to the main menu of Author mode (on page 2181)**, which contains all the actions that the author will have in a menu for the current document type.
8. At this time the action has no functionality added to it. Next you must define how this action operates. An action can have multiple operation modes. The first action mode enabled by the evaluation of its associated XPath expression will be executed when the action is triggered by the user. The XPath expression needs to be version 2.0 and its scope must be only element and attribute nodes of the edited document. Otherwise, the expression will not return a match and will not trigger the action. If the expression is left empty, the action will be enabled anywhere in the scope of the root element. For this example, suppose you want the action to add a section only if the current element is either a `<book>`, `<article>`, or another `<section>`.

a. Set the XPath expression field to:

   ```xml
   local-name()='section' or local-name()='book' or
   local-name()='article'
   ```

b. Set the `invoke operation` field to `InsertFragmentOperation` built-in operation, designed to insert an XML fragment at the cursor position. This belongs to a set of built-in operations, a complete list of which can be found in the `Author Default Operations (on page 2152)` section. This set can be expanded with your own Java operation implementations.

c. Configure the arguments section as follows:

   ```xml
   <section xmlns="http://www.oxygenxml.com/sample/documentation">
   <title/>
   </section>
   ```

   `insertLocation` - leave it empty. This means the location will be at the cursor position.
   `insertPosition` - select "Inside".

**Example: Configuring the Insert Table Action for a Framework**

This topic describes the procedure for defining the `Insert Table` action for a custom framework (on page 3219). Suppose that you want to create an action that inserts a table with three rows and three columns into a document and the first row is the table header. As with the `insert section` action (on page 2178), you will use the `InsertFragmentOperation` built-in operation.

Place the icon files for the menu item, and for the toolbar, in your custom `framework` directory (`{OXYGEN_INSTALL_DIR}\frameworks\[CUSTOM_FRAMEWORK_DIR]`).

1. Set `ID` field to `insert_table`.
2. Set `Name` field to `Insert table`.
3. Set `Menu access key` field to `t`.
4. Set `Description` field to `Adds a table element`.
5. Set `Toolbar icon` to `$(framework) / toolbarIcon.png`.
6. Set `Menu icon` to `$(framework) / menuIcon.png`.
7. Set `Shortcut key` to `Ctrl + Shift + T` (Command + Shift + T on macOS).
8. Set up the action's functionality:

   a. Set **XPath expression** field to `true()`.  
      `true()` is equivalent with leaving this field empty.

   b. Set **Invoke operation** to use **InsertFragmentOperation** built-in operation that inserts an XML fragment to the cursor position.

   c. Configure operation's arguments as follows:

      **fragment** - set it to:

      ```xml
      <table xmlns="http://www.oxygenxml.com/sample/documentation">
      <header>
        <td/> <td/> <td/>
      </header>
      <tr>
        <td/> <td/> <td/>
      </tr>
      <tr>
        <td/> <td/> <td/>
      </tr>
      </table>
      ```

      **insertLocation** - to add tables at the end of the section use the following code:

      ```xml
      ancestor::section/*[last()]
      ```

      **insertPosition** - Select After.

**Using Retina/HiDPI Icons for the Actions from a Framework**

Higher resolution icons can also be included in customized frameworks (on page 3219) for rendering them in a Retina or HiDPI display. The icons can be referenced directly from the **Document Type Configuration** dialog box (on page 140) (from the **Action** dialog box (on page 148)) or from an API (ro.sync.exml.workspace.api.node.customizer.XMLNodeRendererCustomizer (on page 2269)).

As with any image, the higher resolution icons are stored in the same images folder as the normal resolution images and they are identified by a scaling factor that is included in the name of the image files. For instance, icons with a Retina scaling factor of 2 will include `@2x` in the name (for example, `myIcon@2x.png`).

Developers should not specify the path of the alternate icons (`@2x` or `@3x`) in the **Action** dialog box (on page 148) or the **XMLNodeRendererCustomizer** API (on page 2269). When using a Retina or HiDPI display, Oxygen XML Editor automatically searches the folder of the normal icon for a corresponding image file with a Retina scaling factor in the name. If the higher resolution icon file does not exist, the normal icon is scaled and used instead.

**Related Information:**

Retina/HiDPI Images in Author Mode (on page 724)

**Customizing the Menu for a Framework**

Defined actions can be grouped into customized menus in the Oxygen XML Editor menu bar.
1. Open the **Document Type** configuration dialog box (on page 140), select your custom framework (on page 3219), and go to the Author tab.

2. Go to the **Menu** subtab. In the left side you have the list of actions and some special entries:
   - **Submenu** - Creates a submenu. You can nest an unlimited number of menus.
   - **Separator** - Creates a separator into a menu. This way you can logically separate the menu entries.

3. The right side of the panel displays the current actions for that menu tree. To change its name, click this label to select it, then click the **Edit** button.

4. Select the **Submenu** label in the left panel section and the appropriate label in the right panel section, then click the **Add as child** button. Change the submenu name to **Table**, using the **Edit** button.

5. Select the **Insert section** action in the left panel section and the **Table** label in the right panel section, then click the **Add as sibling** button.

6. Now select the **Insert table** action in the left panel section and the **Table** in the right panel section. Click the **Add as child** button.

**Figure 571. Configuring the Menu**

When opening a test document for a custom framework in **Author** mode, the menu you created is displayed in between the **Tools** and the **Document** menus. The upper part of the menu contains generic **Author** mode actions (common to all document types) and the two actions created previously (with **Insert table** under the **Table** submenu).
Customizing the Contextual Menu for a Framework

The contextual menu is displayed when you right-click in the Author editing area. You can only configure the bottom part of the menu, since the top part is reserved for a list of generic actions (such as Copy, Paste, Undo, etc.)

1. Open the Document Type configuration dialog box (on page 140) for the particular framework (on page 3219) and go to the Author tab. Next, go to the Contextual Menu subtab.
2. Follow the same steps as explained in the Configuring the Main Menu (on page 2181), except changing the menu name because the contextual menu does not have a name.

Note:
You can choose to reuse a submenu that contains general authoring actions. In this case, all actions (both general and framework-specific ones) are grouped together under the same submenu.
To test it, open the test file, and open the contextual menu. In the lower part there is shown the Table sub-menu and the Insert section action.

**Customizing the Content Completion Assistant for Author Mode Only**

You can customize the content of the following **Author** controls, adding items (which, when invoked, perform custom actions) or filtering the default contributed ones:

- **Content Completion Assistant (on page 3217)** window
- **Elements view (on page 633)**
- **Insert Element** menus (from the **Outline view (on page 539)** or breadcrumb **(on page 602)** contextual menus)

You can use the content completion customization support in a custom **framework (on page 3219)** by following this procedure:

1. Open the **Document type configuration dialog box (on page 140)** for your custom **framework** and select the **Author** tab. Next, go to the **Content Completion** tab (**on page 160)**.

![Figure 574. Customize Content Completion](image)

The top side of the **Content Completion** section contains the list with all the actions defined within the custom **framework** and the list of actions that you decided to include in the **Content Completion Assistant** list of proposals. The bottom side contains the list with all the items that you decided to remove from the **Content Completion Assistant** list of proposals.

2. If you want to add a custom action to the list of current **Content Completion** proposals, select the action item from the **Available actions** list and click the **Add as child** or **Add as sibling** button to include it in the **Current actions** list. A **Content Completion Item** dialog box appears, giving you the possibility to select where to provide the selected action.
3. If you want to exclude a certain item from the Content Completion proposals, you can use the Add button from the Filter - Remove content completion items list. The Remove item dialog box is displayed, allowing you to input the item name and to choose the controls that filter it. The Item name combo box accepts wildcards.

![Figure 576. Remove Item Dialog Box](image)

**Note:**
In the Item name drop-down menu, `<SPLIT>` refers to the action of splitting the element and creating a new one, while `<ENTER>` refers to the action of inserting a new line.

Related Information:
Customizing the Content Completion Assistant Using a Configuration File (on page 2191)
Customizing the Toolbars for a Framework

This procedure describes how to add defined actions to a toolbar for a custom framework. You can also create additional custom toolbars with existing or custom actions.

1. Open the Document Type configuration dialog box for your custom framework and select the Author tab.

2. Go to the Toolbar subtab.

   Figure 577. Configuring the Toolbar

   ![Toolbar Configuration Panel]

   The panel is divided in two sections. The left side contains a list of actions, while the right side contains an action tree, displaying the list of actions added in the toolbar. The special entry called Separator allows you to visually separate the actions in the toolbar.

3. To add an action, select it in the left panel and select the particular toolbar label where you want it added in the right panel section, then click the Add as child or Add as sibling button.

   **Result:** When opening a document for the particular framework in Author mode, the toolbar with the new buttons will be displayed in the toolbar area.

   **Tip:**

   If you have many custom toolbar actions, or want to group actions according to their category, add more toolbars with custom names and split the actions to better suit your purpose. If your toolbar is not displayed when switching to the Author mode, right-click the main toolbar, select Configure Toolbars, and make sure the appropriate toolbar (such as the Author Custom Actions toolbar) is selected.
Note:
A maximum of 16 toolbars can be added. If you add more, all extra toolbars will be automatically converted to sub-toolbars for the last added toolbar.

Customizing Text-to-Markup Shortcut Patterns

Some built-in frameworks include a configuration file that defines shortcut patterns that can be used in Author mode to automatically insert a certain XML structure. More specifically, the XML structure (fragment) automatically replaces a specific prefix pattern. For example, if you are editing a DITA document using the built-in DITA framework, entering a hyphen (-) followed by a space at the beginning of a paragraph would automatically replace them with an unordered list element \(<ul>\) with a child list item element \(<li>\). This is made possible by the AutoCorrect mechanism in Oxygen XML Editor.

It is possible to customize the particular configuration file (structureAutocorrect.xml) to define your own markup insertion shortcut patterns by following these steps:

1. Create a new resources folder (if it does not already exist) in the frameworks directory for the particular document type (for example, OXYGEN_INSTALL_DIR/frameworks/dita/resources).
2. Open the Preferences dialog box (Options > Preferences) (on page 124) and go to Document Type Association. Select the particular document type, click the Edit button, and in the Classpath tab (on page 144) add a link to that resources folder (if it does not already exist).
3. Create a new structureAutocorrect.xml file or edit an existing one (this file already exists in the resources folder of particular document types and you can use an existing file as a starting point for your customization).
4. Make the appropriate changes to your structureAutocorrect.xml file. The file should look like this:

```xml
<structure-autocorrect>
  <!-- Unordered lists -->
  <prefix-replacement prefix="-">  
    <ul><li/></ul>
  </prefix-replacement>

  <!-- Ordered lists -->
  <prefix-replacement prefix="1.">
    <ol><li/></ol>
  </prefix-replacement>

  <!-- Code block -->
  <prefix-replacement prefix="```">  
    <codeblock/>
  </prefix-replacement>

</structure-autocorrect>
```
Using this example, when a user enters one of the defined prefixes at the start of an element that allows the corresponding fragment, Oxygen XML Editor will automatically replace the prefix with its corresponding fragment. For example, entering a hyphen (-) at the beginning of a paragraph followed by a space would automatically replace them with an unordered list element (<ul>) with a child list item element (<li>). Any subsequently added content would be placed inside the first node/element that does not have a child node/element (in this example, the cursor would be placed in the first <li> element).

5. Save the file in the resources folder for the particular document type, using the fixed name: structureAutocorrect.xml (for example, OXYGEN_INSTALL_DIR/frameworks/dita/resources/structureAutocorrect.xml).

6. Restart the application and open a document for your particular framework to test your customization.

Note:
Once the file is created, changes that you make to it are processed by Oxygen XML Editor when you press the Reload toolbar button.

Customizing Smart Paste Support

The Smart Paste feature (on page 613) preserves certain style and structure information when copying content from some of the most common applications and pasting into frameworks (document types) that support Smart Paste (on page 613) in Oxygen XML Editor. For other document types, the default behavior of the paste operation is to keep only the text content without the styling.

The style of the pasted content can be customized by editing an XSLT stylesheet for a particular document type (framework (on page 3219)). The XSLT stylesheet must accept an XHTML flavor of the copied content as input, and transform it to the equivalent XML markup that is appropriate for the target document type of the paste operation.

How to Customize the Smart Paste Mapping

To customize the mapping between the markup of the copied content and the markup of the pasted content for a particular document type, follow these steps:

1. Make sure the particular framework contains a folder named resources in the following path structure:

```
/frameworks/[Document Type]/resources
```

2. Create an XSLT file named xhtml2content.xsl and save it in the resources folder for the particular framework.

   For example: /frameworks/[Document Type]/resources/xhtml2content.xsl

3. Add your customized styling in the XSLT file. A list of supported parameters can be found in the Supported Parameters for the Custom Smart Paste XSLT (on page 2190) section below.
Tip:
The built-in DITA framework includes an `xhtml2ditaDriver.xsl` file (in 
`[OXYGEN_INSTALL_DIR]/frameworks/dita/resources`) that imports various other 
stylesheets that apply cleanup and handle the conversion from the pasted HTML content to 
DITA. If you are using a custom extension of the DITA framework, you can copy the entire 
contents of the built-in `dita/resources` folder and customize the stylesheets according to 
your needs.

4. You can test modifications done in the stylesheet by pasting content without having to restart Oxygen 
XML Editor.

Result: When you paste content from external applications (such as a web browser or and Office document) 
to a document that is open in Author mode, and that matches the particular framework, the styling from the 
`xhtml2content.xsl` stylesheet will be applied on the clipboard contents.

Customized Smart Paste Stylesheet Sample:

```xml
<xsl:stylesheet version="1.0"
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    xmlns:xhtml="http://www.w3.org/1999/xhtml"
    xmlns="urn:hl7-org:v3"
    exclude-result-prefixes="xsl xhtml">

    <xsl:output method="xml" indent="no" omit-xml-declaration="yes"/>

    <xsl:template match="xhtml:b | xhtml:strong">
        <content styleCode="bold">
            <xsl:apply-templates select="@* | node()"/>
        </content>
    </xsl:template>

    <xsl:template match="*">
        <xsl:apply-templates select="@* | node()"/>
    </xsl:template>

    <xsl:template match="@* | node()">
        <xsl:copy>
            <xsl:apply-templates select="@* | node()"/>
        </xsl:copy>
    </xsl:template>

</xsl:stylesheet>
```
Supported Parameters for the Custom Smart Paste XSLT

The following parameters can be used in your XSLT stylesheet for customizing the Smart Paste mechanism:

- **inTableContext**
  - The custom XSLT stylesheet receives this parameter with a value of true if the end-user is pasting content inside a table.

- **folderOfPasteTargetXml**
  - A URL pointing to the folder where the currently edited XML document is located. This is used to save images relative to the current XML document.

- **context.path.names**
  - A sequence of element names showing the current context in the XML document where the paste occurred.

- **context.path.uris**
  - A sequence of namespaces, one for each context path name.

- **context.path.separator**
  - The separator between the path names. Its value can be used to split the context path names to a sequence.

By default, there is an extra check in place to ensure that the applied XSLT does not remove the original text from the pasted content. If there is a file called externalPasteOptions.xml in the resources folder, you can use it to specify the default behavior for checking if the XSLT stylesheet loses content during conversion:

```xml
<!-- Options that control external paste
    (automatic conversions when pasting HTML and URL flavors from the clipboard). -->
<pasteOptions>
  <!-- True to check if the entire sequence of words which get pasted are
       converted to the target vocabulary. If the check fails, the content
       will be inserted as a simple sequence of words without any formatting. -->
  <checkEntireContentIsFullyPreserved>true</checkEntireContentIsFullyPreserved>
</pasteOptions>
```

Related Information:
- Smart Paste in Author Mode (on page 613)
- Oxygen XML Blog: How Special Paste Works in Oxygen (DITA)
- Handling When URLs or XHTML Fragments are Dropped or Pasted in Author Mode (on page 2285)

Customize the Content Completion Assistant

Oxygen XML Editor gathers information from the associated schemas (DTDs, XML Schema, RelaxNG) to determine the proposals that appear in the Content Completion Assistant (on page 3217). Oxygen XML Editor
also includes support that allows you to customize the Content Completion Assistant to suit your specific needs.

There are two ways to customize the Content Completion Assistant in Oxygen XML Editor:

- You can add, modify, or remove actions that are proposed for each particular document type (framework (on page 3219)) by using the Content Completion subtab in the Document Type Association configuration dialog box (on page 160). To access this subtab, open the Preferences dialog box (Options > Preferences) (on page 124), go to Document Type Association, use the New, Edit, Duplicate, or Extend button (on page 138), click on the Author tab, and then the Content Completion subtab.

  **Note:**
  This works only for Author visual mode.

- You can use a cc_config.xml configuration file that is specific to each document type (framework) to configure the values that are proposed in certain contexts, to customize the attributes or elements that are proposed, or to customize how certain aspects of the proposals are rendered in the interface. The rest of the topics in this section explain how you can use this configuration file to customize the content completion.

**Resources**

To see more ideas for various advanced customization possibilities (including how to insert or reject proposals for the content completion assistant), watch our Webinar: Working with DITA in Oxygen - Customizing the Editing Experience.

**Customizing the Content Completion Assistant Using a Configuration File**

Oxygen XML Editor gathers information from the associated schemas (DTDs, XML Schema, RelaxNG) to determine the proposals that appear in the Content Completion Assistant (on page 3217). Oxygen XML Editor also includes support that allows you to customize the Content Completion Assistant to suit your specific needs.

There are two ways to customize the Content Completion Assistant in Oxygen XML Editor:

- You can add, modify, or remove actions that are proposed for each particular document type (framework (on page 3219)) by using the Content Completion subtab in the Document Type Association configuration dialog box (on page 160). To access this subtab, open the Preferences dialog box (Options > Preferences) (on page 124), go to Document Type Association, use the
New, Edit, Duplicate, or Extend button (on page 138), click on the Author tab, and then the Content Completion subtab.

- You can use a cc_config.xml configuration file that is specific to each document type (framework) to configure the values that are proposed in certain contexts, to customize the attributes or elements that are proposed, or to customize how certain aspects of the proposals are rendered in the interface. The rest of the topics in this section explain how you can use this configuration file to customize the content completion.

Related Information:
- Customizing the Content Completion Assistant for Author Mode Only (on page 2184)

## Configuring the Proposals for Elements and Attributes

There are many cases where elements have a relaxed content model and can accept a large number of child elements. For example, the DITA list item element (<li>) accepts more than 60 child elements. Oxygen XML Editor includes support to allow the content architect to put some constraints on the possible elements or attributes, or to impose some best practices in the way content is edited.

For an example of a specific use-case, suppose that you want to restrict DITA list item elements (<li>) to only accept paragraph elements (<p>). In this case, the Content Completion Assistant (on page 3217) should not offer any element other than a paragraph (<p>) when a list item (<li>) is inserted into a document. It would also be helpful if the required child element (<p>) was automatically inserted whenever a list item (<li>) is inserted.

One method of changing the content model is to alter the element definition in the associated schema (XML Schema, DTD, RelaxNG), but this may be complicated in some cases. Fortunately, Oxygen XML Editor offers a simple, alternative method of using a configuration file to customize the content completion proposals for each element.

### Setting up the Content Completion Configuration File

To customize the configuration file for the Content Completion Assistant (on page 3217), follow these steps:

1. Create a new resources folder (if it does not already exist) in the frameworks directory for the particular document type (for example, OXYGEN_INSTALL_DIR/frameworks/dita/resources).
2. Open the Preferences dialog box (Options > Preferences) (on page 124) and go to Document Type Association. Select the particular document type, click the Edit button, and in the Classpath tab (on page 144) add a link to that resources folder (if it does not already exist).
3. Create a new configuration file or edit an existing one.
   a. To easily create a new configuration file, you can use the Content Completion Configuration document template that is included in Oxygen XML Editor (File > New > Framework templates > Oxygen Extensions > Content Completion Configuration). The document template includes details about how each element and attribute is used in the configuration file.
   b. If a configuration file (cc_config.xml) already exists for the particular document type (in the resources folder), you can modify this existing file.
c. If you extend a framework, you need to copy the content of the `cc_config.xml` file from the base framework and modify it (e.g. create a `resources` folder in your framework extension folder and place the file there). You also need to make sure that the folder that contains the `cc_config.xml` file in your extension (e.g. `resources`) is listed in the Classpath tab (on page 144) before the one from the base framework.

If you only want to make small changes or add extra rules in your custom content completion configuration file, you need to name it `cc_config_ext.xml` and all the rules inside it are merged with the base `cc_config.xml` file. The merging is done by taking all the rules specified in the `cc_config_ext.xml` file into consideration after processing the set of rules from the base `cc_config.xml` file.

4. Make the appropriate changes to your custom configuration file.
5. Save the file in the `resources` folder for the particular document type, using the fixed name: `cc_config.xml` (for example, `OXYGEN_INSTALL_DIR/frameworks/dita/resources/cc_config.xml`).
6. Restart the application and open an XML document. In the Content Completion Assistant you should see your customizations.

Tip:
In some cases, you can simply use the `Refresh (F5)` action to test your customizations, without having to restart the application.

Attention:
In the Classpath tab (on page 144), if you have references to multiple `resources` folders, each with its own `cc_config.xml` file, the first reference listed in the Classpath tab takes precedence and the multiple configuration files are not combined.

Configuring Elements or Attributes that are Proposed for Each Element

For the purposes of customizing the elements or attributes that are proposed for each individual element, the configuration file (`cc_config.xml`) uses `<elementProposals>` elements. This element allows you to customize or filter the child elements and attributes for an element.

Warning:
Note that you can only choose elements or attributes that are already allowed by the schema in a particular context. For example, you cannot specify an element that is not allowed by the schema as a child of a particular node.

Elements:

To control the elements that are proposed for an element, you can use the following attributes for the `<elementProposals>` element:
• **path** - A path within the document that matches the element that will have its content completion proposals changed. For example, "title" matches all the `<title>` elements in the document, while "chapter/title" matches only the `<title>` elements that are direct children of the `<chapter>` element. You can use simplified forms of XPath in this attribute.

The XPath expressions can accept multiple attribute conditions and inside each condition you can use **AND/OR** boolean operators and parentheses to override the priority.

You can use one or more of the following attribute conditions (default attribute values are not taken into account):

- `element[@attr]` - Matches all instances of the specified element that include the specified attribute.
- `element[not(@attr)]` - Matches all instances of the specified element that do not include the specified attribute.
- `element[@attr = "value"]` - Matches all instances of the specified element that include the specified attribute with the given value.
- `element[@attr != "value"]` - Matches all instances of the specified element that include the specified attribute and its value is different than the one given.

**Example:** The following are examples of how you could use multiple boolean operators and parentheses inside an attribute condition:

```
*[a and b or c and d]
*[a and (b or c) and d]
```

The following are just examples of how simplified XPath expressions might look like:

- `elementName`
- `//elementName`
- `/elementName1/elementName2/elementName3`
- `//xs:localName`
- `//xs:documentation[@lang="en"]`

**Note:**
Using a namespace prefix requires that you declare it on the `<elementProposals>` element or on an ancestor element. For example:

```
<elementProposals xmlns:db5="http://docbook.org/ns/docbook"
                  path="db5:listitem"
                  insertElements="db5:para"/>
```

**Other Important Notes:**

- If the `path` attribute is missing, the customization will apply to the proposals for all elements. You can intentionally omit this attribute and use `possibleElements (on page 2195)` or `rejectElements (on page 2196)` to specify or restrict particular elements for a framework (on page 3219).
For example, suppose that in your DITA documents, you want to restrict your users from using `<image>` and `<fig>` elements because you do not want images to be included in your output. The configuration file should look like this:

```xml
<elementProposals rejectElements="image fig" />
```

Since the `@path` attribute is missing, the specified element will be filtered out from the proposals for the entire framework.

- If the particular document type has name namespaces, the `@path` should contain the qualified name. For example, in TEI documents, if you want to set a list of possible attributes for the `<span>` element, you need to use a qualified name like this (notice the declaration of the namespace prefix "t" and its usage):

```xml
<config xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.oxygenxml.com/ns/ccfilter/config
http://www.oxygenxml.com/ns/ccfilter/config/ccConfigSchemaFilter.xsd"
    xmlns="http://www.oxygenxml.com/ns/ccfilter/config"
    xmlns:t="http://www.tei-c.org/ns/1.0">

    <elementProposals path="t:span" possibleAttributes="type"/>

</config>
```

- **insertElements** - A space-separated sequence of child element names. Each time the element specified in the `@path` attribute is inserted into the document, these child elements will also be inserted in the order that they are listed. For example, `insertElements="b i"` will insert exactly one `<b>` element, followed by an `<i>` element. An empty value ("") means that no child elements should be inserted.

  **Note:**
  If this attribute is missing, the default required child elements will be inserted, as specified in the associated schema for the document.

- **possibleElements** - A space-separated list of element names that will be shown in the content completion list when invoked inside an element that is specified in the `@path` attribute. For example, "b i codeph ph" means that the Content Completion Assistant will contain these four elements when invoked on the element specified in the `@path` attribute. The following other possible values are also supported:

  - **NONE** - There will be no proposals in the content completion list.
  - **ALL** - All the possible elements specified in the associated schema will be presented in the content completion list. This is also the default behavior if this attribute is missing.
  - **INSERTED** - The proposals will be the same list of elements that are defined in the `@insertElements` attribute.

When using this attribute to specify multiple elements, only use one entry with the element names separated by a space:
• **rejectElements** - A space-separated list of element names that will be filtered out from the list of proposals that are presented in the content completion list. Each time the element specified in the `@path` attribute is inserted into the document, the list of proposals in the Content Completion Assistant will include the entries that are defined in the associated schema, minus the elements specified in this attribute.

**Note:**

This setting makes the application behave as if the rejected elements were not allowed by the schema in that location. Most of the toolbar actions take the schema into account when inserting content. If the inserted content is not allowed by the schema in that particular location, the application tries to find another location within close proximity where the content is allowed.

For example, suppose you reject the insertions of images in paragraphs. If a user has the cursor inside a paragraph and uses the toolbar action that inserts an image, the image will be inserted after the current paragraph rather than at the current location.

If you want to avoid having users insert an element directly from the content completion mechanism and want them to use a toolbar action instead, it is better to use the Document Type Configuration (on page 160) dialog box to remove the element.

When using this attribute to specify multiple elements, only use one entry with the element names separated by a space:

```xml
<elementProposals rejectElements="image fig imagemap foreign" />
```

**Attributes:**

To control the attributes that are proposed for an element, you can use the following attributes for the `<elementProposals>` element:

• **path** - A path within the document that matches the element that will have its attribute proposals changed. For example, "title" matches all the `<title>` elements in the document, while "chapter/title" matches only the `<title>` elements that are direct children of the `<chapter>` element. You can use simplified forms of XPath in this attribute. For examples of such forms of XPath expressions, see the note in XML Preferences (on page 207).

**Note:**

If this attribute is missing, the customization will apply to the proposals for all elements. You can intentionally omit this attribute and use possibleAttributes (on page 2197) or rejectAttributes (on page 2197) to specify or restrict attributes for an entire framework.
For example, suppose that you only want to allow a limited set of attributes in a customized framework. The configuration file should look like this:

```xml
<elementProposals possibleAttributes=""
    id domains href scope format type conref
    props keyref class"/>
```

Since the `@path` attribute is missing, this applies to the entire framework and only the specified attributes will be proposed.

- **insertAttributes** - A space-separated sequence of attribute names that will be inserted along with the element.

  ```xml
  <elementProposals path="ol/li" insertAttributes="product platform"/>
  ```

- **insertAttribute** - This is similar to the preceding attribute, but this one also allows you to specify a value for the attribute that will be inserted. This attribute should be used like this:

  ```xml
  <elementProposals path="ol/li">
    <insertAttribute name="platform" value="test"/>
  </elementProposals>
  ```

- **possibleAttributes** - A space-separated list of attribute names that will be shown in the content completion list when invoked inside an element that is specified in the `@path` attribute.

  When using this attribute to specify multiple attributes, only use one entry with the attribute names separated by a space:

  ```xml
  <elementProposals possibleAttributes="scope format type"/>
  ```

- **rejectAttributes** - A space-separated list of attribute names that will be filtered out from the list of proposals that are presented in the content completion list. Each time the element specified in the `@path` attribute is inserted into the document, the list of proposals in the Content Completion Assistant will include the entries that are defined in the associated schema, minus the attributes specified in this attribute.

  When using this attribute to specify multiple attributes, only use one entry with the attribute names separated by a space:

  ```xml
  <elementProposals rejectAttributes="importance platform product"/>
  ```
Other Important Notes About the Configuration File

---

Important:

- By default, the element names that do not have a namespace prefix are considered from no-namespace. Consider declaring the namespace mapping on the root of the configuration file and prefixing the element names from the @elementPath and @model attributes.

- This configuration file only affects the content completion assistance. It has no effect on validation or operations invoked from other areas in the interface (such as the toolbar or menus).

- To test the effects of your changes, you should restart the application, although in some cases, you can simply use the Reload (F5) action to test your customizations.

- When an XML element from the document is matched against a list of configured elementProposals, the first one in sequence takes precedence. Therefore, make sure you place the more specific elementProposals (those with a longer path) first in your configuration file.

- Simple wildcard patterns can be used in the following attributes: @possibleElements, @rejectElements, @possibleAttributes, and @rejectAttributes. For example, code*, *block, con*ref, _*. 

- Editor variables (on page 323) can be used in the @value attribute of the <insertAttribute> element. For example:

  ```xml
  <elementProposals path="prolog/critdates/created">
    <insertAttribute name="date" value="${date(yyyy-MM-dd)}"/>
  </elementProposals>
  ```

- Only simple recursion cases are detected and avoided by the editor, and logged to the console. Therefore, if complex elementProposals patterns are defined, you should avoid infinite recursions.

---

Examples: Configuring the Element Proposals

- **Example 1: Automatically Insert Elements**

  Suppose that you want to automatically insert a paragraph element (p) whenever a DITA ordered list item element (ol/li) is inserted, and also to not allow any other element besides a paragraph inside the ordered list items.

  To achieve this, the configuration file should include the following:

  ```xml
  <elementProposals path="ol/li" insertElements="p"
    possibleElements="_INSERTED_"/>
  ```
Tip:
This particular example modifies an action that inserts a list in a DITA document. There are several ways to invoke this action in the interface. For example, there is a toolbar button and an action in the DITA menu that inserts a list. However, since the configuration file only affects the Content Completion Assistant, this modification will have no effect on the behavior of the toolbar or menu action. Those actions would need to be configured separately if you want the result to be the same as the content completion proposal. For more information, see Customizing the Author Mode Editing Experience for a Framework (on page 2145).

• Example 2: Insert Complex Element Structure

For a more complex example, suppose that you want to insert a complex structure whenever a DITA `<prolog>` element is inserted.

For instance, if you want to insert the following structure inside `<prolog>` elements:

```
<prolog>
  <author/></author>
  <metadata>
    <keywords>
      <keyword></keyword>
      <keyword></keyword>
    </keywords>
  </metadata>
</prolog>
```

the configuration file should include the following:

```
<elementProposals path="prolog" insertElements="author metadata"/>
<elementProposals path="prolog/metadata" insertElements="keywords"/>
<elementProposals path="prolog/metadata/keywords" insertElements="keyword, keyword"/>
```

• Example 3: Limit Possible Elements

Suppose that you also want to limit the proposals for the `<keywords>` element to only allow the user to insert `<audience>` or `<keyword>` elements. The configuration file should include the following:

```
<elementProposals path="prolog/metadata" insertElements="keywords"
  possibleElements="audience keywords"/>
```

Suppose that you want to simply restrict your users from inserting `<image>` elements inside DITA list item elements (`<li>`), but still propose all the other elements that are defined in the associated schema. The configuration file should look like this:

```
<elementProposals path="li" rejectElements="image" />
Examples: Configuring the Attributes Proposals

- **Example 1: Automatically Insert Attributes**

  Suppose that you want to insert an `@id` attribute (with an empty value) whenever a DITA list item element (`<li>`) is inserted. The configuration file should include the following:

  ```xml
  <elementProposals path="li" insertAttributes="id"/>
  ```

- **Example 2: Limit Possible Attributes**

  Suppose that you also want to limit the number of choices for attributes that are presented to the user whenever a DITA list item element (`<li>`) is inserted. The configuration file should look like this:

  ```xml
  <elementProposals path="li" insertAttributes="id"
      possibleAttributes="id product platform audience"/>
  ```

  Suppose that you want to simply restrict your users from inserting `@conref` attributes inside DITA topics (`<topic>`) element, but still propose all the other attributes that are defined in the associated schema. The configuration file should look like this:

  ```xml
  <elementProposals path="topic" rejectAttributes="conref"/>
  ```

Related information

- Configuring the Proposals for Attribute and Element Values *(on page 2200)*
- Customizing the Rendering of Elements *(on page 2207)*

Configuring the Proposals for Attribute and Element Values

Oxygen XML Editor includes support for configuring the proposed values that appear in the *Content Completion Assistant* *(on page 3217)*. To do so, a configuration file is used, along with the associated schema, to add or replace possible values for attributes or elements that are proposed in the *Content Completion Assistant*.

For an example of a specific use-case, suppose that you want the *Content Completion Assistant* to propose several possible values for the language code when you use an `@xml:lang` attribute.

Setting up the Content Completion Configuration File

To customize the configuration file for the *Content Completion Assistant* *(on page 3217)*, follow these steps:

1. Create a new `resources` folder (if it does not already exist) in the `frameworks` directory for the particular document type (for example, `OXYGEN_INSTALL_DIR/frameworks/dita/resources`).
2. Open the *Preferences* dialog box *(Options > Preferences)* *(on page 124)* and go to Document Type Association. Select the particular document type, click the Edit button, and in the Classpath tab *(on page 144)* add a link to that `resources` folder (if it does not already exist).
3. Create a new configuration file or edit an existing one.
a. To easily create a new configuration file, you can use the Content Completion Configuration document template that is included in Oxygen XML Editor (File > New > Framework templates > Oxygen Extensions > Content Completion Configuration). The document template includes details about how each element and attribute is used in the configuration file.
b. If a configuration file (cc_config.xml) already exists for the particular document type (in the resources folder), you can modify this existing file.
c. If you extend a framework, you need to copy the content of the cc_config.xml file from the base framework and modify it (e.g. create a resources folder in your framework extension folder and place the file there). You also need to make sure that the folder that contains the cc_config.xml file in your extension (e.g. resources) is listed in the Classpath tab (on page 144) before the one from the base framework.

If you only want to make small changes or add extra rules in your custom content completion configuration file, you need to name it cc_config_ext.xml and all the rules inside it are merged with the base cc_config.xml file. The merging is done by taking all the rules specified in the cc_config_ext.xml file into consideration after processing the set of rules from the base cc_config.xml file.

4. Make the appropriate changes to your custom configuration file.
5. Save the file in the resources folder for the particular document type, using the fixed name: cc_config.xml (for example, OXYGEN_INSTALL_DIR/frameworks/dita/resources/cc_config.xml).
6. Restart the application and open an XML document. In the Content Completion Assistant you should see your customizations.

Tip:
In some cases, you can simply use the Refresh (F5) action to test your customizations, without having to restart the application.

Attention:
In the Classpath tab (on page 144), if you have references to multiple resources folders, each with its own cc_config.xml file, the first reference listed in the Classpath tab takes precedence and the multiple configuration files are not combined.

Configuring Proposed Values

For the purposes of adding or replacing the values that are proposed, the configuration file (cc_config.xml) includes a series of valueProposals instructions that will match an element or attribute name and has the following attributes:
• **path** - A path within the document that matches the element or attribute that will have its content completion proposals changed. For example:
  - `path="title"` matches all the `<title>` elements in the document.
  - `path="chapter/title"` matches only the `<title>` elements that are direct children of the `<chapter>` element.
  - `path="/xml:lang"` matches all the `@xml:lang` attributes in the document.
  - `path="title/@xml:lang"` matches only the `@xml:lang` attributes that appear on `<title>` elements.

You can use simplified forms of XPath in this attribute.

The XPath expressions can accept multiple attribute conditions and inside each condition you can use **AND/OR** boolean operators and parentheses to override the priority.

You can use one or more of the following attribute conditions (default attribute values are not taken into account):
  - `element[@attr]` - Matches all instances of the specified element that include the specified attribute.
  - `element[not(@attr)]` - Matches all instances of the specified element that do not include the specified attribute.
  - `element[@attr = "value"]` - Matches all instances of the specified element that include the specified attribute with the given value.
  - `element[@attr != "value"]` - Matches all instances of the specified element that include the specified attribute and its value is different than the one given.

**Example:** The following are examples of how you could use multiple boolean operators and parentheses inside an attribute condition:

```
*[0a and 0b or 0c and 0d]
*[0a and (0b or 0c) and 0d]
```

The following are just examples of how simplified XPath expressions might look like:
  - `elementName`
  - `//elementName`
  - `/elementName1;elementName2;elementName3`
  - `//xs:localName`
  - `//xs:documentation[@lang="en"]`

**Note:**

Using a namespace prefix requires that you declare it on the `<elementProposals>` element or on an ancestor element. For example:

```xml
<elementProposals xmlns:db5="http://docbook.org/ns/docbook"
  path="db5:listitem" insertElements="db5:para"/>
```
• **editable** - Specifies the editable state of the attribute values, as reflected in the **Attributes view (on page 627)** and the **In-place Attributes Editor (on page 630)**. The possible values for the **@editable** attribute are:
  ◦ **true** - The attribute values can be edited by choosing from a combo box or manually providing a value.
  ◦ **false** - The attribute values cannot be edited.
  ◦ **onlyAllowedItems** - The attribute values can be edited, but only by choosing from a list of proposed values, in a non-editable combo box.

The new value proposals are specified in the `<valueProposals>` element through:

• One or more `<item>` elements, which are grouped inside an `<items>` element.

**Tip:**

The `<item>` element can have a `@listValue` attribute, which can be set to **true** if you want those items to be part of a list attribute value (such as `attr="item1 item2"`).

• An `<xslt>` element that references an XSLT script that gets executed and must return an `<items>` element.

The behavior of the `<items>` or `<xslt>` elements are specified with the help of the `@action` attribute, which can have any of the following values:

• **append** - Adds new values to appear in the proposals list (default value).
• **addIfEmpty** - Adds new values to the proposals list only if no other values are contributed by the schema.
• **replace** - Replaces the values contributed by the schema with new values to appear in the proposals list.

The values in the configuration file can be specified either directly or by calling an external XSLT file that will extract data from an external source. An `<xslt>` element must be used in this situation.

**Note:**

`valueProposals` offers more flexibility compared to the old `match` element that was marked as deprecated.

```xml
<match elementName="lg" elementNS="http://www.oxygenxml.com/ns/samples">
  <items action="replace">
    <item value="stanza"/>
    <item value="refrain"/>
  </items>
</match>
```
Other Important Notes About the Configuration File

Important:

• This configuration file only affects the content completion assistance, not validation.
• To test the effects of your changes, you should Refresh the source document (on page 754).

Example: Specifying Values Directly

If you want to specify the values directly, the configuration file should look like this:

```xml
<!-- Replaces the values for an element with the local name "lg", from the given namespace -->
<valueProposals path="x:lg" xmlns:x="http://www.oxygenxml.com/ns/samples">
  <items action="replace">
    <item value="stanza"/>
    <item value="refrain"/>
  </items>
</valueProposals>

<!-- Adds two values for an attribute "type", from no namespace -->
<valueProposals path="@type" editable="onlyAllowedItems">
  <items>
    <item value="stanza"/>
    <item value="refrain"/>
  </items>
</valueProposals>
```

Example: Using Attribute Conditions

The possible values of an attribute depend on the value of another attribute from the same element:

```xml
<valueProposals path="property[@name='color']">
  <items>
    <item value="red"/>
    <item value="blue"/>
  </items>
</valueProposals>

<valueProposals path="property[@name='shape']">
  <items>
    <item value="rectangle"/>
    <item value="square"/>
  </items>
</valueProposals>
```
Example: Calling an External XSLT Script

If you want to collect values from an external XSLT script, the configuration file should include something like this:

```xml
<xslt href="../xsl/get_values_from_db.xsl" useCache="false" action="replace"/>
```

In this example, the `get_values_from_db.xsl` is executed to extract values from a database.

**Tip:**
You can use `xsl:message` as a debugging mechanism. These messages are presented in the results area at the bottom of the application whenever the Content Completion Assistant is invoked.

**Note:**
A comprehensive XSLT sample is included in the Content Completion Configuration document template (in the Framework Templates > Oxygen Extensions section of the New document wizard (on page 368)).

**Note:**
If `@useCache` is set to `false`, then the XSLT will be invoked any time the proposals are needed. If `@useCache` is set to `true`, then the XSLT is executed once and the obtained proposals are kept in a cache and returned every time the proposals are requested again. You can use the Validate (Ctrl + Shift + V (Command + Shift + V on macOS)) action to drop the cached values and recompute them.

Configuring Proposed Values in the Context Where the Content Completion was Invoked

**Web Author Customization Note:**
This particular scenario is not supported for an Oxygen XML Web Author customization.

A more complex scenario is if you want to choose the possible values to propose, depending on the context of the element where the content completion was invoked.

Suppose that you want to propose certain possible values for one property (for example, `color`) and other values for another property (for example, `shape`). If the property represents a color, then the values should represent applicable colors, while if the property represents a shape, then the values should represent applicable shapes. See the following code snippets:

Your main document:

```xml
<sampleArticle>
  <!-- The possible values for @value should be "red" and "blue" -->
</sampleArticle>
```
<property name="color" value=""/>
<!-- The possible values for @value should be "square" and "rectangle" -->
<property name="shape" value=""/>
</sampleArticle>

The content completion configuration file:

```xml
<config xmlns="http://www.oxygenxml.com/ns/ccfilter/config">
  <valueProposals path="property/@value">
    <xslt href="get_values.xsl" useCache="false" action="replace"/>
  </valueProposals>
</config>
```

The stylesheet that defines the possible values based on the context of the property on which the content completion was invoked:

```xml
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
                 xmlns:xs="http://www.w3.org/2001/XMLSchema"
                 exclude-result-prefixes="xs"
                 version="3.0">
  <xsl:param name="documentSystemID" as="xs:string"/>
  <xsl:param name="contextElementXPathExpression" as="xs:string"/>

  <xsl:template name="start">
    <xsl:apply-templates select="doc($documentSystemID)"/>
  </xsl:template>

  <xsl:template match="/">
    <xsl:variable name="propertyElement" as="element()">
      <xsl:evaluate xpath="$contextElementXPathExpression" context-item=""/>
    </xsl:variable>

    <items>
      <xsl:if test="$propertyElement/@name = 'color'">
        <item value='red'/>
        <item value='blue'/>
      </xsl:if>
      <xsl:if test="$propertyElement/@name = 'shape'">
        <item value='rectangle'/>
        <item value='square'/>
      </xsl:if>
    </items>
  </xsl:template>
</xsl:stylesheet>
```
The `contextElementXPathExpression` parameter will be bound to an XPath expression that identifies the element in the context where the content completion was invoked.

**Related information**

- Configuring the Proposals for Elements and Attributes *(on page 2192)*
- Customizing the Rendering of Elements *(on page 2207)*

### Customizing the Rendering of Elements

In addition to the support for configuring the proposals that appear in the [Content Completion Assistant *(on page 3217)*](https://www.oxygenxml.com/doc/24.1/user-guide/html/about-frameworks.html), Oxygen XML Editor also includes support for customizing how the elements are rendered. You can do this by using the [XMLNodeRendererCustomizer API extension *(on page 2269)*](https://www.oxygenxml.com/doc/24.1/javadoc/org/oxygenxml/editor/extension/renders/XmlNodeRendererCustomizer.html), but you can also use the same configuration file that is used to configure the content completion proposals.


### Setting up the Content Completion Configuration File

To customize the configuration file for the [Content Completion Assistant *(on page 3217)*](https://www.oxygenxml.com/doc/24.1/user-guide/html/about-frameworks.html), follow these steps:

1. Create a new `resources` folder (if it does not already exist) in the `frameworks` directory for the particular document type (for example, `OXYGEN_INSTALL_DIR/frameworks/dita/resources`).
2. Open the [Preferences dialog box *(Options > Preferences)* *(on page 124)*](https://www.oxygenxml.com/doc/24.1/user-guide/html/about-frameworks.html) and go to [Document Type Association](https://www.oxygenxml.com/doc/24.1/user-guide/html/about-frameworks.html). Select the particular document type, click the **Edit** button, and in the **Classpath** tab *(on page 144)* add a link to that `resources` folder (if it does not already exist).
3. Create a new configuration file or edit an existing one.
   a. To easily create a new configuration file, you can use the [Content Completion Configuration document template](https://www.oxygenxml.com/doc/24.1/user-guide/html/about-frameworks.html) that is included in Oxygen XML Editor *(File > New > Framework templates > Oxygen Extensions > Content Completion Configuration)*. The document template includes details about how each element and attribute is used in the configuration file.
   b. If a configuration file *(cc_config.xml)* already exists for the particular document type (in the `resources` folder), you can modify this existing file.
   c. If you extend a framework, you need to copy the content of the cc_config.xml file from the base framework and modify it (e.g. create a `resources` folder in your framework extension folder and place the file there). You also need to make sure that the folder that contains the cc_config.xml file in your extension (e.g. `resources`) is listed in the **Classpath** tab *(on page 144)* before the one from the base framework.
If you only want to make small changes or add extra rules in your custom content completion configuration file, you need to name it `cc_config_ext.xml` and all the rules inside it are merged with the base `cc_config.xml` file. The merging is done by taking all the rules specified in the `cc_config_ext.xml` file into consideration after processing the set of rules from the base `cc_config.xml` file.

4. Make the appropriate changes to your custom configuration file.
5. Save the file in the resources folder for the particular document type, using the fixed name: `cc_config.xml` (for example, `OXYGEN_INSTALL_DIR/frameworks/dita/resources/cc_config.xml`).
6. Restart the application and open an XML document. In the Content Completion Assistant you should see your customizations.

**Tip:**
In some cases, you can simply use the Refresh (F5) action to test your customizations, without having to restart the application.

**Attention:**
In the Classpath tab (on page 144), if you have references to multiple resources folders, each with its own `cc_config.xml` file, the first reference listed in the Classpath tab takes precedence and the multiple configuration files are not combined.

**Changing the Rendering of Elements (Their Names, Annotations, and Icons)**

For the purposes of customizing how the content completion elements are rendered, you can use the `<render>` element inside a `<elementRenderings>` element to specify how element names, their annotations, and their icons are rendered.

The `<elementRenderings>` element supports the `@platform` attribute, which can have one of the following values:

- **webapp**
  The element renderings are only applied to Oxygen XML Web Author.

- **standalone**
  The element renderings are only applied to standalone distributions of Oxygen.

- **eclipse**
  The element renderings are only applied to Eclipse plugin distributions of Oxygen.

**Note:**
If the `@platform` attribute is missing, the element renderings are applied to all types of distributions.
You can use the following attributes for the `<render>` element:

- **element**
  
  Identifies the element to be customized, in the form of a qualified name. If it does not have a prefix, it is considered to be from `noNamespace`.

- **as**
  
  Provides the name (label) that will be displayed for the element in various components in Author mode (the `Content Completion Assistant`, the breadcrumb navigation bar, the `Full Tags` display mode (on page 594), and the `Outline` (on page 539), `Elements` (on page 633), and `Attributes` (on page 627) views). This attribute is optional. If it is missing, the name of the element is used.

If you want to translate this label into another language, use the `$i18n(key_name)` editor variable (on page 331). The following code snippet shows how the DITA paragraph elements `<p>` can be translated:

```xml
<elementRenderings>
  <render element="p" as="$i18n(cc_p)"/>
</elementRenderings>
```

**Note:**

The `cc_p` id is a key that identifies the translations available for the paragraph element.

- **iconPath**
  
  Optional attribute that specifies the icon for the element. This is shown in the `Content Completion Assistant` and the `Outline view (on page 539)` in Author mode. If it is a relative path, the full path of the icon image file will be computed starting from the directory of the configuration file (for example, a value of "myImg.png" will cause Oxygen XML Editor to load "frameworks/$ (framework)/resources/myImg.png"). If you want to access a built-in resource, the value can begin with a forward slash `/`, and the image file will be searched for in the Oxygen XML Editor classpath resources (for example,"/images/OrderedList16.png" will load an icon from the built-in Oxygen XML Editor JAR file resources.

- **xml:lang**
  
  (Deprecated)

  It is recommended to use the `$i18n(key_name)` editor variable (on page 331) instead. Optional attribute that could be used to render the same element differently, depending on the language. If there are multiple `<render>` elements for the same `@element` attribute (element name) and the `@xml:lang` attribute is missing on one of them, that one will be considered the default fallback value to be used if none of the others match the language specified in the interface.

**Note:**

The default entry should be listed first, since the application tries to match them in sequence and the last match found is the one that is used.
For example, suppose that you want the name of DITA paragraph elements (<p>) to be rendered as "Paragraphe" if the language is French, "Absatz" if the language is German, and "Paragraph" if the language is English (or any other language). Your configuration file should look something like this:

```
<elementRenderings>
  <render element="p" as="Paragraph"/>
  <render element="p" as="Paragraphe" xml:lang="fr"/>
  <render element="p" as="Absatz" xml:lang="de"/>
</elementRenderings>
```

You can also use the configuration file to customize the annotations for elements. For this purpose, the `<render>` element also accepts the following element to change the tooltip annotations for an element (in both Author mode and Text mode):

`annotation`

This element can be used within the `<render>` element to customize the tooltip annotations that are displayed for the element in various components in Author mode (such as tooltips shown in the Content Completion Assistant documentation window, the breadcrumb navigation bar, the Full Tags display mode (on page 594), and the Outline (on page 539), Elements (on page 633), Attributes (on page 627) views), as well as the tooltips that are displayed when you hover over elements in Text mode. You can use HTML content to style the annotations (see the example below (on page 2211)).

Note:
If this element is missing, the styling for the annotations for that element is collected from the associated schema (on page 619).

Tip:
The annotations can also be translated in the configuration file. For example:

```
<elementRenderings>
  <render element="p" as="${i18n(cc_p)}">
    <annotation>${i18n(cc_p)}</annotation>
  </render>
</elementRenderings>
```
Other Important Notes About the Configuration File for Rendering Elements

Important:

- This configuration file only affects the content completion assistance, not validation.
- To test the effects of your changes, you should restart the application, although in some cases, you can simply use the Reload (F5) action to test your customizations.
- If the framework (on page 3219) has an associated style guide, then the annotations defined in the configuration file will take precedence over those defined in the style guide. To check to see if your framework uses a style guide, look for the following folder: ${oxygenInstallDir}frameworks/${framework}/styleguide/. If that folder exists, it is recommended that you make your annotation changes directly in the style guide, rather than in the configuration file.
- If an XMLNodeRendererCustomizer API extension (on page 2269) has been implemented for the framework and a configuration file is also used, the rendering customization for an element will be the result of merging the two. For example, if the XMLNodeRendererCustomizer implementation customizes the element name, while the configuration file specifies an icon for the element, the properties of both customizations will be rendered. However, if both implementations define the same property (for example, both specify the rendering of an element name), the customizations defined in the configuration file take precedence.
- The rendering customizations defined in the configuration file also apply to aspects of the Oxygen XML Web Author interface.

Example: Changing the Rendering of an Element

Suppose that you want to render the name of the DITA <title> element to begin with a capital letter, use a custom icon for it, and provide specific documentation for that element in the various components in Author mode. The configuration file should look like this:

```xml
<elementRenderings>
  <render element="title" as="Title" iconPath="cimg/AcceptAll16.png">
    <html xmlns="http://www.w3.org/1999/xhtml">
      <head>
        <title>Documentation for the Title Element</title>
      </head>
      <body>
        <p>A <i>heading</i> or <b>label</b> for the main parts of a topic</p>
      </body>
    </html>
  </render>
</elementRenderings>
```
Customizing Annotations in the Content Completion Assistant

Oxygen XML Editor gathers documentation from the associated schemas (DTD, XML Schema, RelaxNG) and presents it for each element or attribute. For example, if you open the Content Completion Assistant (on page 3217) for a recognized XML vocabulary, documentation is displayed for each element provided by the associated schema. Similar information is displayed when you hover over tag names presented in the Elements view (on page 633). If you hover over attributes in the Attributes view (on page 627) you also see information about each attribute, gathered from the same schema.

If you have a framework configuration (on page 140) set up for your XML vocabulary, there is a special XML configuration file that can be added to provide additional documentation information or links to additional information for certain elements and attributes.

To provide this additional information in the Content Completion Assistant, follow these steps:

1. Create a new folder in the configuration directory for the document type.

   **Example:** OXYGEN_INSTALL_DIR/frameworks/dita/styleguide

2. Use the New document wizard to create a file using the Content Completion Styleguide document template (in the Framework Templates > Oxygen Extensions section).

3. Save the file in the folder created in step 1, using the fixed name:

   contentCompletionElementsMap.xml.

4. Open the Preferences dialog box (Options > Preferences) (on page 124), go to Document Type Association, and edit the document type configuration for your XML vocabulary. Now you need to indicate where Oxygen XML Editor will locate your mapping file by doing one of the following:

   ◦ In the Classpath tab add a link to the newly created folder.

   ◦ In the Catalogs tab add a new catalog file (on page 826). The selected file needs to contain the following:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE catalog PUBLIC "-//OASIS//DTD XML Catalogs V1.1//EN"
   "http://www.oasis-open.org/committees/entity/release/1.1/catalog.dtd">
<catalog xmlns="urn:oasis:names:tc:entity:xmlns:xml:catalog">
```

where \textit{(processed Dt name)} is the name of the document type in lower case and with spaces replaced by underscores.

\begin{itemize}
\item \textbf{Note:}
  If Oxygen XML Editor finds a mapping file in both locations, the one in the \textbf{Catalogs tab} takes precedence.
\end{itemize}

5. Make the appropriate changes to your custom mapping file.

\textbf{Example:} You can look at how the DITA mapping file is configured: \texttt{OXYGEN_INSTALL_DIR/frameworks/dita/styleguide/contentCompletionElementsMap.xml}

The associated XML Schema contains additional details about how each element and attribute is used in the mapping file.

6. Re-open the application and open an XML document.

In the \textit{Content Completion Assistant (on page 3217)}, you should see the additional annotations for each element.

\textbf{Translating Annotations}

Annotations in the Content Completion Assistant can be displayed in various languages. Based on the language set for the interface, Oxygen XML Editor looks for several filename formats to determine the information to load for the content completion annotations. These files that begin with the name \textit{contentCompletionElementsMap}, are located in the \textit{styleguide} folder for each built-in framework (for example, \texttt{OXYGEN_INSTALL_DIR/frameworks/dita/styleguide}).

For example, for English, the files are loaded in the following order (from specific to more general):

\begin{itemize}
\item \texttt{contentCompletionElementsMap_en_US.xml} or \texttt{contentCompletionElementMap_en_UK.xml}, and so on
\item \texttt{contentCompletionElementsMap_en.xml}
\item \texttt{contentCompletionElementsMap.xml}
\end{itemize}

If you want the annotations to be displayed in another language, you need to create similar files for the particular language. For example, to show annotations in German, create a file with one of the following names (and store it in the \textit{styleguide} folder for your framework):

\begin{itemize}
\item \texttt{contentCompletionElementsMap_de_DE.xml}
\item \texttt{contentCompletionElementsMap_de.xml}
\end{itemize}
Customizing the Content Completion Assistant for Author Mode Only

You can customize the content of the following **Author** controls, adding items (which, when invoked, perform custom actions) or filtering the default contributed ones:

- **Content Completion Assistant** (on page 3217) window
- **Elements** view (on page 633)
- **Insert Element** menus (from the **Outline** view (on page 539) or breadcrumb (on page 602) contextual menus)

You can use the content completion customization support in a custom **framework** (on page 3219) by following this procedure:

1. Open the **Document type** configuration dialog box (on page 140) for your custom **framework** and select the **Author** tab. Next, go to the **Content Completion** tab (on page 160).

The top side of the **Content Completion** section contains the list with all the actions defined within the custom **framework** and the list of actions that you decided to include in the **Content Completion Assistant** list of proposals. The bottom side contains the list with all the items that you decided to remove from the **Content Completion Assistant** list of proposals.

2. If you want to add a custom action to the list of current **Content Completion** proposals, select the action item from the **Available actions** list and click the **Add as child** or **Add as sibling** button to include it in the **Current actions** list. A **Content Completion Item** dialog box appears, giving you the possibility to select where to provide the selected action.
3. If you want to exclude a certain item from the **Content Completion** proposals, you can use the **Add** button from the **Filter - Remove content completion items** list. The **Remove item** dialog box is displayed, allowing you to input the item name and to choose the controls that filter it. The **Item name** combo box accepts wildcards.

**Note:**

In the **Item name** drop-down menu, `<SPLIT>` refers to the action of splitting the element and creating a new one, while `<ENTER>` refers to the action of inserting a new line.
Configuring Transformation Scenarios for a Framework

When distributing a framework (on page 3219) to users, it is a good idea to have the transformation scenarios already configured. This helps the content authors publish their work in various formats. By being contained in the framework configuration, the scenarios can be distributed along with the actions, menus, toolbars, and catalogs.

These are the steps that allow you to create a transformation scenario for your framework.

1. Create an xsl folder inside your custom framework directory
   ([OXYGEN_INSTALL_DIR]\frameworks\[CUSTOM_FRAMEWORK_DIR]).

   The folder structure for the documentation framework should be:

   ```
   oxygen
   frameworks
   [CUSTOM_FRAMEWORK_DIR]
     schema
     css
     templates
     xsl
   ```

2. Create an xsl file and save it in the xsl folder. To help you get started, you can use the sample sdf.xsl file found in the sample framework customization package.

3. Open the Preferences dialog box (Options > Preferences) (on page 124) and go to Document Type Associations. Select the particular framework, click the Edit button to open Document Type Configuration dialog box (on page 140), and choose the Transformation tab. Click the ☰️ • New button and choose the appropriate type of transformation (for example, XML transformation with XSLT).

   In the New scenario dialog box, fill in the following fields:
   - Fill in the Name field with the name of your transformation scenario.
   - Set the XSL URL field to path of your custom stylesheet (for example, ${framework}/xsl/mycustom.xsl).
4. Change to the **Output** tab. Configure the fields as follows:
   - Set the **Save as** field to `${cfd}/${cfn}.html`. This means the transformation output file will have the name of the XML file and the *html* extension and will be stored in the same folder.
   - Select the **Open in Browser/System Application** option.

   **Note:**
   To set the browser or system application that will be used, open the **Preferences** dialog box (**Options > Preferences**) (on page 124), go to **Global**, and set it in the **Default Internet browser** field. This will take precedence over the default system application settings.

   - Select the **Saved file** option.

5. Click the **OK** button to save the new scenario.

Now the scenario is listed in the **Transformation** tab:
Figure 582. Transformation Tab

To test the transformation scenario that you just created, you can use the sample sdf.xml file found in the sample framework customization package. Click the Apply Transformation Scenario(s) button to display the Transform with dialog box. The scenario list contains the scenario you defined earlier. Select the SDF to HTML scenario that you just defined and click the Apply associated button. The HTML file is saved in the same folder as the XML file and displayed in the browser.

Configuring Validation Scenarios for a Framework

You can distribute a framework (on page 3219) with a series of already configured validation scenarios. Also, this provides enhanced validation support that allows you to use multiple grammars to check the document. For example, you can use Schematron rules to impose guidelines that are otherwise impossible to enforce using conventional validation.

Note:

If a main file is associated with the current file, the validation scenarios defined in the main file, along with any Schematron schema defined in the default scenarios for that particular framework, are used for the validation. These take precedence over other types of validation units defined in the default scenarios for the particular framework. For more information on main files, see Contextual Project Operations Using 'Main Files' Support (on page 418) or Modular Contextual XML Editing Using 'Main Files' Support (on page 829).

To associate a validation scenario with a specific framework, follow these steps:

1. Open the Preferences dialog box (Options > Preferences) (on page 124) and go to Document Type Association.
2. Select the document type and click the Edit button to open the Document Type Configuration dialog box (on page 140), then choose the Validation tab. This tab displays a list of document types. To set one or more of the validation scenarios listed in this tab to be used as the default validation scenario (when another one is not specified in the validation process) for a specific document type, check the Default box for that specific document type.
3. To edit an existing scenario, select the scenario and click the ✍️ Edit button. To add a new scenario, click the 🎁 New button.

In either case, a scenario configuration dialog box is displayed. It lists all the validation units for the scenario.
This scenario configuration dialog box allows you to configure the following information and options:

**Name**

The name of the validation scenario.

**URL of the file to validate**

The URL of the main module that includes the current module. It is also the entry module of the validation process when the current one is validated. To edit the URL, double-click its cell and specify the URL of the main module by doing one of the following:

- Enter the URL in the text field or select it from the drop-down list.
- Use the `Browse` drop-down button to browse for a local, remote, or archived file.
- Use the `Insert Editor Variable` button to insert an editor variable (on page 323) or a custom editor variable (on page 332).
Figure 584. Insert an Editor Variable

```
${Desktop} - My Desktop
${start-dir} - Start directory of custom validator
${standard-params} - List of standard params for command line
${cn} - The current file name without extension
${currentFileURL} - The path of the currently edited file (URL)
${cwd} - The path of current file directory (URL)
${frameworks} - Oxygen frameworks directory (URL)
${pdir} - Project directory (URL)
${oxhome} - Oxygen installation directory (URL)
${home} - The path to user home directory (URL)
${proj} - Project name
${env(VAR_NAME)} - Value of environment variable VAR_NAME
${system(var.name)} - Value of system variable var.name
```

**File type**

The type of the document that is validated in the current validation unit. Oxygen XML Editor automatically selects the file type depending on the value of the **URL of the file to validate** field.

**Validation engine**

You can select one of the engines available in Oxygen XML Editor for validation of the particular document type:

- **Default engine** - The default engine is used to run the validation for the current document type, as specified in the preferences page for that type of document (for example, XSLT preferences page (on page 245), XQuery preferences page (on page 253), XML Schema preferences page (on page 238)).
- **DITA Validation** engine - Performs DITA-specific checks in the context of the specifications (it is similar to the process when using the ![Validate and Check for Completeness action](on page 2941) in the DITA Maps Manager, but for a local file rather than an entire DITA map (on page 3218)).
- **DITA Map Validation and Completeness Check** engine - Performs a validation process that checks the DITA map document and all referenced topics and maps (it is similar to the process when using the ![Validate and Check for Completeness action](on page 2941) in the DITA Maps Manager).
- **DITA-OT Project Validation and Completeness Check** engine - Performs a validation process that checks each context from the provided DITA-OT project file (it is similar to the process when using the ![Validate and Check for Completeness action](on page 2941) in the DITA Maps Manager).
- **Table Layout Validation** engine - Looks for table layout problems (for more information, see the Report table layout problems option (on page 2946)).
Automatic validation

If this option is selected, the validation operation defined by this row is also applied by the automatic validation feature (on page 775). If the Automatic validation feature is disabled in the Document Checking preferences page (on page 226), then this option is ignored, as the preference setting has a higher priority.

Schema

This option becomes active when you set the File type to XML Document and allows you to specify the schema used for the validation unit.

Settings

Depending on the selected validation engine, clicking the Settings button either opens the Specify Schema dialog box or the Configure validation engine dialog box.

* Specify Schema Dialog Box

This dialog box allows you to specify a custom schema to be used for the validation process.

Figure 585. Specify Schema Dialog Box

The Specify Schema dialog box contains the following options:

**Use detected schema**

Uses the schema detected for the particular document (on page 817).

**Use custom schema**

Allows you to specify the schema using the following options:

- **URL** - Allows you to specify or select a URL for the schema. It also keeps a history of the last used schemas. The URL must point to the schema file that can be loaded from the local disk or from a remote server through HTTP(S), FTP(S) or a custom protocol (on page 2444). You can specify the URL by
using the text field, the history drop-down, the Insert Editor Variables (on page 323) button, or the browsing actions in the Browse drop-down list.

- **Schema type** - Select a possible schema type from this combo box that is populated based on the extension of the schema file that was entered in the URL field. The possible schema types are: XML Schema, DTD, Relax NG, Relax NG Compact, Schematron, or NVDL.

- **Embedded Schematron rules** - If you have selected XML Schema or Relax NG schemas with embedded Schematron rules and you want to use those embedded rules, select this option.

- **Extensions** - Opens a dialog box that allows you to specify Java extension JARs (on page 3220) to be used during the validation.

- **Public ID** - Allows you to specify a public ID if you have selected a DTD.

- **Schematron phase** - If you select a Schematron schema, this drop-down list allows you to select a Schematron phase that you want to use for validation. The listed phases are defined in the Schematron document.

- **Configure Validation Engine Dialog Box**

  This dialog box allows you to configure options for checking the DITA map document and all referenced topics and maps (similar to the process done when using the Validate and Check for Completeness action on page 2941 in the DITA Maps Manager).

**Note:**

The options presented in the Configure validation engine dialog box depends on type of validation engine. For example, when configuring the DITA-OT Project Validation and Completeness Check validation engine, the dialog box has slightly fewer options (omitting those that are not applicable).
Figure 586. Example of the Configure Validation Engine Dialog Box

The **Configure Validation Engine** dialog box contains the following options:

**Batch validate referenced DITA resources**

This option specifies the level of validation that applies to referenced DITA files:

- If the checkbox is left unchecked (default setting), the DITA files will be validated using the rules defined in the DTD or XML Schema declared in the document.
- If the checkbox is selected, the DITA files will be validated using rules defined in their associated validation scenario (on page 787).
Check the existence of non-DITA references resources

Extends the validation of referenced resources to non-DITA files.

Include remote resources

Select this option if you want to check that remote referenced binary resources (such as images, movie clips, ZIP archives) exist at the specified location.

Use DITAVAL filters

The content of the map is filtered by applying a profiling condition set before validation. You can choose between the following options:

- **From the current condition set** - The map is filtered using the condition set currently applied in the DITA Maps Manager view (on page 2897). Clicking the Details icon opens a topic in the Oxygen XML Editor User Guide that explains how to create a profiling condition set.

- **From all available condition sets** - For each available condition set, the map content is filtered using that set before validation.

- **From the associated transformation scenario** - The filtering condition set is specified explicitly as a DITAVAL file in the current transformation scenario associated with the DITA map.

- **Other DITAVAL files** - For each DITAVAL file from this list, the map content is filtered using the DITAVAL file before validation. Use the Add or Remove buttons to configure the list. The Add button opens a dialog box that allows you to select a local or remote path to a DITAVAL file. You can specify the path by using the text field, its history drop-down, the Insert Editor Variables (on page 323) button, or the browsing actions in the Browse drop-down list.

Report references to resources outside of the DITA map folder

If selected, it will report any references to DITA resources that are located outside the main DITA map (on page 3223) folder.

Report links to topics not referenced in DITA maps

Checks that all the topics referenced by other topics are also linked in the DITA map.
Report multiple references to the same topic

If selected, it will report warnings when a topic is referenced multiple times in the DITA map, unless a unique \texttt{@copy-to} attribute is used on the \texttt{<topicref>} element for any topic that is referenced multiple times.

For example, it will \textbf{not} report a warning if there is a topic referenced twice, but the second \texttt{<topicref>} has a \texttt{@copy-to} attribute set:

\begin{verbatim}
<topicref href="topic.dita"/>
......
<topicref href="topic.dita" copy-to="topic2.dita"/>
\end{verbatim}

On the other hand, it \textbf{will} report a warning if there is a topic referenced twice and none of the reference-type elements has a \texttt{@copy-to} attribute set or both of them have the \texttt{@copy-to} attribute set to the same value:

\begin{verbatim}
<topicref href="topic.dita" copy-to="topic2.dita"/>
......
<topicref href="topic.dita" copy-to="topic2.dita"/>
\end{verbatim}

Check for duplicate topic IDs within the DITA map context

Checks for multiple topics with the same ID in the context of the entire map.

Report duplicate key definitions

Checks the DITA map for multiple key references with the same key defined for them. This is helpful because if you have two different resources with the same value for the \texttt{@keys} attribute, all references will point to the first one encountered and the other will be ignored.

\textbf{Note:}

This option takes \textit{key scopes (on page 3058)} into account. For example, if you have something like this:

\begin{verbatim}
<topicref href="t2.dita" keys="k2"/>
<topicgroup keyscope="ks">
  <topicref href="t2.dita" keys="k2"/>
</topicgroup>
\end{verbatim}

it will not report the "k2" key as a duplicate because it is defined in a \textit{key scope (on page 3058)} on the second occurrence.

Report unreferenced key definitions
Checks the entire DITA map and reports any key definitions that are not referenced anywhere. Note that if the Use DITAVAL filters option is selected, this check will search for unreferenced key definitions based upon your selected filter.

Report unreferenced reusable elements

Checks the entire DITA map and reports any detected reusable elements that are not referenced anywhere. It looks for elements that have an ID specified in the following types of topic references:

- Any `<topicref>` that contains a `@processing-role` attribute set to `resource-only`.
- Any other referenced topic that contains elements that are reused elsewhere through a `@conref` or `@conkeyref`.

Report table layout problems

Looks for table layout problems. The types of errors that may be reported include:

- If a row has fewer cells than the number of columns detected.
- For a CALS table, if a cell has a vertical span greater than the available rows count.
- For a CALS table, if the number of `<colspecs>` is different than the number of columns detected from the table `@cols` attribute.
- For a CALS table, if the number of columns detected from the table `@cols` attribute is different than the number of columns detected in the table structure.
- For a CALS table, if the value of the `@cols`, `@rowsep`, or `@colsep` attributes are not numeric.
- For a CALS table, if the `@namest`, `@nameend`, or `@colname` attributes point to an incorrect column name.

Identify possible conflicts in profile attribute values

When the profiling attributes of a topic contain values that are not found in parent topic profiling attributes, the content of the topic is overshadowed when generating profiled output. This option reports these possible conflicts.

Report attributes and values that conflict with profiling preferences

Looks for profiling attributes and values that are not defined in the Profiling / Conditional Text preferences page (on page 187) (you can click the Profiling Preferences button to open this preferences page). It also checks if profiling attributes defined as `single-value` have multiple values set in the searched topics.

Additional Schematron checks
Allows you to select a Schematron file that Oxygen XML Editor will use for the validation of DITA resources. You can specify the path by using the text field, its history drop-down, the Insert Editor Variables (on page 323) button, or the browsing actions in the Browse drop-down list.

**Move Up**

Moves the selected validation unit up one spot in the list.

**Move Down**

Moves the selected validation unit down one spot in the list.

**Add**

Adds a new validation unit to the list.

**Remove**

Removes an existing validation unit from the list.

4. Configure any of the existing validation units according to the information above, and you can use the buttons at the bottom of the table to add, remove, or move validation units. Note that if you add a Schematron validation unit, you can also select the validation phase.

5. Click Ok.

The newly created validation scenario is now included in the list of scenarios in the Validation tab (on page 166). You can use the Default checkbox to specify that the new scenario be used as the default validation scenario when another specific scenario is not specified in the validation process.

### Customizing New Document Templates for a Framework

You can create your own custom document templates and attach them to a custom framework (on page 3219). You can then share the custom framework (on page 2288) so that all users will have access to the templates in the New document wizard (on page 368).

To create your own custom document template and have it appear in the new document wizard, follow these steps:

1. Create a new file and customize it to become a starting point for creating new files of this type.

   **Tip:**
   
   You can use editor variables (on page 323) in the template file content and they will be expanded when the files are opened. Also, see Customizing Document Templates (on page 377) for other template customization tips (for example, you could add placeholders or hints (on page 380) to assist authors).

2. Save the new template in a directory (for example, called templates) within your custom framework directory.
Oxygen XML Editor 24.1 | 17 - Framework and Author Mode Customization | 228

Attention:
The name that you use to save the template will be the name that appears in the new document wizard, including capitalization, space, and characters (for example, My Custom Template1.xml will appear in the new file wizard as My Custom Template1). You can also configure the displayed name in a properties file by following the procedure found in the Configure the Displayed Names for Document Templates (on page 379) section.

3. Open the Document Type configuration dialog box (on page 140) for that specific framework, go to the Templates tab (on page 163), and click the button in the bottom-right corner to add your new directory to the list. It is recommended that the reference be made relative to the framework directory (for example, ${frameworkDir}/templates). Binding to an absolute file (e.g., C:\some_dir\templates) makes the association difficult to share between users.

4. Click OK for all of the dialog boxes to save your changes.

5. To test the template, open the new document wizard (New toolbar button or File > New) and you should see your custom template in the folder for your custom framework (in the Framework templates section).

Related Information:
Customizing Document Templates (on page 377)

Configuring XML Catalogs

For cases where you need to reference the location of a schema file from a remote web location and an Internet connection may not be available, an XML Catalog (on page 3225) may be used to map the web location to a local file system entry. The following procedure presents an example of using an XML catalog in a custom framework (on page 3219) by modifying an XML Schema file.

1. Create a catalog file that will help the parser locate the schema for validating the XML document. The file must map the location of the schema to a local version of the schema.

   Example:

   Create a new XML file called catalog.xml and save it in your custom framework directory (/OXYGEN_INSTALL_DIR\frameworks\[CUSTOM_FRAMEWORK_DIR]). The content of the file should look like this:

   ```xml
   <catalog xmlns="urn:oasis:names:tc:entity:xmlns:xml:catalog">
     <url name="http://www.oxygenxml.com/SDF/abs.xsd" url="schema/abs.xsd"/>
     <url name="http://www.oxygenxml.com/SDF/abs.xsd"/>
   </catalog>
   ```
2. Add catalog files to your custom framework using the **Catalogs tab (on page 164)** from the **Document Type** configuration dialog box (on page 140).

To test the catalog settings, restart Oxygen XML Editor and try to validate a new sample document for your custom framework. There should be no errors.

**Example:**

The schema that validates the document refers the other file `abs.xsd` through an **import** element:

```xml
<xs:import namespace="http://www.oxygenxml.com/sample/documentation/abstracts"
          schemaLocation="http://www.oxygenxml.com/SDF/abs.xsd"/>
```

The `@schemaLocation` attribute references the `abs.xsd` file:

```xml
xsi:schemaLocation="http://www.oxygenxml.com/sample/documentation/abstracts"
                 http://www.oxygenxml.com/SDF/abs.xsd"/>
```

The catalog mapping is:

```xml
http://www.oxygenxml.com/SDF/abs.xsd -> schema/abs.xsd
```

This means that all the references to `http://www.oxygenxml.com/SDF/abs.xsd` must be resolved to the `abs.xsd` file located in the `schema` directory (note that the `schema` directory needs to be in the same folder as the XML Catalog). The URI element is used by URI resolvers (for example, to resolve a URI reference used in an XSLT stylesheet).

**Localizing Frameworks**

Oxygen XML Editor supports **framework (on page 3219)** localization (translating framework actions, buttons, and menu entries to various languages). This lets you develop and distribute a framework to users that speak other languages without changing the distributed framework. Changing the language used in Oxygen XML Editor in the Global preferences page is enough to set the right language for each framework.

To localize the content of a framework, follow this procedure:

1. Create a **translation.xml** file that contains all the translation (key, value) mappings. The **translation.xml** has the following format:

```xml
<translation>
  <languageList>
    <language description="English" lang="en_US"/>
    <language description="German" lang="de_DE"/>
    <language description="French" lang="fr_FR"/>
  </languageList>
</translation>
```
Oxygen XML Editor matches the GUI language with the language set in the translation.xml file. If this language is not found, the first available language declared in the <languagelist> tag for the corresponding framework is used.

2. The translation.xml file must be stored in a directory named i18n located in the framework folder. You also need to add a reference to the i18n directory in the Classpath list corresponding to the edited document type (on page 144).

3. After you create this file, you can use the keys defined in it to customize the name and description of the following:
   - Actions
   - Menu entries
   - Contextual menus
   - Toolbars
   - Static CSS content

   For example, if you want to localize the bold action, open the Preferences dialog box (Options > Preferences) (on page 124) and go to Document Type Association. Use the New or Edit button to open the Document type configuration dialog box (on page 140), go to Author > Actions, and rename the bold action to ${i18n(translation_key)}. Actions with a name format other than ${i18n(translation_key)} are not localized. Translation_key corresponds to the key from the translation.xml file.

4. Next, open the translation.xml file and edit the translation entry if it exists or create one if it does not exist. This is an example of an entry in the translation.xml file:

To use a description from the translation.xml file in the Java code used by your custom framework, use the new ro.sync.ecss.extensions.api.AuthorAccess.getAuthorResourceBundle() API method to request the associated value for a certain key. This allows all the dialog boxes that you present from your custom operations to have labels translated in multiple languages.
You can also reference a key directly in the CSS content:

```css
title:before{
    content: "${i18n(title.key)} : ";
}
```

**Note:**
You can enter any language you want in the `<langagelist>` tag and any number of keys.

**DocBook Example:**

The `translation.xml` file for the DocBook framework is located here: `[OXYGEN_INSTALL_DIR]/frameworks/docbook/i18n/translation.xml`. In the Classpath list corresponding to the DocBook document type, the following entry was added: `${framework}/i18n/`.

To see how the DocBook actions are defined to use these keys for their name and description, open the Preferences dialog box (Options > Preferences) and go to Document Type Association > Author > Actions. If you look in the Java class `ro.sync.ecss.extensions.docbook.table.SADocbookTableCustomizerDialog` available in the `oxygen-sample-framework` module of the Oxygen SDK Maven archetype, you can see how the new `ro.sync.ecss.extensions.api.AuthorResourceBundle` API is used to retrieve localized descriptions for various keys.

**Framework Java Extensibility Guide**

Advanced users can extend the functionality of custom frameworks and Author mode. The Oxygen SDK is also available to provide developers the ability to extend the functionality of Oxygen XML Editor.

You can add extensions to your custom framework (document type) by using the Extensions tab from the Document Type configuration dialog box.

**Note:**
It is possible for a plugin to share the same classes with a framework. For further details, go to How to Share the Classloader Between a Framework and a Plugin.

**Related Information:**
- Extending Oxygen With the SDK
- SDK Common Use Cases
Configuring an Extensions Bundle

All extensions (on page 3222) that are provided by Oxygen XML Editor are included in a single bundle.

Note:
The individual extensions can still be set (open the Preferences dialog box (Options > Preferences) (on page 124), go to Document Type Association, double-click a document type, and go to the extension tab), and if present, they take precedence over the single provider. However, this practice is discouraged and the single provider should be used instead.

The extensions bundle is represented by the ro.sync.ecss.extensions.api.ExtensionsBundle class. The provided implementation of the ExtensionsBundle is instantiated when the Document Type Association (on page 3218) rules defined for the custom framework (on page 3219) matches a document opened in the editor. Therefore, references to objects that need to be persistent throughout the application running session must not be kept in the bundle because the next detection event can result in creating another ExtensionsBundle instance.

To configure an extensions bundle, follow this procedure:

1. Create a new Java project in your IDE. Create a lib folder in the Java project folder and copy in it the oxygen.jar file from the /OXYGEN_INSTALL_DIR/lib folder.
2. Create the class (for example, simple.documentation.framework.SDFExtensionsBundle) to extend the abstract class ro.sync.ecss.extensions.api.ExtensionsBundle.
   For example:
   ```java
   public class SDFExtensionsBundle extends ExtensionsBundle {
   }
   ```
3. A Document Type ID and a short description should be defined by implementing the getDocumentTypeID and getDescription methods. The Document Type ID is used to uniquely identify the current framework. Such an ID must be provided especially if options related to the framework need to be persistently stored and retrieved between sessions.
   For example:
   ```java
   public String getDocumentTypeID() {
   }
   
   public String getDescription() {
       return "A custom extensions bundle used for the Simple Document" +
              "Framework document type";
   }
   ```
4. [Optional] To be notified about the activation of the custom Author Extension in relation with an open document, ro.sync.ecss.extensions.api.AuthorExtensionStateListener should be implemented. The activation and deactivation events received by this listener should be used to perform custom initializations and to register or remove listeners such as ro.sync.ecss.extensions.api.AuthorListener,
ro.sync.ecss.extensions.api.AuthorMouseListener, or ro.sync.ecss.extensions.api.AuthorCaretListener.

The custom Author Extension state listener should be provided by implementing the createAuthorExtensionStateListener method.

For example:

```java
public AuthorExtensionStateListener createAuthorExtensionStateListener() {
    return new SDFAuthorExtensionStateListener();
}
```

The AuthorExtensionStateListener is instantiated and notified about the activation of the framework when the rules of the Document Type Association match a document opened in the Author editing mode. The listener is notified about the deactivation when another framework is activated for the same document, the user switches to another mode or the editor is closed. A complete description and implementation of ro.sync.ecss.extensions.api.AuthorExtensionStateListener can be found in Implementing an Author Extension State Listener (on page 2245).

If Schema-Aware mode (on page 180) is active in Oxygen XML Editor, all actions that can generate invalid content will be redirected toward the ro.sync.ecss.extensions.api.AuthorSchemaAwareEditingHandler. The handler can resolve a specific case, let the default implementation take place, or reject the edit entirely by throwing ro.sync.ecss.extensions.api.InvalidEditException. The actions that are forwarded to this handler include typing, delete, or paste.

For more details about this handler, see Handling Schema-Aware Editing Events (on page 2283).

5. [Optional] You can customize the content completion proposals by creating a schema manager filter extension. The interface that declares the methods used for content completion proposals filtering is ro.sync.contentcompletion.xml.SchemaManagerFilter. The filter can be applied on elements, attributes, or on their values. The createSchemaManagerFilter method is responsible for creating the content completion filter. A new SchemaManagerFilter will be created each time a document matches the rules defined by the Document Type Association that contains the filter declaration.

For example:

```java
public SchemaManagerFilter createSchemaManagerFilter() {
    return new SDFSchemaManagerFilter();
}
```

A detailed presentation of the schema manager filter can be found in the Configuring a Content Completion Handler (on page 2238) section.

6. [Optional] The Author mode supports link-based navigation between documents and document sections. Therefore, if the document contains elements defined as links to other elements (for example, links based on the @id attributes), the extension should provide the means to find the referenced content. To do this, an implementation of the ro.sync.ecss.extensions.api.link.ElementLocatorProvider interface should be returned by the createElementLocatorProvider method. Each time an element pointed by a link needs to be located, the method is invoked.
For example:

```java
public ElementLocatorProvider createElementLocatorProvider() {
    return new DefaultElementLocatorProvider();
}
```

For more information on how to implement an element locator provider, see the Configuring a Link Target Element Finder (on page 2263) section.

7. [Optional] The drag and drop functionality can be extended by implementing the `ro.sync.exml.editor.xmleditor.pageauthor.AuthorDnDListener` interface. Relevant methods from the listener are invoked when the mouse is dragged, moved over, or exits the Author editing mode, when the drop action changes, and when the drop occurs. Each method receives the `DropTargetEvent` containing information about the drag and drop operation. The drag and drop extensions are available in Author mode for both Oxygen XML Editor Eclipse plugin and standalone application. The Text mode corresponding listener is available only for Oxygen XML Editor Eclipse plugin. The methods corresponding to each implementation are: `createAuthorAWTDndListener`, `createTextSWTDndListener`, and `createAuthorSWTDndListener`.

```java
public AuthorDnDListener createAuthorAWTDndListener() {
    return new SDFAuthorDndListener();
}
```

For more details about the Author mode drag and drop listeners, see the Configuring a custom Drag and Drop Listener (on page 2241) section.

8. [Optional] Another extension that can be included in the bundle is the reference resolver. For example, the references represented by the `ref` element and the attribute indicating the referenced resource is `location`. To be able to obtain the content of the referenced resources you will have to implement a Java extension class that implements `ro.sync.ecss.extensions.api.AuthorReferenceResolver`. The method responsible for creating the custom references resolver is `createAuthorReferenceResolver`. The method is called each time a document opened in an Author editing mode matches the Document Type Association where the extensions bundle is defined. The instantiated references resolver object is kept and used until another extensions bundle corresponding to another document type is activated as result of the detection process.

```java
public AuthorReferenceResolver createAuthorReferenceResolver() {
    return new ReferencesResolver();
}
```

A more detailed description of the references resolver can be found in the Configuring a References Resolver (on page 2241) section.

9. [Optional] To be able to dynamically customize the default CSS styles for a certain `ro.sync.ecss.extensions.api.node.AuthorNode`, an implementation of
ro.sync.ecss.extensions.api.StylesFilter can be provided. The extensions bundle method responsible for creating the StylesFilter is createAuthorStylesFilter. The method is called each time a document opened in an Author editing mode matches the Document Type Association where the extensions bundle is defined. The instantiated filter object is kept and used until another extensions bundle corresponding to another document type is activated as a result of the detection process.

For example:

```java
public StylesFilter createAuthorStylesFilter() {
    return new SDFStylesFilter();
}
```

See the Configuring CSS Styles Filter (on page 2262) section for more details about the styles filter extension.

10. [Optional] To edit data in custom tabular format, implementations of the ro.sync.ecss.extensions.api.AuthorTableCellSpanProvider and the ro.sync.ecss.extensions.api.AuthorTableColumnWidthProvider interfaces should be provided. The two methods from the ExtensionsBundle specifying these two extension points are createAuthorTableCellSpanProvider and createAuthorTableColumnWidthProvider.

For example:

```java
public AuthorTableCellSpanProvider createAuthorTableCellSpanProvider() {
    return new TableCellSpanProvider();
}
```

```java
public AuthorTableColumnWidthProvider createAuthorTableColumnWidthProvider() {
    return new TableColumnWidthProvider();
}
```

The two table information providers are not reused for different tables. The methods are called for each table in the document so new instances should be provided every time. Read more about the cell span and column width information providers in Configuring a Table Cell Span Provider (on page 2254) and Configuring a Table Column Width Provider (on page 2249) sections.

If the functionality related to one of the previous extension points does not need to be modified, then the developed ro.sync.ecss.extensions.api.ExtensionsBundle should not override the corresponding method and leave the default base implementation to return null.

11. [Optional] An XML vocabulary can contain links to various areas of a document. If the document contains elements defined as links, you can choose to present a more relevant text description for each link. To do this, an implementation of the ro.sync.ecss.extensions.api.link.LinkTextResolver interface should be returned by the createLinkTextResolver method. This implementation is used each time the oxy_link-text() function (on page 2365) is encountered in the CSS styles associated with an element.

For example:
public LinkTextResolver createLinkTextResolver() {
    return new DitaLinkTextResolver();
}


12. Pack the compiled class into a JAR (on page 3220) file.
13. Copy the JAR file into your custom framework directory (for example, frameworks/sdf).
14. Add the JAR file to the class path. To do this, open the Preferences dialog box (Options > Preferences) (on page 124), go to Document Type Association, select the document type (for example, SDF), click the Edit button, select the Classpath tab, and click the Add button. In the displayed dialog box, enter the location of the JAR file relative to the Oxygen XML Editor frameworks folder.
15. Register the Java class by going to the Extensions tab. Click the Choose button and select the name of the class (for example, SDFExtensionsBundle).

Note:
The complete source code for framework customization examples can be found in the oxygen-sample-framework module of the Oxygen SDK, available as a Maven archetype on the Oxygen XML Editor website.

Related information
https://www.oxygenxml.com/InstData/Editor/SDK/javadoc/ro/sync/ecss/extensions/api/ExtensionsBundle.html

Adding a Custom Image Decorator for Author Mode

The AuthorImageDecorator extension point allows you to add a custom decorator over images in Author mode. For example, you could use it to add a message over an image informing the user that they can double-click the image to edit it.

How to Implement an AuthorImageDecorator

To implement your own AuthorImageDecorator, follow this procedure:

1. Implement the `ro.sync.ecss.extensions.api.AuthorImageDecorator` interface.
2. To instruct Oxygen XML Editor to use this newly created implementation, use either of the following methods:
a. If you have configured an extensions bundle (on page 2232), you can return the `AuthorImageDecorator` implementation using the `ro.sync.ecss.extensions.api.ExtensionsBundle.getAuthorImageDecorator()` method.

b. Specify the `AuthorImageDecorator` in the **Author image decorator** individual extension in the Extensions tab (on page 167) of the **Document Type** configuration dialog box (on page 140) for your particular document type.

**Example**

The following example illustrates an implementation for presenting a simple message over an image that informs the user that they can double-click the image to edit it:

```java
/**
 * Custom Author image decorator for drawing string over images.
 */
public class CustomAuthorImageDecorator extends AuthorImageDecorator {

    /**
     * @see ro.sync.ecss.extensions.api.AuthorImageDecorator#paint
     *(ro.sync.exml.view.graphics.Graphics, int, int, int,
     * ro.sync.exml.view.graphics.Rectangle,
     * ro.sync.ecss.extensions.api.node.AuthorNode,
     * ro.sync.ecss.extensions.api.AuthorAccess, boolean)
     */
    @Override
    public void paint(Graphics g, int x, int y, int imageWidth, int imageHeight,
                      Rectangle originalSize, AuthorNode element,
                      AuthorAccess authorAccess, boolean wasAnnotated) {
        if ("image".equals(CommonsOperationsUtil.getLocalName(element.getName()))) {
            g.drawString("[Double-click to edit image]",
                         // Draw near the top-left corner
                         x + 15,
                         y + 15);
        }
    }
```

**Example result:** In the top-left corner of the image, the following message will be displayed: **[Double-click to edit image]**.

**Adding Custom Persistent Highlights**

The **Author API** includes a class that allows you to create or remove custom persistent highlights, set new properties for the highlights, and customize their appearance. An example of a possible use case would be
if you want to implement your own way of editing review comments. The custom persistent highlights get serialized in the XML document as processing instructions, with the following format:

```
<?oxy_custom_start prop1="val1"....?> xml content <?oxy_custom_end?>
```

This functionality is available through the `AuthorPersistentHighlighter` class that is accessible through the `AuthorEditorAccess#getPersistentHighlighter()` method.

For more information, see the JavaDoc details for this class at [https://www.oxygenxml.com/InstData/Editor/SDK/javadoc/ro/sync/ecss/extensions/api/highlights/AuthorPersistentHighlighter.html](https://www.oxygenxml.com/InstData/Editor/SDK/javadoc/ro/sync/ecss/extensions/api/highlights/AuthorPersistentHighlighter.html).

### Configuring the Automatic ID Generation and Unique Attributes Recognizer

The `ro.sync.ecss.extensions.api.UniqueAttributesRecognizer` interface can be implemented if you want to provide for your framework (on page 3219) the following features:

- **Automatic ID generation** - You can automatically generate unique IDs for newly inserted elements. Implementations are already available for the DITA and DocBook frameworks (on page 3219). The following methods can be implemented to accomplish this: `assignUniqueIDs(int startOffset, int endOffset)`, `isAutoIDGenerationActive()`

- **Avoiding copying unique attributes when "Split" is called inside an element** - You can split the current block element (on page 3216) by pressing the "Enter" key and then choosing "Split". This is a very useful way to create new paragraphs, for example. All attributes are by default copied on the new element but if those attributes are IDs you sometimes want to avoid creating validation errors in the editor. Implementing the following method, you can decide whether or not an attribute should be copied during the split: `boolean copyAttributeOnSplit(String attrQName, AuthorElement element)`

**Tip:**

The `ro.sync.ecss.extensions.commons.id.DefaultUniqueAttributesRecognizer` class is an implementation of the interface that can be extended by your customization to provide easy assignation of IDs in your framework (on page 3219). You can also check out the DITA and DocBook implementations of `ro.sync.ecss.extensions.api.UniqueAttributesRecognizer` to see how they were implemented and connected to the extensions bundle.

**Note:**

The complete source code for framework customization examples can be found in the `oxygen-sample-framework` module of the Oxygen SDK, available as a Maven archetype on the Oxygen XML Editor website.

### Configuring Content Completion Proposals

You can filter or contribute to proposals offered for content completion by implementing the `ro.sync.contentcompletion.xml.SchemaManagerFilter` interface.
import java.util.List;
import ro.sync.contentcompletion.xml.CIAttribute;
import ro.sync.contentcompletion.xml.CIElement;
import ro.sync.contentcompletion.xml.CIValue;
import ro.sync.contentcompletion.xml.Context;
import ro.sync.contentcompletion.xml.SchemaManagerFilter;
import ro.sync.contentcompletion.xml.WhatAttributesCanGoHereContext;
import ro.sync.contentcompletion.xml.WhatElementsCanGoHereContext;
import ro.sync.contentcompletion.xml.WhatPossibleValuesHasAttributeContext;

public class SDFSchemaManagerFilter implements SchemaManagerFilter {

You can implement the various callbacks of the interface either by returning the default values given by Oxygen XML Editor or by contributing to the list of proposals. The filter can be applied on elements, attributes or on their values. Attributes filtering can be implemented using the filterAttributes method and changing the default content completion list of ro.sync.contentcompletion.xml.CIAttribute for the element provided by the current ro.sync.contentcompletion.xml.WhatAttributesCanGoHereContext context. For example, the SDFSchemaManagerFilter checks if the element from the current context is the table element and adds the frame attribute to the table list of attributes.

/**
 * Filter attributes of the "table" element.
 */
public List<CIAttribute> filterAttributes(List<CIAttribute> attributes, WhatAttributesCanGoHereContext context) {
    // If the element from the current context is the 'table' element add the
    // attribute named 'frame' to the list of default content completion proposals
    if (context != null) {
        ContextElement contextElement = context.getParentElement();
        if ("table".equals(contextElement.getQName())) {
            CIAttribute frameAttribute = new CIAttribute();
            frameAttribute.setName("frame");
            frameAttribute.setRequired(false);
            frameAttribute.setFixed(false);
            frameAttribute.setDefaultValue("void");
            if (attributes == null) {
                attributes = new ArrayList<CIAttribute>();
            }
            attributes.add(frameAttribute);
        }
    }
}
}
The elements that can be inserted in a specific context can be filtered using the `filterElements` method. The `SDFSchemaManagerFilter` uses this method to replace the `td` child element with the `th` element when `header` is the current context element.

```java
public List<CIElement> filterElements(List<CIElement> elements,
        WhatElementsCanGoHereContext context) {
    // If the element from the current context is the 'header' element remove the 'td' element from the list of content completion proposals and add the 'th' element.
    if (context != null) {
        Stack<ContextElement> elementStack = context.getElementStack();
        if (elementStack != null) {
            ContextElement contextElement = context.getElementStack().peek();
            if ("header".equals(contextElement.getQName())) {
                if (elements != null) {
                    for (Iterator<CIElement> iterator = elements.iterator(); iterator.hasNext();)
                    {
                        CIElement element = iterator.next();
                        // Remove the 'td' element
                        if ("td".equals(element.getQName())) {
                            elements.remove(element);
                            break;
                        }
                    }
                } else {
                    elements = new ArrayList<CIElement>();
                }
                // Insert the 'th' element in the list of content completion proposals
                CIElement thElement = new SDSElement();
                thElement.setName("th");
                elements.add(thElement);
            }
        }
    } else {
        // If the given context is null then the given list of content completion proposals contains global elements.
    }
    return elements;
}
```
The elements or attributes values can be filtered using the `filterElementValues` or `filterAttributeValues` methods.

**Note:**
The complete source code for framework customization examples can be found in the `oxygen-sample-framework` module of the Oxygen SDK, available as a Maven archetype on the Oxygen XML Editor website.

### Configuring a Custom Drag and Drop Listener

Sometimes it is useful to perform various operations when certain objects are dropped from outside sources in the editing area. You can choose from three interfaces to implement depending on whether you are using the Eclipse plugin or the standalone version of the application, or if you want to add the handler for the Text or Author modes.

**Interfaces for the Drag and Drop Listener**

- `ro.sync.exml.editor.xmleditor.pageauthor.AuthorDnDListener`
  Receives callbacks from the standalone application for Drag And Drop in Author mode.

- `com.oxygenxml.editor.editors.author.AuthorDnDListener`
  Receives callbacks from the Eclipse plugin for Drag And Drop in Author mode.

- `com.oxygenxml.editor.editors.TextDnDListener`
  Receives callbacks from the Eclipse plugin for Drag And Drop in Text mode.

**Note:**
The complete source code for framework customization examples can be found in the `oxygen-sample-framework` module of the Oxygen SDK, available as a Maven archetype on the Oxygen XML Editor website.

To configure how dropped URLs or XHTML fragments are handled in documents, see Handling When URLs or XHTML Fragments are Dropped or Pasted in Author Mode (on page 2285).

**Related Information:**
- Customizing Smart Paste Support (on page 2188)

### Configuring a Reference Resolver

This information is helpful if you need to provide a handler for resolving references and obtain the content they reference. For example, suppose the element that has references is `ref` and the attribute indicating the referenced resource is `location`. You need to implement a Java extension class for obtaining the referenced resources.
1. Create the class `simple.documentation.framework.ReferencesResolver`. This class must implement the `ro.sync.ecss.extensions.api.AuthorReferenceResolver` interface.

```
import ro.sync.ecss.extensions.api.AuthorReferenceResolver;
import ro.sync.ecss.extensions.api.AuthorAccess;
import ro.sync.ecss.extensions.api.node.AttrValue;
import ro.sync.ecss.extensions.api.node.AuthorElement;
import ro.sync.ecss.extensions.api.node.AuthorNode;

public class ReferencesResolver
    implements AuthorReferenceResolver {

    public boolean hasReferences(AuthorNode node) {
        boolean hasReferences = false;
        if (node.getType() == AuthorNode.NODE_TYPE_ELEMENT) {
            AuthorElement element = (AuthorElement) node;
            if ("ref".equals(element.getLocalName())) {
                AttrValue attrValue = element.getAttribute("location");
                if (attrValue != null) {
                    hasReferences = attrValue != null;
                }
            }
        }
        return hasReferences;
    }

    public String getDisplayName(AuthorNode node) {
        String displayName = "ref-fragment";
        if (node.getType() == AuthorNode.NODE_TYPE_ELEMENT) {
            AuthorElement element = (AuthorElement) node;
            if ("ref".equals(element.getLocalName())) {
                AttrValue attrValue = element.getAttribute("location");
                if (attrValue != null) {
                    displayName = attrValue.getValue();
                }
            }
        }
        return displayName;
    }
```

2. The `hasReferences` method verifies if the handler considers the node to have references. It takes `AuthorNode` as an argument that represents the node that will be verified. The method will return `true` if the node is considered to have references. In the following example, to be a reference, the node must be an element with the name `ref` and it must have an attribute named `location`.

3. The method `getDisplayName` returns the display name of the node that contains the expanded referenced content. It takes `AuthorNode` as an argument that represents the node that needs the display name. The referenced content engine will ask this `AuthorReferenceResolver` implementation for the display name for each node that is considered a reference. In the following example, the display name is the value of the `location` attribute from the `ref` element.
The method `resolveReference` resolves the reference of the node and returns a `SAXSource` with the parser and its input source. It takes `AuthorNode` as an argument that represents the node that needs the reference resolved, the `systemID` of the node, the `AuthorAccess` with access methods to the `Author` mode data model and a `SAX EntityResolver` that resolves resources that are already opened in another editor or resolve resources through the `XML Catalog (on page 3225)`. In the implementation, you need to resolve the reference relative to the `systemID`, and create a parser and an input source over the resolved reference.

```java
public SAXSource resolveReference(
    AuthorNode node,
    String systemID,
    AuthorAccess authorAccess,
    EntityResolver entityResolver) {
    SAXSource saxSource = null;

    if (node.getType() == AuthorNode.NODE_TYPE_ELEMENT) {
        AuthorElement element = (AuthorElement) node;
        if ("ref".equals(element.getLocalName())) {
            AttrValue attrValue = element.getAttribute("location");
            if (attrValue != null) {
                String attrStringVal = attrValue.getValue();
                try {
                    URL absoluteUrl = new URL(new URL(systemID),
                        authorAccess.getUtilAccess().correctURL(attrStringVal));

                    InputSource inputSource = entityResolver.resolveEntity(null,
                        absoluteUrl.toString());
                    if (inputSource == null) {
                        inputSource = new InputSource(absoluteUrl.toString());
                    }
                    XMLReader xmlReader = authorAccess.newNonValidatingXMLReader();
                    xmlReader.setEntityResolver(entityResolver);
                    saxSource = new SAXSource(xmlReader, inputSource);
                } catch (MalformedURLException e) {
                    logger.error(e, e);
                } catch (SAXException e) {
                    logger.error(e, e);
                } catch (IOException e) {
                    logger.error(e, e);
                }
            }
        }
    }
    return saxSource;
}
```
5. The method `getReferenceUniqueID` should return a unique identifier for the node reference. The unique identifier is used to avoid resolving the references recursively. The method takes `AuthorNode` as an argument that represents the node with the reference. In the following example, the unique identifier is the value of the `location` attribute from the `ref` element.

```java
public String getReferenceUniqueID(AuthorNode node) {
    String id = null;
    if (node.getType() == AuthorNode.NODE_TYPE_ELEMENT) {
        AuthorElement element = (AuthorElement) node;
        if ("ref".equals(element.getLocalName())) {
            AttrValue attrValue = element.getAttribute("location");
            if (attrValue != null) {
                id = attrValue.getValue();
            }
        }
    }
    return id;
}
```

6. The method `getReferenceSystemID` should return the `systemID` of the referenced content. It takes `AuthorNode` as an argument that represents the node with the reference and the `AuthorAccess` with access methods to the `Author` mode data model. For example, the value of the `location` attribute is used from the `ref` element and resolved relatively to the XML base URL of the node.

```java
public String getReferenceSystemID(AuthorNode node, 
                                  AuthorAccess authorAccess) {
    String systemID = null;
    if (node.getType() == AuthorNode.NODE_TYPE_ELEMENT) {
        AuthorElement element = (AuthorElement) node;
        if ("ref".equals(element.getLocalName())) {
            AttrValue attrValue = element.getAttribute("location");
            if (attrValue != null) {
                String attrStringVal = attrValue.getValue();
                try {
                    URL absoluteUrl = new URL(node.getXMLBaseURL(),
                                            authorAccess.getUtilAccess().correctURL(attrStringVal));
                    systemID = absoluteUrl.toString();
                }
            }
        }
    }
    return systemID;
}
```
In the listing below, the XML document contains the `ref` element:

```xml
<ref location="referenced.xml">Reference</ref>
```

When no reference resolver is specified, the reference has the following layout:

**Figure 587. Reference with no specified reference resolver**

Reference without a reference resolver

dReference

When the above implementation is configured, the reference has the expected layout:

**Figure 588. Reference with reference resolver**

Reference with a reference resolver

Reference

Edit referenced content

Referred paragraph

---

**Note:**

The complete source code for framework customization examples can be found in the `oxygen-sample-framework` module of the Oxygen SDK, available as a Maven archetype on the Oxygen XML Editor website.

---

**Configuring a State Listener for Author Mode**

The `ro.sync.ecss.extensions.api.AuthorExtensionStateListener` implementation is notified when the Author mode extension (where the listener is defined) is activated or deactivated in the document type detection process.

```java
import ro.sync.ecss.extensions.api.AuthorAccess;
import ro.sync.ecss.extensions.api.AuthorExtensionStateListener;

public class SDFAuthorExtensionStateListener implements AuthorExtensionStateListener {
```
private AuthorListener sdfAuthorDocumentListener;
private AuthorMouseListener sdfMouseListener;
private AuthorCaretListener sdfCaretListener;
private OptionListener sdfOptionListener;

When the association rules of the framework (on page 3219) (document type) configuration match that of a document open in the Author editing mode, the activation event received by this listener should be used to perform custom initializations and to register listeners such as ro.sync.ecss.extensions.api.AuthorListener, ro.sync.ecss.extensions.api.AuthorMouseListener, or ro.sync.ecss.extensions.api.AuthorCaretListener.

```java
public void activated(AuthorAccess authorAccess) {
    // Get the value of the option.
    String option = authorAccess.getOptionsStorage().getOption("sdf.custom.option.key", "<<null>>");
    // Use the option for some initializations...

    // Add an OptionListener.
    authorAccess.getOptionsStorage().addOptionListener(sdfOptionListener);

    // Add author DocumentListeners.
    sdfAuthorDocumentListener = new SDFAuthorListener();
    authorAccess.getDocumentController().addAuthorListener(sdfAuthorDocumentListener);

    // Add MouseListener.
    sdfMouseListener = new SDFAuthorMouseListener();
    authorAccess.getEditorAccess().addAuthorMouseListener(sdfMouseListener);

    // Add CaretListener.
    sdfCaretListener = new SDFAuthorCaretListener();
    authorAccess.getEditorAccess().addAuthorCaretListener(sdfCaretListener);

    // Other custom initializations...
}
```

The authorAccess parameter received by the activated method can be used to gain access to specific Author mode actions and informations related to components such as the editor, document, workspace, tables, or the change tracking manager.

If options specific to the custom developed Author Extension need to be stored or retrieved, a reference to the ro.sync.ecss.extensions.api.OptionsStorage can be obtained by calling the getOptionsStorage method from the authorAccess. The same object can be used to register ro.sync.ecss.extensions.api.OptionListener
listeners. An option listener is registered in relation with an option key and will be notified about the value changes of that option.

An AuthorListener can be used if events related to the Author mode document modifications are of interest. The listener can be added to the ro.sync.ecss.extensions.api.AuthorDocumentController. A reference to the document controller is returned by the getDocumentController method from the authorAccess. The document controller can also be used to perform operations involving document modifications.

To provide access to the Author mode component-related functionality and information, the authorAccess has a reference to the ro.sync.ecss.extensions.api.access.AuthorEditorAccess that can be obtained when calling the getEditorAccess method. At this level, AuthorMouseListener and AuthorCaretListener can be added to provide notification of mouse and cursor events that occur in the Author editor mode.

The deactivation event is received when another framework is activated for the same document, the user switches to another editor mode or the editor is closed. The deactivate method is typically used to unregister the listeners previously added on the activate method and to perform other actions. For example, options related to the deactivated Author Extension can be saved at this point.

```java
public void deactivated(AuthorAccess authorAccess) {
    // Store the option.
    authorAccess.getOptionsStorage().setOption(
        "sdf.custom.option.key", optionValue);

    // Remove the OptionListener.
    authorAccess.getOptionsStorage().removeOptionListener(sdfOptionListener);

    // Remove DocumentListeners.
    authorAccess.getDocumentController().removeAuthorListener(
        sdfAuthorDocumentListener);

    // Remove MouseListener.
    authorAccess.getEditorAccess().removeAuthorMouseListener(sdfMouseListener);

    // Remove CaretListener.
    authorAccess.getEditorAccess().removeAuthorCaretListener(sdfCaretListener);

    // Other actions...
}
```

**Note:**
The complete source code for framework customization examples can be found in the oxygen-sample-framework module of the Oxygen SDK, available as a Maven archetype on the Oxygen XML Editor website.
Configuring Tables

There are standard CSS properties used to indicate what elements are tables, table rows and table cells. What CSS is missing is the possibility to indicate the cell spanning, row separators or the column widths. Oxygen XML Editor offers support for adding extensions to solve these problems.

The table in this example is a simple one. The header must be formatted in a different way than the ordinary rows, so it will have a background color.

```css
table{
    display:table;
    border:1px solid navy;
    margin:1em;
    max-width:1000px;
    min-width:150px;
}
table[width] {
    width:attr(width, length);
}
tr, header{
    display:table-row;
}
header{
    background-color: silver;
    color: inherit
}
td{
    display:table-cell;
    border:1px solid navy;
    padding:1em;
}
```

Suppose that in the schema, the `<td>` tag has the attributes `@row_span` and `@column_span` that are not automatically recognized by Oxygen XML Editor, a Java extension will be implemented that will provide information about the cell spanning. See the section Configuring a Table Cell Span Provider (on page 2254).

Suppose the column widths are specified by the `@width` attribute of the `<customcol>` elements that are not automatically recognized by Oxygen XML Editor. It is necessary to implement a Java extension that will provide information about the column widths. For more information, see Configuring a Table Column Width Provider (on page 2249).
The table from the example does not make use of the attributes @colsep and @rowsep (which are automatically recognized) but if you want the rows to be separated by horizontal lines, it is necessary to implement a Java extension that will provide information about the row and column separators. For more information, see Configuring a Table Cell Row and Column Separator Provider (on page 2258).

Configuring a Table Column Width Provider

In a custom framework (on page 3219), the <table> element as well as the table columns can have specified widths. For these widths to be considered by Author mode, you need to provide the means for determining them. As explained in Configuring Tables (on page 2248), if you use the table element attribute width Oxygen XML Editor can determine the table width automatically. In this example the table has <col> elements with @width attributes that are not recognized by default. You will need to implement a Java extension class to determine the column widths.

1. Create the class simple.documentation.framework.TableColumnWidthProvider. This class must implement the ro.sync.ecss.extensions.api.AuthorTableColumnWidthProvider interface.

   ```java
   import ro.sync.ecss.extensions.api.AuthorAccess;
   import ro.sync.ecss.extensions.api.AuthorOperationException;
   import ro.sync.ecss.extensions.api.AuthorTableColumnWidthProvider;
   import ro.sync.ecss.extensions.api.node.AuthorElement;

   public class TableColumnWidthProvider
            implements AuthorTableColumnWidthProvider {
   
   2. Method init is taking as argument an ro.sync.ecss.extensions.api.node.AuthorElement that represents the XML <table> element. In our case the column widths are specified in <col> elements from the <table> element. In such cases you must collect the span information by analyzing the <table> element.

      ```java
      public void init(AuthorElement tableElement) {
          this.tableElement = tableElement;
          AuthorElement[] colChildren = tableElement.getElementsByClassName("customcol");
          if (colChildren != null && colChildren.length > 0) {
              for (int i = 0; i < colChildren.length; i++) {
                  AuthorElement colChild = colChildren[i];
                  if (i == 0) {
                      colsStartOffset = colChild.getStartOffset();
                  }
                  if (i == colChildren.length - 1) {
                      colsEndOffset = colChild.getEndOffset();
                  }
              }
              // Determine the 'width' for this col.
              AttrValue colWidthAttribute = colChild.getAttribute("width");
              String colWidth = null;
          }
      ```
if (colWidthAttribute != null) {
    colWidth = colWidthAttribute.getValue();
    // Add WidthRepresentation objects for the columns this 'customcol'
    // specification spans over.
    colWidthSpecs.add(new WidthRepresentation(colWidth, true));
}
}
}

3. The method **isTableAcceptingWidth** should check if the table cells are a `<td>` element.

```java
public boolean isTableAcceptingWidth(String tableCellsTagName) {
    return "td".equals(tableCellsTagName);
}
```

4. The method **isTableAndColumnsResizable** should check if the table cells are a `<td>` element. This method determines if the table and its columns can be resized by dragging the edge of a column.

```java
public boolean isTableAndColumnsResizable(String tableCellsTagName) {
    return "td".equals(tableCellsTagName);
}
```

5. Methods **getTableWidth** and **getCellWidth** are used to determine the table and column width. The table layout engine will ask this `ro.sync.ecss.extensions.api.AuthorTableColumnWidthProvider` implementation what is the table width for each table element and the cell width for each cell element from the table that was marked as cell in the CSS using the property `display:table-cell`. The implementation is simple and just parses the value of the `width` attribute. The methods must return null for the tables / cells that do not have a specified width.

```java
public WidthRepresentation getTableWidth(String tableCellsTagName) {
    WidthRepresentation toReturn = null;
    if (tableElement != null && "td".equals(tableCellsTagName)) {
        AttrValue widthAttr = tableElement.getAttribute("width");
        if (widthAttr != null) {
            String width = widthAttr.getValue();
            if (width != null) {
                toReturn = new WidthRepresentation(width, true);
            }
        }
    }
    return toReturn;
}

public List<WidthRepresentation> getCellWidth(AuthorElement cellElement, int colNumberStart, int colSpan) {
    
```
List<WidthRepresentation> toReturn = null;
int size = colWidthSpecs.size();
if (size >= colNumberStart && size >= colNumberStart + colSpan) {
toReturn = new ArrayList<WidthRepresentation>(colSpan);
for (int i = colNumberStart; i < colNumberStart + colSpan; i++) {
    // Add the column widths
    toReturn.add(colWidthSpecs.get(i));
}
}
return toReturn;

6. Methods commitTableWidthModification and commitColumnWidthModifications are used to commit changes made to the width of the table or its columns when using the mouse drag gestures.

public void commitTableWidthModification
(AuthorDocumentController authorDocumentController,
 int newTableWidth, String tableCellsTagName) throws AuthorOperationException {
if ("td".equals(tableCellsTagName)) {
    if (newTableWidth > 0) {
        if (tableElement != null) {
            String newWidth = String.valueOf(newTableWidth);
            authorDocumentController.setAttribute(
                "width",
                new AttrValue(newWidth),
                tableElement);
        } else {
            throw new AuthorOperationException("Cannot find the table element.");
        }
    }
}
}

public void commitColumnWidthModifications
(AuthorDocumentController authorDocumentController,
 WidthRepresentation[] colWidths, String tableCellsTagName)
throws AuthorOperationException {
if ("td".equals(tableCellsTagName)) {
    if (colWidths != null && tableElement != null) {
        if (colsStartOffset >= 0 && colsEndOffset >= 0 && colsStartOffset < colsEndOffset) {
            authorDocumentController.delete(colsStartOffset, colsEndOffset);
        }
    }
}
String xmlFragment = createXMLFragment(colWidths);
    int offset = -1;
    AuthorElement[] header = tableElement.getElementsByLocalName("header");
    if (header != null && header.length > 0) {
        // Insert the cols elements before the 'header' element
        offset = header[0].getStartOffset();
    }
    if (offset == -1) {
        throw new AuthorOperationException("No valid offset to insert column width");
    }
    authorDocumentController.insertXMLFragment(xmlFragment, offset);
}

private String createXMLFragment(WidthRepresentation[] widthRepresentations) {
    StringBuffer fragment = new StringBuffer();
    String ns = tableElement.getNamespace();
    for (int i = 0; i < widthRepresentations.length; i++) {
        WidthRepresentation width = widthRepresentations[i];
        fragment.append("<customcol");
        String strRepresentation = width.getWidthRepresentation();
        if (strRepresentation != null) {
            fragment.append(" width=" + width.getWidthRepresentation() + ":");
        }
        if (ns != null && ns.length() > 0) {
            fragment.append(" xmlns="" + ns + ":");
        }
        fragment.append("/>");
    }
    return fragment.toString();
}

7. The following three methods are used to determine what type of column width specifications the table
column width provider support. In our case all types of specifications are allowed:

public boolean isAcceptingFixedColumnWidths(String tableCellsTagName) {
    return true;
}

public boolean isAcceptingPercentageColumnWidths(String tableCellsTagName) {
    return true;
public boolean isAcceptingProportionalColumnWidths(String tableCellsTagName) {
    return true;
}

Note:
The complete source code for framework customization examples can be found in the oxygen-sample-framework module of the Oxygen SDK, available as a Maven archetype on the Oxygen XML Editor website.

In the listing below, the XML document contains the table element:

```xml
<table width="300">
    <customcol width="50.0px"/>
    <customcol width="1*"/>
    <customcol width="2*"/>
    <customcol width="20%"/>
    <header>
        <td>C1</td>
        <td>C2</td>
        <td>C3</td>
        <td>C4</td>
    </header>
    <tr>
        <td>cs=1, rs=1</td>
        <td>cs=1, rs=1</td>
        <td row_span="2">cs=1, rs=2</td>
        <td row_span="3">cs=1, rs=3</td>
    </tr>
    <tr>
        <td>cs=1, rs=1</td>
        <td>cs=1, rs=1</td>
    </tr>
    <tr>
        <td column_span="3">cs=3, rs=1</td>
    </tr>
</table>
```

When no table column width provider is specified, the table has the following layout:
When the above implementation is configured, the table has the correct layout:

**Note:**
The complete source code for framework customization examples can be found in the oxygen-sample-framework module of the Oxygen SDK, available as a Maven archetype on the Oxygen XML Editor website.

### Configuring a Table Cell Span Provider

In a custom framework (on page 3219), the `<table>` element can have cells that span over multiple columns and rows. As explained in Configuring Tables (on page 2248), you need to indicate Oxygen XML Editor a method to determine the cell spanning. If you use the `@rowspan` and `@colspan` attributes, Oxygen XML Editor can determine the cell spanning automatically. In the following example, the `<td>` element uses the `@row_span` and `@column_span` attributes that are not recognized by default. You will need to implement a Java extension class for defining the cell spanning.
1. Create the class `simple.documentation.framework.TableCellSpanProvider`. This class must implement the `ro.sync.ecss.extensions.api.AuthorTableCellSpanProvider` interface.

   ```java
   import ro.sync.ecss.extensions.api.AuthorTableCellSpanProvider;
   import ro.sync.ecss.extensions.api.node.AttrValue;
   import ro.sync.ecss.extensions.api.node.AuthorElement;

   public class TableCellSpanProvider
       implements AuthorTableCellSpanProvider {
   }
   ```

2. The `init` method takes `ro.sync.ecss.extensions.api.node.AuthorElement` that represents the XML `<table>` element as its argument. In this example, the cell span is specified for each of the cells so you leave this method empty. However, there are cases (such as the CALS table model) when the cell spanning is specified in the `<table>` element. In such cases, you must collect the span information by analyzing the `<table>` element.

   ```java
   public void init(AuthorElement table) {
   }
   ```

3. The `getColSpan` method is taking as argument the table cell. The table layout engine will ask this `AuthorTableSpanSupport` implementation what is the column span and the row span for each XML element from the table that was marked as cell in the CSS using the property `display:table-cell`. The implementation is simple and just parses the value of `column_span` attribute. The method must return null for all the cells that do not change the span specification.

   ```java
   public Integer getColSpan(AuthorElement cell) {
       Integer colSpan = null;

       AttrValue attrValue = cell.getAttribute("column_span");
       if(attrValue != null) {
           // The attribute was found.
           String cs = attrValue.getValue();
           if(cs != null) {
               try {
                   colSpan = new Integer(cs);
               } catch (NumberFormatException ex) {
                   // The attribute value was not a number.
               }
           }
       }
       return colSpan;
   }
   ```

4. The row span is determined in a similar manner:

   ```java
   public Integer getRowSpan(AuthorElement cell) {
       Integer rowSpan = null;
   }
   ```
5. The method `hasColumnSpecifications` always returns `true` considering column specifications always available.

```java
public boolean hasColumnSpecifications(AuthorElement tableElement) {
    return true;
}
```

Note:
The complete source code for `framework` customization examples can be found in the `oxygen-sample-framework` module of the Oxygen SDK, available as a Maven archetype on the Oxygen XML Editor website.

6. In the listing below, the XML document contains the table element:

```xml
<table>
    <header>
        <td>C1</td>
        <td>C2</td>
        <td>C3</td>
        <td>C4</td>
    </header>
    <tr>
        <td cs=1, rs=1</td>
        <td column_span="2" row_span="2">cs=2, rs=2</td>
        <td row_span="3">cs=1, rs=3</td>
    </tr>
    <tr>
        <td cs=1, rs=1</td>
    </tr>
</table>
```
When no table cell span provider is specified, the table has the following layout:

**Figure 591. Table layout when no cell span provider is specified**

![Table layout when no cell span provider is specified](image)

When the above implementation is configured, the table has the correct layout:
Configuring a Table Cell Row and Column Separator Provider

In a custom framework (on page 3219), the `<table>` element has separators between rows. As explained in Configuring Tables (on page 2248), you need to indicate a method to determine the way rows and columns are separated. If you use the `@rowsep` and `@colsep` cell element attributes, or your table is conforming to the CALS table model, Oxygen XML Editor can determine the cell separators. Even if there are no attributes that define the separators, you can still force a separator between rows by implementing a Java extension.

1. Create the class `simple.documentation.framework.TableCellSepProvider`. This class must implement the `ro.sync.ecss.extensions.api.AuthorTableCellSepProvider` interface.

```java
import ro.sync.ecss.extensions.api.AuthorTableCellSepProvider;
import ro.sync.ecss.extensions.api.node.AuthorElement;

public class TableCellSepProvider implements AuthorTableCellSepProvider {

```

2. The `init` method takes the `ro.sync.ecss.extensions.api.node.AuthorElement` interface that represents the XML `<table>` element as its argument. If the separator information is implicit, it does not depend on the current table, so you leave this method empty. However, there are cases (such as the CALS table
model) when the cell separators are specified in the `<table>` element. In such cases, you should initialize your provider based on the given argument.

```java
public void init(AuthorElement table) {
}
```

3. The `getColSep` method takes the table cell as its argument. The table layout engine will ask this `AuthorTableCellSepProvider` implementation if there is a column separator for each XML element from the table that was marked as cell in the CSS using the property `display:table-cell`. The following example returns `false`, meaning there will not be column separators.

```java
/**
   * @return false - No column separator at the right of the cell.
   */
@Override
public boolean getColSep(AuthorElement cellElement, int columnIndex) {
    return false;
}
```

4. The row separators are determined in a similar manner. This time the example returns `true`, forcing a separator between the rows.

```java
/**
   * @return true - A row separator below each cell.
   */
@Override
public boolean getRowSep(AuthorElement cellElement, int columnIndex) {
    return true;
}
```

Note:
The complete source code for framework customization examples can be found in the `oxygen-sample-framework` module of the Oxygen SDK, available as a Maven archetype on the Oxygen XML Editor website.

5. In the example below, the XML document contains the table element:

```xml
<table>
  <header>
    <td>H1</td>
    <td>H2</td>
    <td>H3</td>
    <td>H4</td>
  </header>
  <tr>
    <td>C11</td>
  </tr>
</table>
```
When the borders for the `<td>` element are removed from the CSS, the row separators become visible:

**Figure 593. Row separators provided by the Java implementation.**

```
<table>
<thead>
<tr>
<th>H1</th>
<th>H2</th>
<th>H3</th>
<th>H4</th>
</tr>
</thead>
<tbody>
<tr>
<td>C11</td>
<td>C12</td>
<td>C13</td>
<td>C14</td>
</tr>
<tr>
<td>C21</td>
<td>C22</td>
<td>C23</td>
<td>C24</td>
</tr>
<tr>
<td>C31</td>
<td>C32</td>
<td>C33</td>
<td>C34</td>
</tr>
</tbody>
</table>
```

**Note:**
The complete source code for `framework` customization examples can be found in the `oxygen-sample-framework` module of the Oxygen SDK, available as a Maven archetype on the Oxygen XML Editor website.

**Customizing Attribute Value Editors**

The `CustomAttributeValueEditor` extension point allows you to customize the attribute value editing mechanisms in Oxygen XML Editor. It changes the `Browse` button found in the attribute editors to an `Edit` button. When a user clicks that `Edit` button, your custom attribute value editor will be presented.
The Edit button can be accessed in the following attribute editors:

- The Attributes view in Author mode (on page 627) (when the Expand button is used to reveal an expanded panel).
- The Attributes view in Text mode (on page 542) (when the Expand button is used to reveal an expanded panel).
- The In-place Attributes Editor (on page 630) when invoked in Author mode.
- The In-place Attributes Editor invoked in the Outline view (on page 539).

How to Implement a CustomAttributeValueEditor

To implement your own CustomAttributeValueEditor, follow this procedure:

1. Extend the ro.sync.ecss.extensions.api.CustomAttributeValueEditor abstract class.
2. To instruct Oxygen XML Editor to use this newly created implementation, use either of the following methods:
   a. If you have configured an extensions bundle (on page 2232), you can return the CustomAttributeValueEditor implementation using the ro.sync.ecss.extensions.api.ExtensionsBundle.createCustomAttributeValueEditor() method.
   b. Specify the CustomAttributeValueEditor in the Author custom attribute value editor individual extension in the Extensions tab (on page 167) of the Document Type configuration dialog box (on page 140) for your particular document type.

Example

The following example creates a very simple custom attribute value editor:

```java
/**
 * A custom attribute value editor.
 */

class MyCustomAttributeValueEditor extends CustomAttributeValueEditor {

    /**
     * @see ro.sync.ecss.extensions.api.Extension#getDescription()
     */
    @Override
    public String getDescription() {
        return "My custom attribute value editor";
    }

    /**
     * @see ro.sync.ecss.extensions.api.CustomAttributeValueEditor#getAttributeValue
     * (ro.sync.ecss.extensions.api.EditedAttribute, java.lang.Object)
     */
    ...
public String getAttributeValue(EditedAttribute attribute, Object parentComponent) throws CancelledByUserException {
    // Show an input dialog for collecting the new value
    return JOptionPane.showInputDialog(
        "Set a new value for " + attribute.getAttributeQName() + ":");
}

/**
 * @see ro.sync.ecss.extensions.api.CustomAttributeValueEditor#shouldHandleAttribute
 * (ro.sync.ecss.extensions.api.EditedAttribute)
 */
@Override
public boolean shouldHandleAttribute(EditedAttribute attribute) {
    // Handle all attributes
    return true;
}
}

Example result: If a user were to click the Edit button in any of the attribute editors, the following dialog box would be displayed that allows the user to insert a value for the particular attribute:

![Dialog box](image)

Customizing the CSS Styles Filter

You can modify the CSS styles for each `ro.sync.ecss.extensions.api.node.AuthorNode` rendered in the Author mode using an implementation of `ro.sync.ecss.extensions.api.StylesFilter`. You can implement the various callbacks of the interface either by returning the default value given by Oxygen XML Editor or by contributing to the value. The received styles `ro.sync.ecss.css.Styles` can be processed and values can be overwritten with your own. For example, you can override the `KEY_BACKGROUND_COLOR` style to return your own implementation of `ro.sync.exml.view.graphics.Color` or override the `KEY_FONT` style to return your own implementation of `ro.sync.exml.view.graphics.Font`.

For instance, in this simple document example, the filter can change the value of the `KEY_FONT` property for the `<table>` element:

```java
package simple.documentation.framework;
```
import ro.sync.ecss.css.Styles;
import ro.sync.ecss.extensions.api.StylesFilter;
import ro.sync.ecss.extensions.api.node.AuthorNode;
import ro.sync.exml.view.graphics.Font;

public class SDFStylesFilter implements StylesFilter {
    public Styles filter(Styles styles, AuthorNode authorNode) {
        if (AuthorNode.NODE_TYPE_ELEMENT == authorNode.getType() && "table".equals(authorNode.getName())) {
            styles.setProperty(Styles.KEY_FONT, new Font(null, Font.BOLD, 12));
        }
        return styles;
    }
}

Note:
The complete source code for framework customization examples can be found in the oxygen-sample-framework module of the Oxygen SDK, available as a Maven archetype on the Oxygen XML Editor website.

Customizing Elements that Wrap Profiled Content

For each framework (on page 3219) (document type), you can configure the phrase-type elements that wrap the profiled content by setting a custom ro.sync.ecss.extensions.api.ProfilingConditionalTextProvider. This configuration is set by default for DITA and DocBook frameworks.

Customizing the Link Target Reference Finder

The link target reference finder represents the support for finding references from links that indicate specific elements inside an XML document. This support will only be available if a schema is associated with the document type.

If you do not define a custom link target reference finder, the DefaultElementLocatorProvider implementation (on page 2264) will be used by default. The interface that should be implemented for a custom link target reference finder is ro.sync.ecss.extensions.api.link.ElementLocatorProvider. As an alternative, the ro.sync.ecss.extensions.commons.DefaultElementLocatorProvider implementation can also be extended.

The used ElementLocatorProvider will be queried for an ElementLocator when a link location must be determined (when a link is clicked). Then, to find the corresponding (linked) element, the obtained ElementLocator will be queried for each element from the document.
Creating a Custom Link Target Reference Finder

If you need to create a custom link target reference finder you can do so by creating the class that will implement the `ro.sync.ecss.extensions.api.link.ElementLocatorProvider` interface. As an alternative, your class could extend `ro.sync.ecss.extensions.commons.DefaultElementLocatorProvider`, the default implementation.

**Note:**
The complete source code of the `ro.sync.ecss.extensions.commons.DefaultElementLocatorProvider`, `ro.sync.ecss.extensions.commons.IDElementLocator` or `ro.sync.ecss.extensions.commons.XPointerElementLocator` can be found in the `oxygen-sample-framework` project.

**Default Link Target Reference Finder**

The `DefaultElementLocatorProvider` implementation is used by default to find link target references. It offers support for the most common types of links:

- Links based on ID attribute values *(on page 2265).*
- XPointer element() scheme *(on page 2266).*

The method `getElementLocator` determines what `ElementLocator` should be used. In the default implementation, it checks if the link is an XPointer element() scheme. Otherwise, it assumes it is an ID. A non-null `IDTypeVerifier` will always be provided if a schema is associated with the document type.

The `link` string argument is the anchor *(on page 3216)* part of the of the URL that is composed from the value of the link property specified for the link element in the CSS.

```java
public ElementLocator getElementLocator(IDTypeVerifier idVerifier,
                                        String link) {
    ElementLocator elementLocator = null;
    try {
        if (link.startsWith("element(")) {
            // xpointer element() scheme
            elementLocator = new XPointerElementLocator(idVerifier, link);
        } else {
            // Locate link element by ID
            elementLocator = new IDElementLocator(idVerifier, link);
        }
    }
    return elementLocator;
}
```
catch (ElementLocatorException e) {
    logger.warn("Exception when create element locator for link: " + link + ". Cause: " + e, e);
}

return elementLocator;

ID Element Locator

The IDElementLocator is an implementation of the abstract class ro.sync.ecss.extensions.api.link.ElementLocator for links that use an ID.

The constructor only assigns field values and the method endElement is empty for this implementation.

The method startElement checks each of the element's attribute values and when one matches the link, it considers the element found if one of the following conditions is satisfied:

- The qualified name of the attribute is xml:id.
- The attribute type is ID.

The attribute type is checked with the help of the method IDTypeVerifier.hasIDType.

```java
public boolean startElement(String uri, String localName, String name, Attr[] atts) {
    boolean elementFound = false;
    for (int i = 0; i < atts.length; i++) {
        if (link.equals(atts[i].getValue())) {
            if ("xml:id".equals(atts[i].getQName())) {
                // xml:id attribute
                elementFound = true;
            } else {
                // check if attribute has ID type
                String attrLocalName =
                    ExtensionUtil.getLocalName(atts[i].getQName());
                String attrUri = atts[i].getNamespace();
                if (idVerifier.hasIDType(localName, uri, attrLocalName, attrUri)) {
                    elementFound = true;
                }
            }
        }
    }

    return elementFound;
}
```
XPointer Element Locator

XPointerElementLocator is an implementation of the abstract class ro.sync.ecss.extensions.api.link.ElementLocator for links that have one of the following XPointer element() scheme patterns:

- **element (elementID)**
  
  Locate the element with the specified ID.

- **element (/1/2/3)**
  
  A child sequence appearing alone identifies an element by means of stepwise navigation, which is directed by a sequence of integers separated by slashes (/). Each integer n locates the nth child element of the previously located element.

- **element (elementID/3/4)**
  
  A child sequence appearing after a NCName identifies an element by means of stepwise navigation, starting from the element located by the given name.

The constructor separates the ID/integers, which are delimited by slashes (/) into a sequence of identifiers (an XPointer path). It will also check that the link has one of the supported patterns of the XPointer element() scheme.

```java
public XPointerElementLocator(IDTypeVerifier idVerifier, String link)
        throws ElementLocatorException {
    super(link);
    this.idVerifier = idVerifier;

    link = link.substring("element ", link.length() - 1);

    StringTokenizer stringTokenizer = new StringTokenizer(link, "/", false);
    xpointerPath = new String[stringTokenizer.countTokens()];
    int i = 0;
    while (stringTokenizer.hasMoreTokens()) {
        xpointerPath[i] = stringTokenizer.nextToken();
        boolean invalidFormat = false;

        // Empty xpointer component is not supported
        if (xpointerPath[i].length() == 0) {
            invalidFormat = true;
        }

        if (i > 0) {
            try {
                Integer.parseInt(xpointerPath[i]);
            } catch (NumberFormatException e) {
                invalidFormat = true;
            }
        }
    }
```
catch (NumberFormatException e) {
    invalidFormat = true;
}
}

if(invalidFormat){
    throw new ElementLocatorException(
        "Only the element() scheme is supported when locating XPointer links."
    + "Supported formats: element(elementID), element(/1/2/3),
    element(elemID/2/3/4)."));
}

if(Character.isDigit(xpointerPath[0].charAt(0))){
    // This is the case when xpointer have the following pattern /1/5/7
    xpointerPathDepth = xpointerPath.length;
} else {
    // This is the case when xpointer starts with an element ID
    xpointerPathDepth = -1;
    startWithElementID  = true;
}

The method startElement will be invoked at the beginning of every element in the XML document (even when the element is empty). The arguments it takes are

uri

The namespace URI, or the empty string if the element has no namespace URI or if namespace processing is disabled.

localName

Local name of the element.

qName

Qualified name of the element.

atts

Attributes attached to the element. If there are no attributes, this argument will be empty.

The method returns true if the processed element is found to be the one indicated by the link.

The XPointerElementLocator implementation of the startElement will update the depth of the current element and keep the index of the element in its parent. If the xpointerPath starts with an element ID then the current
element ID is verified to match the specified ID. If this is the case the depth of the XPointer is updated taking into account the depth of the current element.

If the XPointer path depth is the same as the current element depth then the kept indices of the current element path are compared to the indices in the XPointer path. If all of them match then the element has been found.

```java
public boolean startElement(String uri, String localName,
                            String name, Attr[] atts) {
    boolean linkLocated = false;
    // Increase current element document depth
    startElementDepth ++;
    
    if (endElementDepth != startElementDepth) {
        // The current element is the first child of the parent
        currentElementIndexStack.push(new Integer(1));
    } else {
        // Another element in the parent element
        currentElementIndexStack.push(new Integer(lastIndexInParent + 1));
    }
    
    if (startWithElementID) {
        // This the case when xpointer path starts with an element ID.
        String xpointerElement = xpointerPath[0];
        for (int i = 0; i < atts.length; i++) {
            if (xpointerElement.equals(atts[i].getValue())){
                if (idVerifier.hasIDType(
                            localName, uri, atts[i].getQName(), atts[i].getNamespace())){
                    xpointerPathDepth = startElementDepth + xpointerPath.length - 1;
                    break;
                }
            }
        }
        if (xpointerPathDepth == startElementDepth){
            // check if xpointer path matches with the current element path
            linkLocated = true;
            try {
                int xpointerIdx = xpointerPath.length - 1;
                int stackIdx = currentElementIndexStack.size() - 1;
                int stopIdx = startWithElementID ? 1 : 0;
                while (xpointerIdx >= stopIdx && stackIdx >= 0) {
```
```java
int xpointerIndex = Integer.parseInt(xpointerPath[xpointerIdx]);
int currentElementIndex =
    ((Integer)currentElementIndexStack.get(stackIdx)).intValue();
if (xpointerIndex != currentElementIndex) {
    linkLocated = false;
    break;
}
xpointerIdx--;
stackIdx--;
}
}
}`

```java
public void endElement(String uri, String localName, String name) {
    endElementDepth = startElementDepth;
    startElementDepth --;
    lastIndexInParent = ((Integer)currentElementIndexStack.pop()).intValue();
}
```

The method `endElement` will be invoked at the end of every element in the XML document (even when the element is empty).

The `XPointerElementLocator` implementation of the `endElement` updates the depth of the current element path and the index of the element in its parent.

Customizing XML Node Rendering

You can use this API extension to customize the way an XML node is rendered in the Outline view (on page 539) in Author mode, breadcrumb navigation bar (on page 602) in Author mode, Outline view (on page 539) in Text mode, Content Completion Assistant (on page 3217) window, or DITA Maps Manager view (on page 2897).

Note:

There are two methods to provide an implementation of
`ro.sync.exml.workspace.api.node.customizer.XMLNodeRendererCustomizer`:

- As a part of a bundle, returning it from the `createXMLNodeCustomizer()` method of the
  `ExtensionsBundle` associated with your document type in the `Document type` configuration dialog box
  (on page 140) (`Extensions bundle` field in the `Extensions` tab).
- As an individual extension, associated with your document type in the `Document type` configuration
dialog box (on page 140) (`XML node renderer customizer` field in the `Individual extensions` section of
the `Extensions` tab).

### Support for Retina/HiDPI Displays

To support Retina or HiDPI displays, the icons provided by the `XMLNodeRendererCustomizer` should be
backed up by a copy of larger size using the proper Retina/HiDPI naming convention (on page 724).

For example, for the `<title>` element, if the `XMLNodeRendererCustomizer` returns the path
`${framework}/images/myImg.png`, then to support Retina images with a scaling factor of 2, an
extra file (`myImg@2x.png`) should be added to the same images directory (`${framework}/images/
myImg@2x.png`). If the higher resolution icon (the `@2x` file) does not exist, the `normal` icon is scaled and used
instead.

For more information about using Retina/HiDPI images, refer to the Using Retina/HiDPI Images in Author
Mode (on page 724) section.

**Note:**
The complete source code for `framework` customization examples can be found in the `oxygen-
sample-framework` module of the Oxygen SDK, available as a Maven archetype on the Oxygen XML
Editor website.

**Related Information:**
- Customizing the Rendering of Elements (on page 2207)
- Using Retina/HiDPI Icons for the Actions from a Framework (on page 2181)

### Customizing Author Operations

Oxygen XML Editor **Author** mode has a built-in set of operations covering the insertion of text and XML
fragments (see the Author Default Operations (on page 2152)) and the execution of XPath expressions on the
current document edited in **Author** mode. However, there are situations where you need to extend this set. The
following examples are just a few of the possible situations:
• You need to enter an element whose attributes will be edited by the user through a graphical user interface.
• The user must send selected element content (or the whole document) to a server for some kind of processing.
• Content authors need to extract pieces of information from a server and insert it directly into the edited XML document.
• You need to apply an XPath expression on the current document and process the nodes of the resulting node set.

To extend the Oxygen XML Editor Author mode functionality through Java, you will need the Oxygen SDK available on the Oxygen XML Editor website. It includes the source code of the Author mode operations in the built-in document types and the full documentation (in Javadoc format) of the public API available for Author mode custom actions.

The subsequent Java examples make use of AWT classes. If you are developing extensions for the Oxygen XML Editor XML Editor plugin for Eclipse, you will have to use their SWT counterparts.

⚠️ Attention:
Make sure the Java classes of your custom Author mode operations are compiled with the same Java version used by Oxygen XML Editor. Otherwise, the classes may not be loaded by the Java virtual machine. For example, if you run Oxygen XML Editor with a Java 1.8 virtual machine but the Java classes of your custom Author mode operations are compiled with a Java 9 or later virtual machine then the custom operations cannot be loaded and used by the Java 1.8 virtual machine.

⚠️ Important:
From a legal standpoint, you can freely develop and share extensions using the Oxygen SDK ONLY if you have a legal, active license to use Oxygen XML Editor and ONLY if such extensions are used from inside Oxygen XML Editor. To use such extensions outside of Oxygen XML Editor (for example, a 3rd-party application that has Oxygen XML Editor built in to it), an additional license must be purchased to use the SDK according the Oxygen XML SDK Licensing Policy.

Related Information:
- Extending Oxygen With the SDK (on page 2411)

Example 1 - Simple Use of a Dialog Box from an Author Mode Operation

In this example, functionality is added for inserting images in a custom framework (on page 3219). The images are represented by the `<image>` element. The location of the image file is represented by the value of the `@href` attribute. In the Java implementation, a dialog box will be displayed with a text field where the user can enter a full URL or browse for a local file.
1. Set up a sample project following this set of instructions. The framework project is `oxygen-sample-framework`.

2. Modify the `simple.documentation.framework.InsertImageOperation` class that implements the `ro.sync.ecss.extensions.api.AuthorOperation` interface. This interface defines three methods: `doOperation`, `getArguments` and `getDescription`.

A short description of these methods follows:

- The `doOperation` method is invoked when the action is performed either by pressing the toolbar button, by selecting the menu item or by pressing the shortcut key. The arguments taken by this method can be one of the following combinations:
  - An object of type `ro.sync.ecss.extensions.api.AuthorAccess` and a map.
  - Argument names and values.
- The `getArguments` method is used by Oxygen XML Editor when the action is configured. It returns the list of arguments (name and type) that are accepted by the operation.
- The `getDescription` method is used by Oxygen XML Editor when the operation is configured. It returns a description of the operation.

**Example:**

Here is the implementation of these three methods:

```java
/**
 * Performs the operation.
 */
public void doOperation(
    AuthorAccess authorAccess,
    ArgumentsMap arguments
) throws IllegalArgumentException,
    AuthorOperationException {

    JFrame oxygenFrame = (JFrame) authorAccess.getWorkspaceAccess().getParentFrame();

    String href = displayURLDialog(oxygenFrame);
    if (href.length() != 0) {
        // Creates the image XML fragment.
        String imageFragment =
            "<image xmlns='http://www.oxygenxml.com/sample/documentation' href='" + href + "]/";

        // Inserts this fragment at the cursor position.
        int caretPosition = authorAccess.getEditorAccess().getCaretOffset();
        authorAccess.getDocumentController().insertXMLFragment(imageFragment, caretPosition);
    }
}
```
/**
 * Has no arguments.
 * @return null.
 */
public ArgumentDescriptor[] getArguments() {
    return null;
}

/**
 * @return A description of the operation.
 */
public String getDescription() {
    return "Inserts an image element. Asks the user for a URL reference.";
}

Note:
The complete source code for framework customization examples can be found in the oxygen-sample-framework module of the Oxygen SDK, available as a Maven archetype on the Oxygen XML Editor website.

Important:
Make sure you always specify the namespace of the inserted fragments.

<image xmlns='http://www.oxygenxml.com/sample/documentation'
href='path/to/image.png'/>

3. Package the compiled class into a JAR (on page 3220) file. An example of an Ant script that packages the classes folder content into a JAR archive named sdf.jar is listed below:

<?xml version="1.0" encoding="UTF-8"?>
<project name="project" default="dist">
    <target name="dist">
        <jar destfile="sdf.jar" basedir="classes">
            <fileset dir="classes">
                <include name="**/*"/>
            </fileset>
        </jar>
    </target>
</project>
4. Copy the sdf.jar file into your custom framework directory
   ({OXYGEN_INSTALL_DIR}/frameworks/[CUSTOM_FRAMEWORK_DIR]).

5. Add the sdf.jar to the class path. To do this, open the Preferences dialog box (Options > Preferences) (on page 124), go to Document Type Association, select SDF, and click the Edit button.

6. Select the Classpath tab in the lower part of the Document Type configuration dialog box (on page 140) and click the Add button. In the displayed dialog box, enter the location of the JAR file, relative to the Oxygen XML Editor frameworks folder.

7. Next, create the action that will use the defined operation. Go to the Actions subtab. Copy the icon files for the menu item and for the toolbar in your custom framework directory ({OXYGEN_INSTALL_DIR}/frameworks/[CUSTOM_FRAMEWORK_DIR]).

8. Define the action's properties:
   ◦ Set ID to insert_image.
   ◦ Set Name to Insert image.
   ◦ Set Menu access key to letter i.
   ◦ Set Toolbar action to ${framework}/toolbarImage.png.
   ◦ Set Menu icon to ${framework}/menuImage.png.
   ◦ Set Shortcut key to Ctrl (Meta on macOS)+Shift+i.

9. Next, set up the operation. You want to add images only if the current element is a <section>, <book> or <article>.
   ◦ Set the value of XPath expression to
     ```
     local-name()='section' or local-name()='book'
     or local-name()='article'
     ```
   ◦ Set the Invoke operation field to simple.documentation.framework.InsertImageOperation.
To test the action, open or create an XML file and place the cursor at a valid location. Then click the button associated with the action from the toolbar. In the dialog box, select an image URL and click OK. The image is inserted into the document.

Example 2 - Operations with Arguments - Report from Database Operation

In this example, an operation is created that connects to a relational database and executes an SQL statement. The result should be inserted in the edited XML document as a table. To make the operation fully configurable, it will have arguments for the database connection string, the user name, the password and the SQL expression.

1. Set up a sample project following these instructions. The framework project is oxygen-sample-framework.
2. Create the class simple.documentation.framework.QueryDatabaseOperation. This class must implement the ro.sync.ecss.extensions.api.AuthorOperation interface.
import ro.sync.ecss.extensions.api.ArgumentDescriptor;
import ro.sync.ecss.extensions.api.ArgumentsMap;
import ro.sync.ecss.extensions.api.AuthorAccess;
import ro.sync.ecss.extensions.api.AuthorOperation;
import ro.sync.ecss.extensions.api.AuthorOperationException;

public class QueryDatabaseOperation implements AuthorOperation {

private static final String ARG_JDBC_DRIVER = "jdbc_driver";
private static final String ARG_USER = "user";
private static final String ARG_PASSWORD = "password";
private static final String ARG_SQL = "sql";
private static final String ARG_CONNECTION = "connection";

4. You must describe the argument name and type. To do this, implement the `getArguments` method that will return an array of argument descriptors:

    public ArgumentDescriptor[] getArguments() {
        ArgumentDescriptor args[] = new ArgumentDescriptor[] {
            new ArgumentDescriptor(
                ARG_JDBC_DRIVER,
                ArgumentDescriptor.TYPE_STRING,
                "The name of the Java class that is the JDBC driver."),
            new ArgumentDescriptor(
                ARG_CONNECTION,
                ArgumentDescriptor.TYPE_STRING,
                "The database URL connection string."),
            new ArgumentDescriptor(
                ARG_USER,
                ArgumentDescriptor.TYPE_STRING,
                "The name of the database user."),
            new ArgumentDescriptor(
                ARG_PASSWORD,
                ArgumentDescriptor.TYPE_STRING,
                "The database password."),
            new ArgumentDescriptor(
                ARG_SQL,
                ArgumentDescriptor.TYPE_STRING,
                "The SQL statement to be executed."))
        };
    }
These names, types and descriptions will be listed in the **Arguments** table when the operation is configured.

5. When the operation is invoked, the implementation of the `doOperation` method extracts the arguments, forwards them to the method that connects to the database and generates the XML fragment. The XML fragment is then inserted at the cursor position.

```java
public void doOperation(AuthorAccess authorAccess, ArgumentsMap map) throws IllegalArgumentException, AuthorOperationException {
    // Collects the arguments.
    String jdbcDriver = (String)map.getArgumentValue(ARG_JDBC_DRIVER);
    String connection = (String)map.getArgumentValue(ARG_CONNECTION);
    String user = (String)map.getArgumentValue(ARG_USER);
    String password = (String)map.getArgumentValue(ARG_PASSWORD);
    String sql = (String)map.getArgumentValue(ARG_SQL);

    int caretPosition = authorAccess.getCaretOffset();

    try {
        authorAccess.getDocumentController().insertXMLFragment(
            getFragment(jdbcDriver, connection, user, password, sql),
            caretPosition);
    } catch (SQLException e) {
        throw new AuthorOperationException(
            "The operation failed due to the following database error: ", e);
    } catch (ClassNotFoundException e) {
        throw new AuthorOperationException(
            "The JDBC database driver was not found. Tried to load '", jdbcDriver + "',
            e);
    }
}
```

6. The `getFragment` method loads the JDBC driver, connects to the database and extracts the data. The result is a `<table>` element from the [http://www.oxygenxml.com/sample/documentation](http://www.oxygenxml.com/sample/documentation) namespace. The `<header>` element contains the names of the SQL columns. All the text from the XML fragment is
escaped. This means that the '<' and '&' characters are replaced with the '&lt;' and '&amp;' character entities to ensure that the fragment is well-formed.

```java
private String getFragment(
    String jdbcDriver,
    String connectionURL,
    String user,
    String password,
    String sql) throws
    SQLException,
    ClassNotFoundException {

    Properties pr = new Properties();
    pr.put("characterEncoding", "UTF8");
    pr.put("useUnicode", "TRUE");
    pr.put("user", user);
    pr.put("password", password);

    // Loads the database driver.
    Class.forName(jdbcDriver);

    // Opens the connection
    Connection connection =
        DriverManager.getConnection(connectionURL, pr);
    java.sql.Statement statement =
        connection.createStatement();
    ResultSet resultSet =
        statement.executeQuery(sql);

    StringBuffer fragmentBuffer = new StringBuffer();
    fragmentBuffer.append(
        "<table xmlns="
            +
        
            "http://www.oxygenxml.com/sample/documentation'">
    //
    // Creates the table header.
    //
    fragmentBuffer.append("<header>");
    ResultSetMetaData metaData = resultSet.getMetaData();
    int columnCount = metaData.getColumnCount();
    for (int i = 1; i <= columnCount; i++) {
        fragmentBuffer.append("<td>");
        fragmentBuffer.append(xmlEscape(metaData.getColumnName(i)));
```
fragmentBuffer.append("</td>");
}
fragmentBuffer.append("</header>");

//
// Creates the table content.
//
while (resultSet.next()) {
    fragmentBuffer.append("<tr>");
    for (int i = 1; i <= columnCount; i++) {
        fragmentBuffer.append("<td>");
        fragmentBuffer.append(xmlEscape(resultSet.getObject(i)));
        fragmentBuffer.append("</td>");
    }
    fragmentBuffer.append("</tr>");
}
fragmentBuffer.append("</table>");

// Cleanup
resultSet.close();
statement.close();
connection.close();
return fragmentBuffer.toString();

Note:
The complete source code for framework customization examples can be found in the oxygen-sample-framework module of the Oxygen SDK, available as a Maven archetype on the Oxygen XML Editor website.

7. Package the compiled class into a JAR (on page 3220) file.
8. Copy the JAR file and the JDBC driver files into your custom framework directory (/OXYGEN_INSTALL_DIR\frameworks\[CUSTOM_FRAMEWORK_DIR]).
9. Add the JARs to the class path. To do this, open the Document Type Association preferences page (on page 138), select SDF and click the Edit button. Select the Classpath tab in the lower part of the Document Type configuration dialog box (on page 140) and click the Add button. In the displayed dialog box, enter the location of the JAR file, relative to the Oxygen XML Editor frameworks folder.
10. Go to the Actions subtab. The action properties are:
    ◦ Set ID to clients_report.
    ◦ Set Name to Clients Report.
10. The action will work only if the current element is a section. Set up the operation as follows:

- Set **XPath expression** to:
  
  ```
  local-name() = 'section'
  ```

- Use the Java operation defined earlier to set the **Invoke operation** field. Click the **Choose** button, then select `simple.documentation.framework.QueryDatabaseOperation`. Once selected, the list of arguments is displayed. In the figure below the first argument, `jdbc_driver`, represents the class name of the MySQL JDBC driver. The connection string has the URL syntax: `jdbc://<database_host>:<database_port>/<database_name>`.

The SQL expression used in the example follows, but it can be any valid SELECT expression that can be applied to the database:

```
SELECT userID, email FROM users
```

11. Add the action to the toolbar, using the **Toolbar** panel.

**Figure 595. Java Operation Arguments Setup**

![Java Operation Arguments Setup](image)

To test the action, open or create an XML file and place the cursor at a valid location. Then click the **Create Report** button from the toolbar. You can see below the toolbar with the action button and sample table inserted by the **Clients Report** action.
Handling Author Mode Action Events

The `AuthorActionEventHandler` extension point allows you to handle certain Author mode actions in a special way. For example, a specific use-case would be if you want to insert new lines when you press `Enter` instead of it opening the `Content Completion Assistant` (on page 3217).

How to Implement an `AuthorActionEventHandler`

To implement your own `AuthorActionEventHandler`, follow this procedure:

1. Implement the `ro.sync.ecss.extensions.api.AuthorActionEventHandler` interface.
2. To instruct Oxygen XML Editor to use this newly created implementation, use either of the following methods:
   a. If you have configured an extensions bundle (on page 2232), you can return the `AuthorActionEventHandler` implementation using the `ro.sync.ecss.extensions.api.ExtensionsBundle.getAuthorActionEventHandler()` method.
   b. Specify the `AuthorActionEventHandler` in the `Author action event handler` individual extension in the `Extensions` tab (on page 167) of the `Document Type` configuration dialog box (on page 140) for your particular document type.

Example

The following example illustrates the use-case mentioned in the introduction, that is an implementation for inserting a new line when the user presses `Enter` in Author mode. It uses the `canHandleEvent` method to make sure the insertion will be performed in an element that will preserve the `new-line` character. Then the `handleEvent` method inserts the new line at the current cursor position.
public class CustomAuthorActionEventHandler implements AuthorActionEventHandler {

    /**
     * @see ro.sync.ecss.extensions.api.AuthorActionEventHandler#canHandleEvent
     * (AuthorAccess, AuthorActionEventType)
     */
    @Override
    public boolean canHandleEvent(AuthorAccess authorAccess, AuthorActionEventType type) {
        boolean canHandle = false;
        if (type == AuthorActionEventType.ENTER) {
            AuthorDocumentController documentController = authorAccess.getDocumentController();
            int caretOffset = authorAccess.getEditorAccess().getCaretOffset();
            try {
                AuthorNode nodeAtOffset = documentController.getNodeAtOffset(caretOffset);
                if (nodeAtOffset instanceof AuthorElement) {
                    AuthorElement elementAtOffset = (AuthorElement) nodeAtOffset;
                    AttrValue xmlSpace = elementAtOffset.getAttribute("xml:space");
                    if (xmlSpace != null && xmlSpace.getValue().equals("preserve")) {
                        canHandle = true;
                    }
                }
            } catch (BadLocationException ex) {
                if (logger.isDebugEnabled()) {
                    logger.error(ex.getMessage(), ex);
                }
            }
        }
        return canHandle;
    }

    /**
     * @see ro.sync.ecss.extensions.api.AuthorActionEventHandler#handleEvent
     * (ro.sync.ecss.extensions.api.AuthorAccess, ro.sync.ecss.extensions.api.AuthorActionEventHandler.AuthorActionEventType)
     */
    @Override
    public boolean handleEvent(AuthorAccess authorAccess,
AuthorActionEventType eventType) {
    int caretOffset = authorAccess.getEditorAccess().getCaretOffset();
    // Insert a new line
    authorAccess.getDocumentController().insertText(caretOffset, "\n");
    return true;
}

/**
 * @see ro.sync.ecss.extensions.api.Extension#getDescription()
 */
@Override
public String getDescription() {
    return "Insert a new line";
}
}

Handling Schema-Aware Editing Events

The AuthorSchemaAwareEditingHandlerAdapter extension point allows you to handle certain Author mode actions in various ways. For example, implementing the AuthorSchemaAwareEditingHandlerAdapter makes it possible to handle events such as typing, the keyboard delete event at a given offset (using Delete or Backspace keys), delete element tags, delete selection, join elements, or paste fragment. It also makes it possible to improve solutions that are proposed by the paste mechanism in Oxygen XML Editor when pasting content (through the use of some specific methods (on page 2284)).

How to Implement an AuthorSchemaAwareEditingHandlerAdapter

For this handler to be called, the Schema-Aware Editing option (on page 181) must be set to On or Custom in the Schema-Aware preferences page (on page 180). The handler can either resolve a specific case, let the default implementation take place, or reject the edit entirely by throwing an InvalidEditException.

To implement your own AuthorSchemaAwareEditingHandlerAdapter, follow this procedure:

1. Implement the ro.sync.ecss.extensions.api.AuthorSchemaAwareEditingHandlerAdapter extension.
2. To instruct Oxygen XML Editor to use this newly created implementation, configure an extensions bundle (on page 2232) and return the AuthorSchemaAwareEditingHandlerAdapter implementation using the ro.sync.ecss.extensions.api.ExtensionsBundle.getAuthorSchemaAwareEditingHandlerAdapter() method.

Example

Typing events can be handled using the handleTyping method. For example, the AuthorSchemaAwareEditingHandler checks if the schema is not a learned one, was loaded successfully, and if
the **Smart paste and drag and drop** option *(on page 181)* is selected. If these conditions are met, the event will be handled.

```java
public class AuthorSchemaAwareEditingHandlerAdapter extends AuthorSchemaAwareEditingHandler {

    /**
     * @see AuthorSchemaAwareEditingHandler#handleTyping
     * (int, char, ro.sync.ecss.extensions.api.AuthorAccess)
     */

    public boolean handleTyping(int offset, char ch, AuthorAccess authorAccess) throws InvalidEditException {
        boolean handleTyping = false;
        AuthorSchemaManager authorSchemaManager = authorAccess.getDocumentController().getAuthorSchemaManager();
        if (!authorSchemaManager.isLearnSchema() && !authorSchemaManager.hasLoadingErrors() && authorSchemaManager.getAuthorSchemaAwareOptions().isEnableSmartTyping()) {
            try {
                AuthorDocumentFragment characterFragment = authorAccess.getDocumentController().createNewDocumentTextFragment(String.valueOf(ch));
                handleTyping = handleInsertionEvent(offset, new AuthorDocumentFragment[] {characterFragment}, authorAccess);
            }
            catch (AuthorOperationException e) {
                throw new InvalidEditException(e.getMessage(), "Invalid typing event: " + e.getMessage(), e, false);
            }
        }
        return handleTyping;
    }
}
```

**Note:**

The complete source code for framework customization examples can be found in the oxygen-sample-framework module of the Oxygen SDK, available as a Maven archetype on the Oxygen XML Editor website.

### Methods for Improving the Paste Mechanism

**getAncestorDetectionOptions**

When pasting content in Author mode, if the result causes the document to become invalid, Oxygen XML Editor will propose solutions to make it valid. As a possible solution, Oxygen XML Editor might surround the pasted content in a sequence of ancestor elements. This
**getAncestorDetectionOptions** method allows you to choose which parent elements might be a possible solution.

**canBeReplaced**

Allows you to improve solutions that might be proposed by the paste mechanism when pasting content in Oxygen XML Editor. For example, when pasting an element inside an empty element with the same name, this **canBeReplaced** method allows Oxygen XML Editor to replace the empty node rather than pasting it after or before the empty node. The callback could also reject this behavior if, for instance, the replacement node contains attributes.

Related Information:

*AuthorDocumentFragment Class*

**Handling When URLs or XHTML Fragments are Dropped or Pasted in Author Mode**

The **AuthorExternalObjectInsertionHandler** extension can be used to configure how URLs or XHTML fragments from external applications are handled when they are dropped or pasted in Author mode.

**How to Implement an AuthorExternalObjectInsertionHandler**

To implement your own **AuthorExternalObjectInsertionHandler**, follow this procedure:

1. Implement the **ro.sync.ecss.extensions.api.AuthorExternalObjectInsertionHandler** interface.
2. To instruct Oxygen XML Editor to use this newly created implementation, use either of the following methods:
   a. If your framework is an extension of DITA, DocBook, TEI, or XHTML, you can specify the **AuthorExternalObjectInsertionHandler** in the **Author extern object Insertion handler** individual extension in the **Extensions** tab (on page 167) of the **Document Type** configuration dialog box (on page 140) for your particular document type.
   b. Otherwise, you can configure an extensions bundle (on page 2232), then return the **AuthorExternalObjectInsertionHandler** implementation using the **ro.sync.ecss.extensions.api.ExtensionsBundle.createAuthorExternalObjectInsertionHandler()** method.
3. You can use a stylesheet to convert the pasted XHTML to your own XML vocabulary by overwriting the following method:

   **ro.sync.ecss.extensions.api.AuthorExternalObjectInsertionHandler.getImporterStylesheetFileName(AuthorAccess)**

   and return the file name of the stylesheet that will be applied. The path to the importer stylesheet must also be added in the **Classpath** tab (on page 144) in the **Document Type** configuration dialog box (on page 140) for your particular document type.
Example

The following example illustrates an implementation for the DITA framework:

```java
/**
 * @see ro.sync.ecss.extensions.api.ExtensionsBundle#createExternalObjectInsertionHandler()
 */
@override
public AuthorExternalObjectInsertionHandler createExternalObjectInsertionHandler() {
    return new DITAExternalObjectInsertionHandler();
}

/**
 * @see ro.sync.ecss.extensions.api.AuthorExternalObjectInsertionHandler#getImporterStylesheetFileName(ro.sync.ecss.extensions.api.AuthorAccess)
 */
@override
protected String getImporterStylesheetFileName(AuthorAccess authorAccess) {
    return "xhtml2ditaDriver.xsl";
}
```

Tip:
For XHTML fragments, there is another method that you could use to configure how they are handled when they are pasted in Author mode. For more information, see Customizing Smart Paste Support (on page 2188).

Presenting an Edit Properties Dialog Box for Actions in Author Mode

The `EditPropertiesHandler` extension point allows you to present a specialized dialog box when the action of double-clicking an element tag is intercepted in Author mode. For example, you could use it to present a dialog box that allows the user to editing the properties of an image.

How to Implement an `EditPropertiesHandler`

To implement your own `EditPropertiesHandler`, follow this procedure:

1. Implement the `ro.sync.ecss.extensions.api.EditPropertiesHandler` interface.
2. To instruct Oxygen XML Editor to use this newly created implementation, use either of the following methods:
a. If you have configured an extensions bundle (on page 2232), you can return the EditPropertiesHandler implementation using the ro.sync.ecss.extensions.api.ExtensionsBundle.createEditPropertiesHandler() method.

b. Specify the EditPropertiesHandler in the Author edit properties handler individual extension in the Extensions tab (on page 167) of the Document Type configuration dialog box (on page 140) for your particular document type.

Example

The following example illustrates an implementation for presenting a simple properties editing dialog box when a user double-clicks an `<image>` tag in Author mode (with tags displayed from the Tags display mode drop-down menu):

```java
public class CustomEditPropertiesHandler implements EditPropertiesHandler {

    /**
     * @see ro.sync.ecss.extensions.api.Extension#getDescription()
     */
    @Override
    public String getDescription() {
        return "Sample implementation that handles properties for a table element.";
    }

    /**
     * @see ro.sync.ecss.extensions.api.EditPropertiesHandler#canEditProperties
     * (ro.sync.ecss.extensions.api.node.AuthorNode)
     */
    @Override
    public boolean canEditProperties(AuthorNode authorNode) {
        return "image".equals(authorNode.getDisplayName());
    }

    /**
     * @see ro.sync.ecss.extensions.api.EditPropertiesHandler#editProperties
     * (ro.sync.ecss.extensions.api.node.AuthorNode,
     * ro.sync.ecss.extensions.api.AuthorAccess)
     */
    @Override
    public void editProperties(AuthorNode authorNode, AuthorAccess authorAccess) {
        AuthorElement imageElement = (AuthorElement) authorNode;
        String currentValue = "";
        AttrValue altValue = imageElement.getAttribute("alt");
```
if (altValue != null) {
    currentValue = altValue.getValue();
}
String newValue = JOptionPane.showInputDialog(
    (Component) authorAccess.getWorkspaceAccess().getParentFrame(),
    "Alternate text",
    currentValue);

if (newValue != null) {
    authorAccess.getDocumentController().setAttribute
    ("alt", new AttrValue(newValue), imageElement);
}
}

Example result: If a user were to double-click an `<image>` tag icon in **Author** mode, the following dialog box would be displayed that allows the user to edit the *alternate text* property for the image:

![Dialog box](image)

**Sharing a Framework**

You can **create a custom framework** by extending a built-in document type *(on page 2133)* (such as DITA or DocBook) using the **Document Type Association preferences page** *(on page 138)*, make modifications to it, and then share the extension with your team.

**Sharing the Extended Framework**

There are several ways that you can share the **extended custom framework** *(on page 2133)* with others:

- Distribute the extended framework along with a project by following these steps:
  1. In a location where you have full write access, create a folder for your project.
  2. Go to the **Project view** *(on page 402)* and create a project. Save it in the folder you created in step 1.
  3. Create a custom framework by extending an existing one *(on page 2133)*, if you have not already done so, and copy the custom framework directory to the folder you created in step 1. Make sure your custom framework directory includes any resources that are referenced in your framework (CSS files, new document templates, schemas used for validation, catalogs, etc.).
4. Go to Options > Preferences > Document Type Association > Locations (on page 139) and select Project Options (on page 3222) at the bottom of the page.

5. In the Additional frameworks directories list, add an entry using the ${pd} editor variable (on page 331) like this: ${pd}/custom_frameworks.

6. You can then share the new project directory with other users. For example, you can commit it to your version control system and have them update their working copy. When they open the customized project file in their Project view (on page 402), the new framework becomes available in the list of document types.

- Pack and deploy the extended framework as an add-on (on page 2289).
- Distribute the directory of the extended framework (on page 2133) to the other members of your team. They will simply copy that directory to their /frameworks directory. The new framework will be available in their list of document types when Oxygen XML Editor starts.

To test the extended framework, the other members of your team can check the list of document types in the Document Type Association preferences page (on page 138) to see if the framework is present and if it appears before the built-in frameworks (meaning that it has higher priority).

Packing and Deploying Frameworks as Add-ons

In Oxygen XML Editor, custom framework (on page 3219) can be packed and deployed as an add-on.

Packing a Framework as an Add-on

This procedure is suitable for developers who want a better control over the add-on (on page 3222) package or those who want to automate some of the steps:

1. Pack the full custom framework (on page 2133) (or an extension of a framework) as a ZIP file or a Java Archive (on page 3220). Note that you should pack the entire root directory not just its contents.

2. [Optional] If you created a Java Archive at the previous step, digitally sign the package. You will need a certificate signed by a trusted authority. To sign the JAR, you can either use the jarsigner command-line tool inside Oracle's Java Development Kit ([JDK_DIR]/bin/jarsigner.exe) or if you are working with Apache Ant (on page 3216), you can use the signjar task (a front for the jarsigner command-line tool). The benefit of having a signed add-on is that you can verify the integrity of the add-on issuer. If you do not have such a certificate, you can generate one yourself using the keytool command-line utility.

   Note:
   This approach is recommended for tests since anyone can create a self-signed certificate.

3. Create a descriptor file. You can use a template that Oxygen XML Editor provides by going to File > New and selecting the Oxygen add-ons update site template. The products the add-on is compatible with can be specified in the template. Once deployed, this descriptor file is referenced as update site.
Alternate Packing Method: Add-ons Packager

Alternatively, you can use the Add-ons Packager plugin by following this procedure:

1. Go to Help > Install new add-ons to open an add-on selection dialog box.
2. Enter or paste https://www.oxygenxml.com/InstData/Addons/optional/updateSite.xml in the Show add-ons from field.
3. Select the Add-ons Packager add-on and click Next.
4. Select the I accept all terms of the end user license agreement option and click Finish.
5. Restart the application.

Result: If the add-on is correctly installed, the Add-ons packager toolbar action is available.

6. Invoke the Add-ons packager toolbar action and input the required information in the displayed dialog box.
7. Click OK to complete the packaging process.

Deploying an Add-on

To deploy an add-on, copy the ZIP or Java Archive (on page 3220) file and the descriptor file to an HTTP server. The URL to this location serves as the Update Site URL.

Related Information:
Oxygen XML Add-on Repositories

Basic Framework Customization Tutorial

This section contains topics meant to provide a general tutorial for customizing a framework (on page 3219). It includes information about creating a basic document type association, some basic customizations, testing the configuration, packaging and deploying the custom framework, and more.

Tip:
A sample framework customization package is available that you can dabble with and use to help you get started. It can be downloaded from: https://www.oxygenxml.com/php/get_oxygen_sample_framework.php. The package includes a sample CSS file, XSL file, schema files, document templates, an XML catalog file, custom icons, and other resources.

Framework Customization Overview

The most important elements of a document type customization are represented by an XML Schema to define the XML structure, the CSS to render the information and the XML instance template that links the first two together.
XML Grammar

To provide as-you-type validation and to compute valid insertion proposals, Oxygen XML Editor needs an XML grammar (XML Schema, DTD, or Relax NG) associated to the XML. The grammar specifies how the internal structure of the XML is defined. For information about associating a schema and how Oxygen XML Editor detects the schema, see Associating a Schema to XML Documents (on page 816).

Consider a use-case where several users are testing a system and must send report results to a content management system. The customization should provide a visual editor for this type of document. The following XML Schema, test_report.xsd defines a report with results of a testing session. The report consists of a title, few lines describing the test suite that was run, and a list of test results (each with a name and a boolean value indicating if the test passed or failed).

```xml
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="report">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="title"/>
        <xs:element ref="description"/>
        <xs:element ref="results"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="title" type="xs:string"/>
  <xs:element name="description">
    <xs:complexType>
      <xs:sequence maxOccurs="unbounded">
        <xs:element name="line">
          <xs:complexType mixed="true">
            <xs:sequence minOccurs="0" maxOccurs="unbounded">
              <xs:element name="important" type="xs:string"/>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="results">
    <xs:complexType>
    </xs:complexType>
  </xs:element>
</xs:schema>
```
### CSS Stylesheet

A set of rules must be defined for describing how the XML document is to be rendered in Author mode. This is done using Cascading Style Sheets (CSS). CSS is a language used to describe how an HTML or XML document should be formatted by a browser. CSS is widely used in the majority of websites.

The elements from an XML document are displayed in the layout as a series of boxes. Some of the boxes contain text and may flow one after the other, from left to right. These are called inline boxes. There are also other types of boxes that flow one below the other (such as paragraphs). These are called block boxes.

For example, consider the way a traditional text editor arranges the text. A paragraph is a block, because it contains a vertical list of lines. The lines are also blocks. However, blocks that contain inline boxes arrange its children in a horizontal flow. That is why the paragraph lines are also blocks, while the traditional "bold" and "italic" sections are represented as inline boxes.

The CSS allows us to specify that some elements are displayed as tables. In CSS, a table is a complex structure and consists of rows and cells. The `table` element must have children that have a `table-row` style. Similarly, the `row` elements must contain elements with a `table-cell` style.

To make it easy to understand, the following section describes how each element from a schema is formatted using a CSS file. Note that this is just one of infinite possibilities for formatting the content.

#### report

The root of a report document. It should be rendered as a box that contains all other elements. To achieve this, the display type is set to `block`. Additionally, some margins are set for it. The CSS rule that matches this element is:

```css
report {
  display: block;
}
```
**title**

The title of the report. Usually titles have a large font. The block display is used so that the subsequent elements will be placed below it, and its font is changed to double the size of the normal text.

```css
title {
    display: block;
    font-size: 2em;
}
```

**description**

Contains several lines of text describing the report. The lines of text are displayed one below the other, so the description has the block display. Also, the background color is changed to make it standout.

```css
description {
    display: block;
    background-color: #EEEEFF;
    color: black;
}
```

**line**

A line of text in the description. A specific aspect is not defined and it just indicates that the display should be block style.

```css
line {
    display: block;
}
```

**important**

Defines important text from the description. Since it can be mixed with text, its display property must be set to inline. Also, the text is emphasized with bold to make it easier to spot.

```css
important {
    display: inline;
    font-weight: bold;
}
```

**results**

Displays the list of test_names and the results for each one. To make it easier to read, it is displayed as a table, with a green border and margins.

```css
results{
    display: table;
```
entry

The results are displayed as a table so the entry is a row in the table. Thus, the display is `table-row`.

```css
entry {
  display: table-row;
}
```

test_name, passed

The name of the individual test, and its result. They are cells in the results table with the display set to `table-cell`. Padding and a border are added to emphasize the table grid.

```css
test_name, passed{
  display: table-cell;
  border: 1px solid green;
  padding: 20px;
}
```

```css
passed{
  font-weight: bold;
}
```

The full content of the CSS file `test_report.css` is:

```css
report {
  display: block;
  margin: 2em;
}

description {
  display: block;
  background-color: #EEEEFF;
  color: black;
}
```

```css
line {
  display: block;
}
```
important {
    display:inline;
    font-weight:bold;
}

title {
    display:block;
    font-size:2em;
}

results{
    display:table;
    margin:2em;
    border:1px solid green;
}

entry {
    display:table-row;
}

test_name, passed{
    display:table-cell;
    border:1px solid green;
    padding:20px;
}

passed{
    font-weight:bold;
}
XML Instance Template

Based on the XML Schema and CSS file Oxygen XML Editor can help the content author in loading, editing, and validating the test reports. An XML document template must be created as a kind of skeleton that the users can use as a starting point for creating new test reports. The template must be generic enough and reference the XML Schema file and the CSS stylesheet.

This is an example:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/css" href="test_report.css"?>
<report xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:noNamespaceSchemaLocation="test_report.xsd">
    <title>Automated test report</title>
    <description>
        <line>This is the report of the test automatically ran. Each test suite is ran at 20:00h each day. Please <important>check</important> the failed ones!</line>
    </description>
    <results>
        <entry>
            Database connection test true
        </entry>
        <entry>
            XSLT Transformation test true
        </entry>
        <entry>
            DTD validation test false
        </entry>
    </results>
</report>
```
The processing instruction `xml-stylesheet` associates the CSS stylesheet to the XML file. The `href` pseudo attribute contains the URI reference to the stylesheet file. In the example, the CSS is in the same directory as the XML file.

The next step is to place the XSD file and the CSS file on a web server and modify the template to use the HTTP URLs, like this:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<xml-stylesheet type="text/css" href="http://www.mysite.com/reports/test_report.css"/>
<report xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:noNamespaceSchemaLocation="http://www.mysite.com/reports/test_report.xsd">
  <title>Test report title</title>
  <description>.......

If you want to share the files with other team members, you could create an archive containing the `test_report.xml`, `test_report.css`, and `test_report.xsd` and send it to the other users.

Creating and Configuring a Custom Framework

This basic tutorial is meant to provide an example of creating and configuring a custom document type (framework (on page 3219)). This basic tutorial offers examples for creating a custom schema, adjusting the authoring experience through custom CSS styling, and creating a custom action.

Step 1: Organize Framework Files

First, create a new folder for your customized framework (on page 3219). This folder will be used to store all files related to the documentation framework. The folder structure will look something like this:
The frameworks directory is the container where all the Oxygen XML Editor framework customizations are located. Each subdirectory contains files related to a specific type of XML documents (schemas, catalogs, stylesheets, CSS stylesheets, etc.) Distributing a framework means delivering a framework directory.

It is assumed that you have the right to create files and folders inside the Oxygen XML Editor installation directory. If you do not have this right, you will have to install another copy of the program in a folder you have access to, the home directory for instance, or your desktop.

To test your framework distribution, copy it in the frameworks directory of the newly installed application and start Oxygen XML Editor by running the provided start-up script files.

You should copy the created schema files abs.xsd and sdf.xsd, sdf.xsd being the main schema, to the schema directory and the CSS file sdf.css to the css directory.

Step 2: Extend an Existing Framework

The easiest way to create a custom framework (on page 3219) (document type) is by extending an existing built-in framework, such as DITA or DocBook, and then making modifications to it.

1. Open the Preferences dialog box (Options > Preferences) (on page 124) and go to Document Type Association > Locations (on page 139). Add the path to your custom framework directory in the Additional frameworks directories list and click OK or Apply to save your changes.

2. Go to the Document Type Association preferences page (on page 138) and select an existing framework configuration (for example, DocBook) and use the Extend button to create an extension for it.

   **Step Result:** This opens the Document Type Configuration dialog box (on page 140) where you can define the set of rules and settings for your custom framework.

3. Give the extension an appropriate name, select External for the Storage option, click the browsing button ( ) to specify the location of your custom framework directory.

4. Click OK to close the configuration dialog box and then OK or Apply to save your changes.

**Results:** You now have a fully functional framework that you can continue to customize.

Step 3: Create a Custom XML Schema

To illustrate an example of creating an XML Schema for a custom DocBook framework (on page 3219), suppose the documents are either articles or books, and composed of sections. The sections may contain
<title>, <para>, <figure>, <table>, and other <section> elements. To complete the picture, each section includes a <def> element from another namespace.

The first schema file:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
    targetNamespace="http://www.oxygenxml.com/sample/documentation"
    xmlns:doc="http://www.oxygenxml.com/sample/documentation"
    xmlns:abs="http://www.oxygenxml.com/sample/documentation/abstracts"
    elementFormDefault="qualified">

    <xs:import namespace="http://www.oxygenxml.com/sample/documentation/abstracts"
        schemaLocation="abs.xsd"/>

</xs:schema>
```


Next, the structure of the sections is defined. They all start with a <title>, then have the optional <def> element then either a sequence of other sections, or a mixture of paragraphs, images, and tables.

```xml
<xs:element name="book" type="doc:sectionType"/>
<xs:element name="article" type="doc:sectionType"/>
<xs:element name="section" type="doc:sectionType"/>

<xs:complexType name="sectionType">
    <xs:sequence>
        <xs:element name="title" type="xs:string"/>
        <xs:element ref="abs:def" minOccurs="0"/>
        <xs:choice>
            <xs:sequence>
                <xs:element ref="doc:section" maxOccurs="unbounded"/>
            </xs:sequence>
            <xs:choice maxOccurs="unbounded">
                <xs:element ref="doc:para"/>
                <xs:element ref="doc:image"/>
                <xs:element ref="doc:table"/>
            </xs:choice>
        </xs:choice>
    </xs:sequence>
</xs:complexType>
```
The paragraph contains text and other custom styling markup, such as bold \((\texttt{\textbf{b}})\) and italic \((\texttt{\textit{i}})\) elements.

```xml
<xs:element name="para" type="doc:paragraphType"/>
<xs:complexType name="paragraphType" mixed="true">
   <xs:choice minOccurs="0" maxOccurs="unbounded">
      <xs:element name="emphasis"/>
      <xs:element name="i"/>
   </xs:choice>
</xs:complexType>
```

The \(<\texttt{image}>\) element has an attribute with a reference to the file containing image data.

```xml
<xs:element name="image">
   <xs:complexType>
      <xs:attribute name="href" type="xs:anyURI" use="required"/>
   </xs:complexType>
</xs:element>
```

The \(<\texttt{table}>\) element contains a header row and then a sequence of rows \(<\texttt{tr}>\) elements each of them containing the cells. Each cell has the same content as the paragraphs.

```xml
<xs:element name="table">
   <xs:complexType>
      <xs:sequence>
         <xs:element name="header">
            <xs:complexType>
               <xs:sequence>
                  <xs:element name="td" maxOccurs="unbounded" type="doc:paragraphType"/>
               </xs:sequence>
            </xs:complexType>
         </xs:element>
         <xs:element name="tr" maxOccurs="unbounded">
            <xs:complexType>
               <xs:sequence>
                  <xs:element name="td" type="doc:tdType" maxOccurs="unbounded"/>
               </xs:sequence>
            </xs:complexType>
         </xs:element>
      </xs:sequence>
   </xs:complexType>
</xs:element>
```
The `<def>` element is defined as a text only element in the imported schema `abs.xsd`:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
    targetNamespace="http://www.oxygenxml.com/sample/documentation/abstracts">
    <xs:element name="def" type="xs:string"/>
</xs:schema>
```

Now the XML data structure will be styled.

**Step 4: Associate the Schema to the Framework**

In the bottom section of the Document Type configuration dialog box (on page 140), there are a series of tabs. The Schema tab refers to the schema that is used for validation of the documents that match the defined association rules.

**Important:**
If the document references a schema directly (for example, using a `DOCTYPE` declaration, `xsi:schemaLocation` attribute, or a Relax NG `xml-model` processing instruction), the schema defined in this Schema tab is not used for validation or content completion.

**Schema Type**
Select from the combo box the value XML Schema.

**Schema URI**
Enter the value of the schema location (for example, `$(framework)/schema/sdf.xsd`). Use the `$${frameworks}` editor variable (on page 330) in the schema URI path instead of a full path to be valid for multiple Oxygen XML Editor installations.

**Important:**
The `$${frameworks}` variable is expanded at the time of validation into the absolute location of the directory containing the framework (on page 3219).
Step 5: Create a Custom CSS

If you read the Framework Customization Overview (on page 2290) then you already have some basic knowledge about creating simple styles. The example document contains elements from various namespaces, so you need to use CSS Level 3 extensions (supported by the Author mode layout engine) to associate specific properties with that element.

Defining the General Layout

Now the basic layout of the rendered documents is created.

Elements that are stacked one on top of the other are: book, article, section, title, figure, table, image. These elements are marked as having block style for display. Elements that are placed one after the other in a flowing sequence are: b, i. These will have inline display.

```css
/* Vertical flow */
book,
section,
para,
title,
image,
ref {
    display:block;
}

/* Horizontal flow */
b,i {
    display:inline;
}
```

⚠️ Important:
Having block display children in an inline display parent results in Oxygen XML Editor changing the style of the parent to block display.

Styling an Element

The title of any section must be bold and smaller than the title of the parent section. To create this effect, a sequence of CSS rules must be created. The * operator matches any element, it can be used to match titles having progressive depths in the document.

```css
title{
    font-size: 2.4em;
    font-weight:bold;
}
* * title{
```
It is useful to have before the title a constant text, indicating that it refers to a section. This text can include also the current section number. The `:before` and `:after` pseudo-elements will be used, plus the CSS counters.

First declare a counter named `sect` for each `book` or `article`. The counter is set to zero at the beginning of each such element:

```css
book,
article{
    counter-reset: sect;
}
```

The `sect` counter is incremented with each `section`, that is a direct child of a `book` or an `article` element.

```css
book > section,
article > section{
    counter-increment: sect;
}
```

The "static" text that will prefix the section title is composed of the constant "Section ", followed by the decimal value of the `sect` counter and a dot.

```css
book > section > title:before,
article > section > title:before{
    content: "Section " counter(sect) ". "
}
```

To make the documents easy to read, you add a margin to the sections. In this way the higher nesting level, the larger the left side indent. The margin is expressed relatively to the parent bounds:

```css
section{
    margin-left: 1em;
    margin-top: 1em;
}
```
In the above screenshot you can see a sample XML document rendered by the CSS stylesheet. The selection "avoids" the text that is generated by the CSS "content" property. This happens because the CSS generated text is not present in the XML document and is just a visual aid.

**Styling Inline Elements**

The "bold" style is obtained by using the `font-weight` CSS property with the value `bold`, while the "italic" style is specified by the `font-style` property:

```css
b {
    font-weight: bold;
}

i {
    font-style: italic;
}
```

**Styling Images**

The CSS 2.1 does not specify how an element can be rendered as an image. To overpass this limitation, Oxygen XML Editor supports a CSS Level 3 extension allowing to load image data from a URL. The URL of the image must be specified by one of the element attributes and it is resolved through the catalogs specified in Oxygen XML Editor.

```css
image{
    display: block;
}
```
The `image` element has the required `@href` attribute of type `xs:anyURI`. The `@href` attribute contains an image location so the rendered content is obtained by using the function:

```css
attr(href, url)
```

The first argument is the name of the attribute pointing to the image file. The second argument of the `attr` function specifies the type of the content. If the type has the `url` value, then Oxygen XML Editor identifies the content as being an image. If the type is missing, then the content will be the text representing the attribute value.

Oxygen XML Editor handles both absolute and relative specified URLs. If the image has an `absolute` URL location (for example: "http://www.oasis-open.org/images/standards/oasis_standard.jpg") then it is loaded directly from this location. If the image URL is `relative` specified to the XML document (for example: "images/my_screenshot.jpg") then the location is obtained by adding this value to the location of the edited XML document.

An image can also be referenced by the name of a DTD entity that specifies the location of the image file. For example, if the document declares an entity `graphic` that points to a JPEG image file:

```xml
<!ENTITY graphic SYSTEM "depo/keyboard_shortcut.jpg" NDATA JPEG>
```

and the image is referenced in the XML document by specifying the name of the entity as the value of an attribute:

```xml
<mediaobject>
  <imageobject>
    <imagedata entityref="graphic" scale="50"/>
  </imageobject>
</mediaobject>
```

The CSS should use the functions `url`, `attr` and `unparsed-entity-uri` for displaying the image in the **Author** mode:

```css
imagedata[entityref]{
  content: url(unparsed-entity-uri(attr(entityref)));
}
```

To take into account the value of the `@width` attribute of the `imagedata` and use it for resizing the image, the CSS can define the following rule:

```css
imagedata[width]{
  width:attr(width, length);
}
```
Figure 599. Samples of images in Author

Section: 3 Documentation frameworks

One of the most important documentation frameworks is Docbook.

The other is the topic-oriented DITA, promoted by OASIS.

Step 6: Associate the Custom CSS to the Framework

Once you have customized your framework through CSS styling rules, you then need to associate the custom CSS file (on page 146).

Step 7: Testing the Framework Customization

To test the new framework (on page 3219) customization, create an XML instance that conforms with the association rules that you specified in your framework customization. You will not specify an XML Schema location directly in the document, using an xsi:schemaLocation attribute. Instead, Oxygen XML Editor will detect its associated document type and use the specified schema.

```xml
<book xmlns="http://www.oxygenxml.com/sample/documentation"
    xmlns:abs="http://www.oxygenxml.com/sample/documentation/abstracts">

    <title>My Technical Book</title>
    <section>
        <title>XML</title>
        <abs:def>Extensible Markup Language</abs:def>
        <para>In this section of the book I will explain different XML applications.</para>
    </section>

</book>
```

When trying to validate the document there should be no errors. Now modify the title to title2. Validate again. This time there should be an error.
CSS Support in Author Mode

The visual Author editing mode can be customized by creating CSS files to define styles for the XML elements and other components. The Author editing mode supports most CSS 2.1 selectors, numerous CSS 2.1 properties, and some CSS 3 selectors. Oxygen XML Editor also supports stylesheets coded with the LESS dynamic stylesheet language. Also, Oxygen XML Editor has added some custom functions and properties that extend the W3C CSS specification and are useful for a wide range of use-cases for developers who customize Author mode through custom frameworks (on page 2133).

Resources

To see a visual demonstration of various advanced customization possibilities (including ideas for tailoring the editing experience using CSS), watch our Webinar: Working with DITA in Oxygen - Customizing the Editing Experience.

Associating a CSS with an XML Document

**Associating a Stylesheet with an XML Document**

The rendering of an XML document in the Author mode is driven by a CSS stylesheet that conforms to the version 2.1 of the CSS specification from the W3C consortium. Some CSS 3 features, such as namespaces and custom extensions, of the CSS specification are also supported. Oxygen XML Editor also supports stylesheets coded with the LESS dynamic stylesheet language.

There are several methods for associating a stylesheet (CSS or LESS) with an XML document:
• Insert the `xml-stylesheet` processing instruction with the `@type` attribute at the beginning of the XML document. The easiest way to do this is by using the ✧ Associate XSLT/CSS Stylesheet action that is available on the toolbar or in the Document > XML Document menu.

CSS example:

```xml
<?xml-stylesheet type="text/css" href="test.css"?>
```

LESS example:

```xml
<?xml-stylesheet type="text/css" href="test.less"?>
```

• Add a new CSS or LESS file to a framework (document type). To do so, open the Preferences dialog box (Options > Preferences) and go to Document Type Association. Edit the appropriate framework, open the Author tab, then the CSS subtab. Click the ✪ New button to add a new CSS or LESS file.

![Note:]
The built-in frameworks are read-only, so you need to Extend (on page 138) or Duplicate (on page 138) them to configure them as custom frameworks.

If a document has no CSS association or the referenced stylesheet files cannot be loaded, a default one is used. A warning message is also displayed at the beginning of the document, presenting the reason why the CSS cannot be loaded.

![Figure 601. Document with no CSS association default rendering](image)

For information about associating a CSS to a framework (document type), see Configuring and Managing Multiple CSS Styles for a Framework (on page 2145).
Handling CSS Imports

When a CSS document contains imports to other CSS documents, the references are also passed through the XML Catalog (on page 3225) URI mappings to determine an indirect CSS referenced location.

Example: CSS Import

For example, if you can have a CSS import, such as:

```xml
@import "http://host/path/to/location/custom.css";
```

and then add your own XML Catalog (on page 3225) file that maps the location to a custom CSS in the XML Catalog preferences page (on page 235):

```xml
<uri name="http://host/path/to/location/custom.css"
     uri="path/to/custom.css"/>
```

Adding a Custom Default CSS for Every XML Document

To add a custom CSS that is applied to every XML document, add a mapping in your XML Catalog (on page 3225) file that looks like this:

```xml
<uri name="http://www.oxygenxml.com/extensions/author/css/userCustom.css"
     uri="path/to/custom.css"/>
```

This extra mapped CSS location will be parsed every time the application processes the CSS stylesheets used to render the opened XML document in the visual Author editing mode. This allows your custom CSS to be used without the need to modify all other CSS stylesheets contributed in the document type configuration.

Editor Variables in CSS Imports

You can use various editor variables (on page 323) in CSS imports. When editing an XML document with an associated CSS in Author mode, the editor variables will be expanded and resolved.

Example: Editor Variable in a CSS Import

For example, the following editor variable:

```xml
@import "${framework(DITA)}/custom.css";
```

is resolved in the DITA framework (on page 3219) folder where the custom.css is placed. In the Document Type Association preferences page (on page 138), you can see a list of document type. The name for your particular document type needs to be passed as a parameter to the framework() function.

Note:
If you use editor variables like ${cfdu} (Current File Directory URL), it will be expanded to the URL of the current CSS document that contains the imports rather than the XML document that references the CSS.
Displaying Processing Instructions from Other XML Editors

By default, some external processing instructions are hidden (for example, certain processing instructions used to store metadata in other XML editors). If you want them to be displayed (for example, to edit them), they must be associated with the current document using a CSS (on page 2307) like this:

```css
@namespace oxy "http://www.oxygenxml.com/extensions/author";
oxy|processing-instruction[Pub],
oxyl|processing-instruction[PubTbl],
oxyl|processing-instruction[xm-replace_text],
oxyl|processing-instruction[xm-deletion_mark],
oxyl|processing-instruction[xm-insertion_mark_start],
oxyl|processing-instruction[xm-insertion_mark_end]
{
    display:block !important;
}
```

Specifying Media Types in the CSS

The CSS stylesheets can specify how a document is presented on different types of media (on the screen, paper, etc.) You can specify that some of the selectors from your CSS should be taken into account only in the Oxygen XML Editor Author mode and ignored in other media types. This can be accomplished by using the oxygen media type.

**Example: oxygen Media Type**

```css
b{
    font-weight:bold;
    display:inline;
}

@media oxygen{
    b{
        text-decoration:underline;
    }
}
```

This example results in the text being bold if the document is opened in a web browser that does not recognize `@media oxygen`, while the text is bold and underlined when opened in Oxygen XML Editor Author mode.

You can also use the oxygen media type to specify CSS selectors to be applied in certain operating systems or platforms by using the os and platform properties. For example, you can specify one set of style rules for displaying Oxygen XML Editor in Windows, and another set of style rules for macOS. The supported properties are as follows:
• os - The possible values are: \texttt{win}, \texttt{linux}, or \texttt{mac}.
• platform - The possible values are: \texttt{standalone}, \texttt{eclipse}, or \texttt{webapp}.

**Example: os and platform Properties**

```css
@media oxygen AND (os:"win") AND (platform:"standalone") { 

  p{ 

    content:"PPP";

  }

}
```

**Related information**

@media Rule (on page 2311)

### CSS At-Rules

Oxygen XML Editor supports some of the standard at-rules specified by CSS Level 2.1 and 3. The \texttt{@media} rule also include support for some style rules that are specific to Oxygen XML Editor.

**@font-face At-Rule**

Oxygen XML Editor allows you to use custom fonts in the Author mode by specifying them in the CSS using the \texttt{@font-face} media type. Only the \texttt{src} and \texttt{font-family} CSS properties can be used for this media type.

**Example: @font-face Rule**

```css
@font-face{ 

  font-family:"Baroque Script";

  /*The location of the loaded TTF font must be relative to the CSS*/

  src:url("BaroqueScript.ttf");

}
```

**@media Rule**

The \texttt{@media} rule allows you to set different style rules for multiple types of media in the same stylesheet. For example, you can set the font size to be different on the screen than on paper. Oxygen XML Editor supports several media types, allowing you to set the style rules for presenting a document on various media (on screen, paper, etc.).

### Supported Media Types

• \texttt{screen} - The styles marked with this media type are used only for rendering a document on screen.
• \texttt{print} - The styles marked with this media type are used only for printing a document.
• \texttt{all} - The styles marked with this media type are used for rendering a document in all supported types of media.
• **oxygen** - The styles marked with this media type are used only for rendering a document in the Oxygen XML Editor **Author** mode. For more information, see *Specifying Media Types in the CSS* *(on page 2310)*.

• **oxygen-dark-theme** - The styles marked with this media type are used only for rendering a document in the Oxygen XML Editor **Author** mode when a dark theme is used (for example, Graphite).

• **oxygen-high-contrast-black** - The styles marked with this media type are used only for rendering a document in the Oxygen XML Editor **Author** mode on a Windows High Contrast Theme with a black background.

• **oxygen-high-contrast-white** - The styles marked with this media type are used only for rendering a document in the Oxygen XML Editor **Author** mode on a Windows High Contrast Theme with a white background.

**Example: @media Rule**

```css
@media oxygen{
  b{
    text-decoration:underline;
  }
}
@media oxygen-high-contrast-white{
  b{
    font-weight:bold;
  }
}
```

**Supported Properties**

Oxygen XML Editor also supports a few properties to set specific style rules that depend upon the size of the visible area in **Author** mode. These supported properties are as follows:

• **min-width** - The styles selected in this property are applied if the visible area in **Author** mode is equal to or greater than the specified value.

• **max-width** - The styles selected in this property are applied if the visible area in **Author** mode is less than or equal to the specified value.

**Example: min-width and max-width Properties**

```css
@media (min-width:500px){
  p{
    content:'XXX';
  }
}
@media (max-width:700px){
  p:after{
    content:'yyy';
  }
}
```
Standard W3C CSS Supported Features

Oxygen XML Editor supports most of the CSS Level 3 selectors and most of the CSS Level 2.1 properties

Supported CSS Selectors

Tip:
CSS rules that match attributes are always more specific than element selectors. For more information, see https://drafts.csswg.org/selectors-3/#specificity.

The following table lists the CSS selectors that are supported in Oxygen XML Editor:

<table>
<thead>
<tr>
<th>Expression</th>
<th>Name</th>
<th>CSS Level</th>
<th>Description / Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>*</td>
<td>Universal selector</td>
<td>CSS Level 2</td>
<td>Matches any element</td>
</tr>
<tr>
<td>E</td>
<td>Type selector</td>
<td>CSS Level 2</td>
<td>Matches any E element (i.e. an element with the local name E)</td>
</tr>
<tr>
<td>E F</td>
<td>Descendant selector</td>
<td>CSS Level 2</td>
<td>Matches any F element that is a descendant of an E element.</td>
</tr>
<tr>
<td>E &gt; F</td>
<td>Child selectors</td>
<td>CSS Level 2</td>
<td>Matches any F element that is a child of an element E.</td>
</tr>
<tr>
<td>E:lang(c)</td>
<td>Language pseudo-class</td>
<td>CSS Level 2</td>
<td>Matches element of type E if it is in (human) language E (the document language specifies how language is determined).</td>
</tr>
<tr>
<td>E + F</td>
<td>Adjacent selector</td>
<td>CSS Level 2</td>
<td>Matches any F element immediately preceded by a sibling element E.</td>
</tr>
<tr>
<td>E ~ F</td>
<td>General sibling selector</td>
<td>CSS Level 3</td>
<td>Matches any F element preceded by a sibling element E.</td>
</tr>
<tr>
<td>E[foo]</td>
<td>Attribute selector</td>
<td>CSS Level 2</td>
<td>Matches any E element with the &quot;foo&quot; attribute set (whatever the value).</td>
</tr>
<tr>
<td>Expression</td>
<td>Name</td>
<td>CSS Level</td>
<td>Description / Example</td>
</tr>
<tr>
<td>---------------------</td>
<td>-------------------------------------------</td>
<td>-----------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>E[foo=&quot;warning&quot;]</td>
<td>Attribute selector with value</td>
<td>CSS Level 2</td>
<td>Matches any E element whose &quot;foo&quot; attribute value is exactly equal to &quot;warning&quot;.</td>
</tr>
<tr>
<td>E[foo~=&quot;warning&quot;]</td>
<td>Attribute selector containing value</td>
<td>CSS Level 2</td>
<td>Matches any E element whose &quot;foo&quot; attribute value is a list of space-separated values, one of which is exactly equal to &quot;warning&quot;.</td>
</tr>
<tr>
<td>E[lang</td>
<td>=&quot;en&quot;]</td>
<td>Attribute selector containing hyphen separated values</td>
<td>CSS Level 2</td>
</tr>
<tr>
<td>E:before and E:after</td>
<td>Pseudo-elements</td>
<td>CSS Level 2</td>
<td>The ':before' and ':after' pseudo-elements can be used to insert generated content before or after an element's content.</td>
</tr>
<tr>
<td>E:first-child</td>
<td>The first-child pseudo-class</td>
<td>CSS Level 2</td>
<td>Matches element E when E is the first child of its parent.</td>
</tr>
<tr>
<td>E:not(s)</td>
<td>Negation pseudo-class</td>
<td>CSS Level 2</td>
<td>An E element that does not match simple selector s.</td>
</tr>
<tr>
<td>E:has</td>
<td>Relational pseudo-class</td>
<td>CSS Level 4</td>
<td>The :has() relational pseudo-class is a functional pseudo-class that takes a relative selector as an argument.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>For more information, see :has Relational Pseudo-Class (on page 2319).</td>
</tr>
<tr>
<td>E:hover</td>
<td>The hover pseudo-class</td>
<td>CSS Level 2</td>
<td>The :hover pseudo-class applies while the user designates an element with a pointing device, but does not necessarily activate it. When moving the pointing device over an element, all the parent elements up to the root are taken into account.</td>
</tr>
<tr>
<td>E:focus</td>
<td>The focus pseudo-class</td>
<td>CSS Level 2</td>
<td>The :focus pseudo-class applies while an element has the focus (accepts keyboard input).</td>
</tr>
</tbody>
</table>
| E:focus-within       | The generalized input focus pseudo-class  | CSS Level 4 | The :focus-within pseudo-class applies to elements that will have the
<table>
<thead>
<tr>
<th>Expression</th>
<th>Name</th>
<th>CSS Level</th>
<th>Description / Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>E:marker</td>
<td>The marker pseudo-class</td>
<td>CSS Level 4</td>
<td>The ::marker pseudo-element represents the automatically generated marker box of a list item.</td>
</tr>
<tr>
<td>E#myid</td>
<td>The ID selector</td>
<td>CSS Level 2</td>
<td>Matches any E element with ID equal to &quot;myid&quot;</td>
</tr>
<tr>
<td>E[att^=&quot;val&quot;]</td>
<td>Substring matching attribute selector</td>
<td>CSS Level 3</td>
<td>An E element whose att attribute value begins exactly with the string val.</td>
</tr>
<tr>
<td>E[att$=&quot;val&quot;]</td>
<td>Substring matching attribute selector</td>
<td>CSS Level 3</td>
<td>An E element whose att attribute value ends exactly with the string val.</td>
</tr>
<tr>
<td>E[att*=&quot;val&quot;]</td>
<td>Substring matching attribute selector</td>
<td>CSS Level 3</td>
<td>An E element whose att attribute value contains the substring val.</td>
</tr>
<tr>
<td>E:root</td>
<td>Root pseudo-class</td>
<td>CSS Level 3</td>
<td>Matches the root element of the document. In HTML, the root element is always the HTML element.</td>
</tr>
<tr>
<td>E:empty</td>
<td>Empty pseudo-class</td>
<td>CSS Level 3</td>
<td>An E element that has no text or child elements.</td>
</tr>
<tr>
<td>E:nth-child(n)</td>
<td>The nth-child pseudo-class</td>
<td>CSS Level 3</td>
<td>An E element, the nth child of its parent.</td>
</tr>
<tr>
<td>E:nth-last-child(n)</td>
<td>The nth-last-child pseudo-class</td>
<td>CSS Level 3</td>
<td>An E element, the nth child of its parent, counting from the last one.</td>
</tr>
<tr>
<td>E:nth-of-type(n)</td>
<td>The nth-of-type pseudo-class</td>
<td>CSS Level 3</td>
<td>An E element, the nth sibling of its type.</td>
</tr>
<tr>
<td>E:nth-last-of-type(n)</td>
<td>The nth-last-of-type pseudo-class</td>
<td>CSS Level 3</td>
<td>An E element, the nth sibling of its type, counting from the last one.</td>
</tr>
<tr>
<td>Expression</td>
<td>Name</td>
<td>CSS Level</td>
<td>Description / Example</td>
</tr>
<tr>
<td>-----------------</td>
<td>------------------------------------------------</td>
<td>-------------</td>
<td>---------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>E:last-child</td>
<td>The last-child pseudo-class</td>
<td>CSS Level 3</td>
<td>An E element, last child of its parent.</td>
</tr>
<tr>
<td>E:first-of-type</td>
<td>The first-of-type pseudo-class</td>
<td>CSS Level 3</td>
<td>An E element, first sibling of its type.</td>
</tr>
<tr>
<td>E:last-of-type</td>
<td>The last-of-type pseudo-class</td>
<td>CSS Level 3</td>
<td>An E element, last sibling of its type.</td>
</tr>
<tr>
<td>E:only-child</td>
<td>The only-child pseudo-class</td>
<td>CSS Level 3</td>
<td>An E element, only child of its parent.</td>
</tr>
<tr>
<td>E:only-of-type</td>
<td>The only-of-type pseudo-class</td>
<td>CSS Level 3</td>
<td>An E element, only sibling of its type.</td>
</tr>
<tr>
<td>ns</td>
<td>E</td>
<td>Element namespace selector</td>
<td>CSS Level 3</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>E!&gt;F</td>
<td>The subject selector</td>
<td>CSS Level 4 (experimental)</td>
<td>An element that has the local name E and has a child F. See Subject Selector (on page 2317).</td>
</tr>
</tbody>
</table>

**Namespace Selector**

In the CSS 2.1 standard, the element selectors ignore the namespaces of the elements they are matching. Only the local name of the elements are considered in the selector matching process.

Oxygen XML Editor uses a different approach that is similar to the CSS Level 3 specification. If the element name from the CSS selector is not preceded by a namespace prefix it is considered to match an element with the same local name as the selector value and ANY namespace. Otherwise, the element must match both the local name and the namespace.

In CSS up to version 2.1, the name tokens from selectors match all elements from ANY namespace that have the same local name. Example:

```xml
<x:b xmlns:x="ns_x"/>
<y:b xmlns:y="ns_y"/>
```

Are both matched by the rule:

```css
b { font-weight: bold}
```

Starting with CSS Level 3, you can create selectors that are namespace aware.

**Example: Defining prefixed and default namespaces**

Given the namespace declarations:
Then:

```
@namespace sync "http://sync.example.org";
@namespace "http://example.com/foo";
```

<table>
<thead>
<tr>
<th>A</th>
<th>Represents the name A in the <a href="http://sync.example.org">http://sync.example.org</a> namespace.</th>
</tr>
</thead>
<tbody>
<tr>
<td>B</td>
<td>Represents the name B in ANY namespace, including NO NAMESPACE.</td>
</tr>
<tr>
<td>C</td>
<td>Represents the name C in ANY namespace, including NO NAMESPACE.</td>
</tr>
</tbody>
</table>

**Example: Defining prefixed namespaces combined with pseudo-elements**

To match the `<def>` element its namespace declares, bind it to the `abs` prefix and then write a CSS rule:

```
@namespace abs "http://www.oxygenxml.com/sample/documentation/abstracts";
```

Then:

```
abs|def
```

 Represents the name "def" in the http://www.oxygenxml.com/sample/documentation/abstracts namespace.

```
abs|def:before
```

 Represents the :before pseudo-element of the "def" element from the http://www.oxygenxml.com/sample/documentation/abstracts namespace.

**Subject Selector**

Oxygen XML Editor supports the subject selector described in CSS Level 4 (currently a working draft at W3C http://www.w3.org/TR/selectors4/). This selector matches a structure of the document, but unlike a compound selector, the styling properties are applied to the subject element (the one marked with "!") instead of the last element from the path.

*The subject of the selector can be explicitly identified by appending an exclamation mark (!) to one of the compound selectors in a selector. Although the element structure that the selector represents is the same with or without the exclamation mark, indicating the subject in this way can change which compound selector represents the subject in that structure.*

**Example:**
A border will be drawn to the table elements that contain a caption, as direct child.

This is different from:

```css
table > caption {
    border: 1px solid red;
}
```

This draws a border around the caption.

**Taking Processing Instructions into Account in CSS Subject Selectors**

You can test for the existence of specific processing instructions (PI) in the child hierarchy of a subject selector.

For example:

```css
@namespace oxy "http://www.oxygenxml.com/extensions/author";

chapter! > oxy|processing-instruction[important][level="high"]{
    color:red;
}
```

This would change the color of a DocBook chapter to red if it contains the `important` processing instruction:

```xml
<chapter>
    <title>A title</title>
    <?important level='high'?>
</chapter>
```

**Descendant Selectors Limitation**

⚠️ **Important:**

The current implementation has a known limitation. The general descendant selectors are taken into account as direct child selectors. For example, the following two CSS selectors are considered equivalent:

```css
a:has(b c)
```

and:

```css
a:has(b>c)
```
Related information

:has Relational Pseudo-Class (on page 2319)

:has Relational Pseudo-Class

Oxygen XML Editor supports the CSS Level 4 subject selector (currently a working draft at W3C http://www.w3.org/TR/selectors4/), as described in Subject Selector (on page 2317). Oxygen XML Editor also supports the :has relational pseudo-class that has similar functionality and it can match an element by taking its child elements into account. For more information, see https://drafts.csswg.org/selectors-4/#relational.

You can create conditions that take into account the structure of the matching element.

**Example: has Pseudo Class**

```css
table:has( tbody > thead){
    border: 1px solid red;
}
```

This example will result in a border being drawn for the table elements that contain at least a `<thead>` element in the `<tbody>` element.

**Taking Processing Instructions into Account in CSS Subject Selectors**

You can test for the existence of specific processing instructions (**pi**) in the child hierarchy of a subject selector.

For example:

```css
@namespace oxy "http://www.oxygenxml.com/extensions/author";

chapter! > oxy|processing-instruction[important][level="high"]{
    color:red;
}
```

This would change the color of a DocBook chapter to red if it contains the `important` processing instruction:

```xml
<chapter>
    <title>A title</title>
    <?important level='high'?>
</chapter>
```

**Descendant Selectors Limitation**

⚠️ **Important:**

The current implementation has a known limitation. The general descendant selectors are taken into account as direct child selectors. For example, the following two CSS selectors are considered equivalent:
### Supported CSS Properties

Oxygen XML Editor validates all CSS 2.1 properties, but does not render *aural* and *paged* categories properties in **Author** mode, as well as some of the values of the *visual* category that are listed below under the **Ignored Values** column. For the Oxygen XML Editor-specific (extension) CSS properties, see [CSS Extensions (on page 2333)](#).

<table>
<thead>
<tr>
<th>Name</th>
<th>Rendered Values</th>
<th>Ignored Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>background</td>
<td>background-color</td>
<td>background-image</td>
</tr>
<tr>
<td>background-attachment</td>
<td>NONE</td>
<td></td>
</tr>
<tr>
<td>background-color</td>
<td>&lt;color&gt;</td>
<td>inherit</td>
</tr>
<tr>
<td>background-image</td>
<td>&lt;uri&gt;</td>
<td>none</td>
</tr>
<tr>
<td>background-position</td>
<td>top</td>
<td>right</td>
</tr>
<tr>
<td>background-repeat</td>
<td>repeat</td>
<td>repeat-x</td>
</tr>
<tr>
<td>border</td>
<td>[&lt;border-width&gt;</td>
<td></td>
</tr>
<tr>
<td>border-collapse</td>
<td>NONE</td>
<td></td>
</tr>
<tr>
<td>border-color</td>
<td>&lt;color&gt;</td>
<td>inherit</td>
</tr>
<tr>
<td>border-radius</td>
<td>&lt;length&gt; (on page 2325)</td>
<td>&lt;percentage&gt;</td>
</tr>
<tr>
<td></td>
<td>Works only for border-type 'solid', 'dashed', 'dotted', 'wave'. Does not work when background-image is specified. Percent values are not fully supported.</td>
<td></td>
</tr>
<tr>
<td>border-spacing</td>
<td>NONE</td>
<td></td>
</tr>
<tr>
<td>Name</td>
<td>Rendered Values</td>
<td>Ignored Values</td>
</tr>
<tr>
<td>-------------------------------------------</td>
<td>---------------------------------------------------------------------------------</td>
<td>--------------------------------</td>
</tr>
<tr>
<td>border-style</td>
<td><code>&lt;border-style&gt;</code></td>
<td>inherit</td>
</tr>
<tr>
<td>border-top / border-right / border-bottom / border-left</td>
<td>[ <code>&lt;border-width&gt;</code></td>
<td></td>
</tr>
<tr>
<td>border-top-color / border-right-color / border-bottom-color / border-left-color</td>
<td><code>&lt;color&gt;</code></td>
<td>inherit</td>
</tr>
<tr>
<td>border-top-left-radius / border-top-right-radius / border-bottom-left-radius / border-bottom-right-radius</td>
<td><code>&lt;length&gt;</code> (on page 2325)</td>
<td><code>&lt;percentage&gt;</code></td>
</tr>
<tr>
<td>border-top-style / border-right-style / border-bottom-style / border-left-style</td>
<td><code>&lt;border-style&gt;</code></td>
<td>inherit</td>
</tr>
<tr>
<td>border-top-width / border-right-width / border-bottom-width / border-left-width</td>
<td><code>&lt;border-width&gt;</code></td>
<td>inherit</td>
</tr>
<tr>
<td>border-width</td>
<td><code>&lt;border-width&gt;</code></td>
<td>inherit</td>
</tr>
<tr>
<td>bottom</td>
<td><code>&lt;length&gt;</code> (on page 2325)</td>
<td><code>&lt;percentage&gt;</code></td>
</tr>
<tr>
<td>caption-side</td>
<td>NONE</td>
<td></td>
</tr>
<tr>
<td>clear</td>
<td>NONE</td>
<td></td>
</tr>
<tr>
<td>clip</td>
<td>NONE</td>
<td></td>
</tr>
<tr>
<td>color</td>
<td><code>&lt;color&gt;</code></td>
<td>inherit</td>
</tr>
<tr>
<td>content</td>
<td>normal</td>
<td>none</td>
</tr>
<tr>
<td>Name</td>
<td>Rendered Values</td>
<td>Ignored Values</td>
</tr>
<tr>
<td>--------------------</td>
<td>---------------------------------------------------------------------------------</td>
<td>----------------------</td>
</tr>
<tr>
<td>counter-increment</td>
<td>`[ &lt;identifier&gt; &lt;integer&gt; ? ]+</td>
<td>none</td>
</tr>
<tr>
<td>counter-reset</td>
<td>`[ &lt;identifier&gt; &lt;integer&gt; ? ]+</td>
<td>none</td>
</tr>
<tr>
<td>cursor</td>
<td>NONE</td>
<td></td>
</tr>
<tr>
<td>direction</td>
<td>ltr</td>
<td>rtl</td>
</tr>
<tr>
<td>display</td>
<td>inline</td>
<td>block</td>
</tr>
<tr>
<td>empty-cells</td>
<td>show</td>
<td>hide</td>
</tr>
<tr>
<td>float</td>
<td>NONE</td>
<td></td>
</tr>
<tr>
<td>font</td>
<td>`[[ 'font-style'</td>
<td></td>
</tr>
<tr>
<td>font-family</td>
<td>[[ &lt;family-name&gt;</td>
<td>&lt;generic-family&gt; ] [, &lt;family-name&gt;</td>
</tr>
<tr>
<td>font-size</td>
<td>&lt;absolute-size&gt;</td>
<td>&lt;relative-size&gt;</td>
</tr>
<tr>
<td>font-style</td>
<td>normal</td>
<td>italic</td>
</tr>
<tr>
<td>font-variant</td>
<td>NONE</td>
<td></td>
</tr>
<tr>
<td>Name</td>
<td>Rendered Values</td>
<td>Ignored Values</td>
</tr>
<tr>
<td>-----------------------</td>
<td>--------------------------------------------------------------------------------</td>
<td>--------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>font-weight</td>
<td>normal</td>
<td>bold</td>
</tr>
<tr>
<td>height</td>
<td>NONE</td>
<td></td>
</tr>
<tr>
<td>left</td>
<td>&lt;length&gt; (on page 2325)</td>
<td>&lt;percentage&gt;</td>
</tr>
<tr>
<td>letter-spacing</td>
<td>normal</td>
<td>&lt;length&gt; (on page 2325)</td>
</tr>
<tr>
<td>line-height</td>
<td>normal</td>
<td>&lt;number&gt;</td>
</tr>
<tr>
<td>list-style</td>
<td>['list-style-type']</td>
<td>inherit</td>
</tr>
<tr>
<td>list-style-image</td>
<td>NONE</td>
<td></td>
</tr>
<tr>
<td>list-style-position</td>
<td>NONE</td>
<td></td>
</tr>
<tr>
<td>list-style-type</td>
<td>disc</td>
<td>circle</td>
</tr>
<tr>
<td>margin</td>
<td>&lt;margin-width&gt;</td>
<td>inherit</td>
</tr>
<tr>
<td>margin-right / margin-left</td>
<td>&lt;margin-width&gt;</td>
<td>inherit</td>
</tr>
<tr>
<td>margin-top / margin-bottom</td>
<td>&lt;margin-width&gt;</td>
<td>inherit</td>
</tr>
<tr>
<td>max-height</td>
<td>NONE</td>
<td></td>
</tr>
<tr>
<td>max-width</td>
<td>&lt;length&gt; (on page 2325)</td>
<td>&lt;percentage&gt;</td>
</tr>
<tr>
<td>Name</td>
<td>Rendered Values</td>
<td>Ignored Values</td>
</tr>
<tr>
<td>-----------------------</td>
<td>---------------------------------------------------------------------------------</td>
<td>--------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>min-height</td>
<td>Absolute values, such as (230\text{px}, 1\text{in}, 7\text{pt}, 12\text{em})</td>
<td>Values proportional to the parent element height, such as (30%)</td>
</tr>
<tr>
<td>min-width</td>
<td>(&lt;\text{length}&gt;(\text{on page 2325})) (&lt;\text{percentage}&gt;) (\text{inherit}) supported for (\text{inline}, \text{block-level}), and replaced elements (such as images, tables, table cells) (\text{initial}) (\text{unset})</td>
<td></td>
</tr>
<tr>
<td>outline</td>
<td>([&lt;\text{outline-width}&gt;</td>
<td></td>
</tr>
<tr>
<td>outline-color</td>
<td>([&lt;\text{color}&gt;</td>
<td></td>
</tr>
<tr>
<td>outline-style</td>
<td>([&lt;\text{border-style}&gt;</td>
<td></td>
</tr>
<tr>
<td>outline-width</td>
<td>([&lt;\text{border-width}&gt;</td>
<td></td>
</tr>
<tr>
<td>overflow</td>
<td>NONE</td>
<td></td>
</tr>
<tr>
<td>padding</td>
<td>(&lt;\text{padding-width}&gt;) (\text{inherit}) (\text{initial}) (\text{unset})&lt;br&gt; padding</td>
<td></td>
</tr>
<tr>
<td>padding-top / padding-right / padding-bottom / padding-left</td>
<td>(&lt;\text{padding-width}&gt;) (\text{inherit}) (\text{initial}) (\text{unset})&lt;br&gt; padding-top / padding-right / padding-bottom / padding-left</td>
<td></td>
</tr>
<tr>
<td>position</td>
<td>absolute (\text{fixed}) (supported for block display elements) (\text{relative}) (supported for (\text{block}) and (\text{inline}) display elements) (\text{absolute}) (\text{fixed not supported for inline}) display elements</td>
<td></td>
</tr>
<tr>
<td>quotes</td>
<td>NONE</td>
<td></td>
</tr>
<tr>
<td>right</td>
<td>(&lt;\text{length}&gt;(\text{on page 2325})) (&lt;\text{percentage}&gt;) (\text{inherit}) (\text{initial}) (\text{unset})&lt;br&gt; right</td>
<td></td>
</tr>
<tr>
<td>table-layout</td>
<td>auto (\text{initial}) (\text{unset})&lt;br&gt; table-layout</td>
<td></td>
</tr>
<tr>
<td>text-align</td>
<td>left (\text{right}) (\text{center}) (\text{inherit}) (\text{initial}) (\text{unset})&lt;br&gt; text-align</td>
<td></td>
</tr>
<tr>
<td>text-decoration</td>
<td>none ([&lt;\text{underline}</td>
<td></td>
</tr>
<tr>
<td>Name</td>
<td>Rendered Values</td>
<td>Ignored Values</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>---------------------------------------------------------------------------------</td>
<td>----------------</td>
</tr>
<tr>
<td>text-decoration-style</td>
<td>solid</td>
<td>double</td>
</tr>
<tr>
<td>text-indent</td>
<td>&lt;length&gt; <em>(on page 2325)</em></td>
<td>&lt;percentage&gt;</td>
</tr>
<tr>
<td>text-transform</td>
<td>none</td>
<td>capitalize</td>
</tr>
<tr>
<td>top</td>
<td>&lt;length&gt; <em>(on page 2325)</em></td>
<td>&lt;percentage&gt;</td>
</tr>
<tr>
<td>unicode-bidi</td>
<td>bidi-override</td>
<td>normal</td>
</tr>
<tr>
<td>vertical-align</td>
<td>baseline</td>
<td>sub</td>
</tr>
<tr>
<td>visibility</td>
<td>visible</td>
<td>hidden</td>
</tr>
<tr>
<td>white-space</td>
<td>normal</td>
<td>pre</td>
</tr>
<tr>
<td>width</td>
<td>&lt;length&gt; <em>(on page 2325)</em></td>
<td>&lt;percentage&gt;</td>
</tr>
<tr>
<td>word-spacing</td>
<td>NONE</td>
<td></td>
</tr>
<tr>
<td>z-index</td>
<td>NONE</td>
<td></td>
</tr>
</tbody>
</table>

*<length>* - Refers to distance measurements and is expressed in units such as mm, cm, in, em, rem, ex, pc, pt, px. For more information, see the W3 CSS Level 3 length type specifications.

Related Information:

CSS Extensions *(on page 2333)*

**Transparent Colors**

CSS3 supports RGBA colors. The RGBA declaration allows you to set opacity (via the Alpha channel) as part of the color value. A value of 0 corresponds to a completely transparent color, while a value of 1 corresponds to a completely opaque color. To specify a value, you can use either a real number between 0 and 1, or a percent.

**Example:** RGBA Color
attr() Function: Properties Values Collected from the Edited Document

In CSS Level 2.1 you may collect attribute values and use them as content only for the pseudo-elements. For instance, the :before pseudo-element can be used to insert some content before an element. This is valid in CSS 2.1:

```
title:before{
    content: "[Audience Level: " attr(audience) "]";
}
```

If the `<title>` element from the XML document is:

```
<title audience="Expert">Changing the Timing Belt</title>
```

Then the title will be displayed as:

```
[Audience Level: Expert] Changing the Timing Belt
```

In Oxygen XML Editor, the use of attr() function is available not only for the content property, but also for any other property. This is similar to the CSS Level 3 working draft: [http://www.w3.org/TR/2006/WD-css3-values-20060919/#functional](http://www.w3.org/TR/2006/WD-css3-values-20060919/#functional). The arguments of the function are:

```
attr ( attribute_name , attribute_type , default_value )
```

attribute_name

The attribute name. This argument is required.

attribute_type

The attribute type. This argument is optional. If it is missing, argument's type is considered string. This argument indicates what is the meaning of the attribute value and helps to perform conversions of this value. Oxygen XML Editor accepts one of the following types:

```
color
```
The value represents a color. The attribute may specify a color in various formats. Oxygen XML Editor supports colors specified either by name (red, blue, green, etc.) or as an RGB hexadecimal value \#FFEEFF.

`url`

The value is a URL pointing to a media object. Oxygen XML Editor supports only images. The attribute value can be a complete URL, or a relative one to the XML document. Note that this URL is also resolved through the catalog resolver.

`integer`

The value must be interpreted as an integer.

`number`

The value must be interpreted as a float number.

`length`

The value must be interpreted as an integer.

`percentage`

The value must be interpreted relative to another value (length, size) expressed in percents.

`em`

The value must be interpreted as a size. 1 em is equal to the font-size of the relevant font.

`ex`

The value must be interpreted as a size. 1 ex is equal to the height of the x character of the relevant font.

`px`

The value must be interpreted as a size expressed in pixels relative to the viewing device.

`mm`

The value must be interpreted as a size expressed in millimeters.

`cm`

The value must be interpreted as a size expressed in centimeters.

`in`

The value must be interpreted as a size expressed in inches. 1 inch is equal to 2.54 centimeters.
pt

The value must be interpreted as a size expressed in points. The points used by CSS2 are equal to 1/72th of an inch.

pc

The value must be interpreted as a size expressed in picas. 1 pica is equal to 12 points.

default_value

This argument specifies a value that is used by default if the attribute value is missing. This argument is optional.

Example: attr Function

Consider the following XML instance:

```xml
<sample>
  <para bg_color="#AAAAFF">Blue paragraph.</para>
  <para bg_color="red">Red paragraph.</para>
  <para bg_color="red" font_size="2">Red paragraph with large font.</para>
  <para bg_color="#00AA00" font_size="0.8" space="4">Green paragraph with small font and margin.</para>
</sample>
```

The `<para>` elements have `@bg_color` attributes with RGB color values (such as `#AAAAFF`). You can use the `attr()` function to change the elements appearance in the editor based on the value of this attribute:

```css
background-color: attr(bg_color, color);
```

The `font_size` represents the font size in *em* units. You can use this value to change the style of the element:

```css
font-size: attr(font_size, em);
```

The complete CSS rule is:

```css
para{
  display:block;
  background-color: attr(bg_color, color);
  font-size: attr(font_size, em);
  margin: attr(space, em);
}
```

The document is rendered as:
CSS Level 3 target-counter() and target-counters() Functions

The CSS Level 3 functions \texttt{target-counter} and \texttt{target-counters} can be used as values for the \texttt{content} property to retrieve counter values and display information obtained from a target at the end of a link.

The \texttt{target-counter} Function

This function retrieves the value of the innermost counter with a given name.

\begin{verbatim}
\texttt{target-counter (<fragment>, <counter-name>[, <counter-style>]?)}
\end{verbatim}

\textbf{fragment}

The URI fragment pointing to the ID of the target element.

\textbf{counter-name}

The name of the counter. This argument is required.

\textbf{counter-style}

This optional argument can be used to format the result.

\textbf{Example:}

\textbf{HTML:}

\begin{verbatim}
<nav>
  <ol>
    <li class="frontmatter"><a href="#pref_01">Preface</a></li>
  </ol>
</nav>
\end{verbatim}
The **target-counters** Function

This function fetches the value of all counters of a given name from the end of a link and formats them by inserting a given string between the value of each nested counter.

```
target-counter ( <fragment> , <counter-name> , <delimiter> [ , <counter-style> ] ? )
```

**fragment**

The URI fragment pointing to the ID of the target element.

**counter-name**

The name of the counter. This argument is required.

**delimiter**

The string to be inserted between the value of each nested counter. This argument is required.

**counter-style**

This optional argument can be used to format the result.

Related information

https://www.w3.org/TR/css-gcpm-3/#target-counter

https://www.w3.org/TR/css-gcpm-3/#target-counters
**calc() Function**

The `calc()` function allows mathematical expressions with addition (+), subtraction (-), multiplication (*), division (/) to be used as component values. Percentages are solved relative to the dimensions of the containing parent block. It can be used when length values are accepted:

```css
elem {
  width: calc(100% - 1em);
}
```

For more information, see: [https://drafts.csswg.org/css-values-3/#calc-notation](https://drafts.csswg.org/css-values-3/#calc-notation)

**Custom CSS Properties (CSS Variables)**

*Custom properties* (also referred to as *CSS variables*) are properties defined by CSS authors that contain specific values to be reused throughout a document.

Complex websites have many CSS rules, often with a lot of repeated values. For example, the same color might be used in dozens of different places, requiring a global search-and-replace operation if that color needs to be changed. Custom properties allow a value to be stored in one place, then referenced in multiple other places. An additional benefit is semantic identifiers. For example, `--main-text-color` is easier to understand than `#00ff00`, especially if this same color is also used in other contexts.

*Custom properties* follow the same rules as other CSS properties, so you are able to define and use them at multiple levels, following standard CSS cascading and specificity rules.

**Usage**

A custom property name begins with a double hyphen (--) and its value that can be any valid CSS value. You use the custom property value by specifying your custom property name inside the `var()` function, in place of a regular property value:

```
defining a Custom Property

```css
element {
  --main-bg-color: brown;
  background-color: var(--main-bg-color);
}
```

**Note:**

Custom property names are case sensitive: `--my-color` will be treated as a separate custom property from `--My-color`.

**Inheritance of Custom Properties**

If you define a custom property on an element, you will be able to access it on all of its descendents.
Inheritance

```xml
<one>
  <two>
    <three/>
    <four/>
  </two>
</one>
```

```css
one {
  --color:green;
}

three {
  --color:red;
}

* {
  color: var(--color);
}
```

Result:

- `<one>` has the color green.
- `<two>` has the color green.
- `<three>` has the color red.
- `<four>` has the color green.

Custom Properties Fallback Values

The `var()` function has two arguments. The first argument is the name of the custom property to be substituted. The second argument (optional) is a fallback value, which is used as the substitution value when the referenced custom property is invalid or undefined.

Specifying a Fallback Value

```css
one {
  color: var(--color, blue);
}
```

```css
one {
  color: var(--color, var(--fallback-color, red));
}
```
Dependencies

A custom property can reference the value of another custom property through the `var()` function.

A Custom Property Safely Using a Variable

```css
:root {
    --border-color: red;
    --main-border: 1px solid var(--border-color, green);
}

p {
    border: var(--main-border);
}
```

Combining Custom Variables with `calc()`

```css
:root {
    --foo: 10px;
    --bar: calc(var(--foo) + 10px);
}

p {
    font-size: var(--bar);
}
```

⚠️ CAUTION:

This can create cyclic dependencies where two or more custom properties each attempt to reference each other.

An Invalid Situation of Variables Depending on Each Other

```css
:root {
    --color: var(--bg-color);
    --bg-color: var(--color);
}
```

CSS Extensions

CSS stylesheets provide support for displaying documents. When editing non-standard documents, Oxygen XML Editor CSS extensions are useful.

Examples of how they can be used:

- Property for marking foldable elements (on page 3219) in large files.
- Enforcing a display mode for the XML tags, regardless of the current mode selected by the user.
• Constructing a URL from a relative path location.
• String processing functions.

Built-in CSS Selectors

When Oxygen XML Editor renders content in the Author mode, it adds built-in CSS selectors (in addition to the CSS stylesheets linked in the XML or specified in the document type associated to the XML document). These built-in CSS selectors are processed before all other CSS content, but they can be overwritten if the CSS developer wants to modify a default behavior.

List of CSS Selector Contributed by Oxygen XML Editor

```xml
@namespace oxy "http://www.oxygenxml.com/extensions/author";
@namespace xi "http://www.w3.org/2001/XInclude";
@namespace xlink "http://www.w3.org/1999/xlink";
@namespace svg "http://www.w3.org/2000/svg";
@namespace mml "http://www.w3.org/1998/Math/MathML";

oxy|document {
  display:block !important;
}

oxy|cdata {
  display:-oxy-morph !important;
  white-space:pre-wrap !important;
  border-width:0px !important;
  margin:0px !important;
  padding: 0px !important;
}

oxy|processing-instruction {
  display:-oxy-morph !important;
  color: rgb(139, 38, 201) !important;
  white-space:pre-wrap !important;
  border-width:0px !important;
  margin:0px !important;
  padding: 0px !important;
}

/*EXM-33415 Avoid showing other editors PIs in content, not useful when editing in Oxygen*/
oxy|processing-instruction[Pub],
oxy|processing-instruction[PubTbl],
oxy|processing-instruction[xm-replace_text],
```
oxy|reference,
oxy|entity {
  -oxy-editable:false !important;
  background-color: rgb(240, 240, 240) !important;
  margin:0px !important;
  padding: 0px !important;
}
oxy|reference[editable='true'] {
  -oxy-editable:true !important;
}
oxy|reference {
  display:-oxy-morph !important;
  /*EXM-28674 No need to present tags for these artificial references.*/
  -oxy-display-tags: none;
}
oxy|entity {
  display:-oxy-morph !important;
}
oxy|entity[name='amp'],
oxy|entity[name='lt'],
oxy|entity[name='gt'],
oxy|entity[name='quot'],
oxy|entity[name='apos']{
  /*EXM-32236, EXM-37026 Do not present tags for simple character entity references.*/
  -oxy-display-tags: none;
}
oxy|entity[href] {
  border: 1px solid rgb(175, 175, 175) !important;
  padding: 0.2em !important;
}
/*Wraps multiple fallback elements*/
oxy|include-wrapper {
  display:-oxy-morph !important;
}
xi|include {
display:-oxy-morph !important;
margin-bottom: 0.5em !important;
padding: 2px !important;

} x|include:before,
x|include:after{
display:inline !important;
background-color: inherit !important;
color:#444444 !important;
font-weight:bold !important;
}

x|include:before {
content:url(../images/link.png) attr(href) !important;
-oxy-link: attr(href) !important;
}

x|include[parse="text"]:before {
content:url(../images/link.png) !important;
}

x|include[xpointer]:before {
content:url(../images/link.png) attr(href) " " attr(xpointer) !important;
-oxy-link: oxy_concat(attr(href), "+", attr(xpointer)) !important;
}

x|fallback {
display:-oxy-morph !important;
margin: 2px !important;
border: 1px solid #CB0039 !important;
}

x|fallback:before {

display:-oxy-morph !important;
content:"XInclude fallback: " !important;
color:#CB0039 !important;
}

oxy|doctype {

display:block !important;
background-color: transparent !important;
color:blue !important;
border-width:0px !important;
margin:0px !important;
}
```css
.padding: 2px !important;

@media oxygen-high-contrast-black, oxygen-dark-theme{
  .doctype {
    color:#D0E2F4 !important;
  }
}

.error {
  display:-oxy-morph !important;
  -oxy-editable:false !important;
  white-space:pre !important;
  font-weight:bold !important;
  color: rgb(178, 0, 0) !important;
  -oxy-display-tags: none;
}

.error:before {
  content:url(../images/ReferenceError12.png) "[" !important;
  color: rgb(178, 0, 0) !important;
}

.error[level='warn']:before {
  content:url(../images/ReferenceWarn12.png) "[" !important;
  color: rgb(200, 185, 0) !important;
}

.error[level='warn'] {
  color: rgb(200, 185, 0) !important;
}

.error:after {
  content:"]" !important;
}

*[xlink:href]:before {
  content:url(../images/link.png);
  -oxy-link: attr(xlink:href) !important;
}
/*No direct display of the MathML and SVG images.*/
svg|svg{
    display:inline !important;
    white-space: -oxy-trim-when-ws-only !important;
}

/*EXM-28827 SVG can contain more than one namespace in it*/
svg|svg * {
    display:none !important;
    white-space:normal !important;
}

mml|math{
    display:inline !important;
    white-space: -oxy-trim-when-ws-only !important;
}
mml|math mml|*{
    display:none !important;
    white-space: normal !important;
}

/*Text direction attributes*/
*[dir='rtl'] { direction:rtl; unicode-bidi:embed; }
*[dir='rlo'] { direction:rtl; unicode-bidi:bidi-override; }

*[dir='ltr'] { direction:ltr; unicode-bidi:embed; }
*[dir='lro'] { direction:ltr; unicode-bidi:bidi-override; }

@media oxygen-high-contrast-black, oxygen-dark-theme{
    xi|include:before,
    xi|include:after{
        color:#808080 !important;
    }
}

/*
 * EXM-40349
 *
 * In DIFF these place holder PIs are not handled so we treat them as normal PIs with a bit of styling.
 *
 */
oxy|processing-instruction{oxy-placeholder} {
Example:

To show all entities in the Author mode as transparent, without a gray background, first define in your CSS after all imports the namespace:

```css
@namespace oxy "http://www.oxygenxml.com/extensions/author";
```

and then add the following selector:
Additional CSS Selectors

Oxygen XML Editor provides support for selecting additional types of nodes. These custom selectors apply to: `document, doctype, processing-instruction, comment, CDATA sections, entities, and reference sections`. Processing-instructions are not displayed by default. To display them, open the Preferences dialog box (Options > Preferences) (on page 124), go to Editor > Author, and select Show processing instructions.

Note:
The custom selectors are presented in the default CSS for Author mode and all of their properties are marked with the `!important` flag. For this reason, you have to set the `!important` flag on each property of the custom selectors from your CSS to be applicable.

For the custom selectors to work in your CSS stylesheets, declare the Author mode extensions namespace at the beginning of the stylesheet documents:

```xml
@namespace oxy url(http://www.oxygenxml.com/extensions/author);
```

- **oxy|document** - The `oxy|document` selector matches the entire document:

  ```xml
  oxy|document {
    display: block !important;
  }
  ```

- **oxy|doctype** - The following example changes the rendering of doctype sections:

  ```xml
  oxy|doctype {
    display: block !important;
    color: blue !important;
    background-color: transparent !important;
  }
  ```

- **oxy|processing-instruction** - To match the processing instructions, you can use the `oxy|processing-instruction` selector:

  ```xml
  oxy|processing-instruction {
    display: block !important;
    color: purple !important;
    background-color: transparent !important;
  }
  ```

A processing instruction usually has a target and one or more pseudo attributes:

```xml
<?target_name data="b"?>
```
You can match a processing instruction with a particular target from the CSS using the following construct:

```
oxy|processing-instruction[target_name]
```

You can also match the processing instructions having a certain target and pseudo attribute value, such as:

```
oxy|processing-instruction[target_name][data="b"]
```

- **oxy|comment** - The XML comments displayed in Author mode can be changed using the `oxy|comment` selector:

  ```
  oxy|comment {
    display: block !important;
    color: green !important;
    background-color: transparent !important;
  }
  ```

- **oxy|cdata** - The `oxy|cdata` selector matches CDATA sections:

  ```
  oxy|cdata {
    display: block !important;
    color: gray !important;
    background-color: transparent !important;
  }
  ```

- **oxy|entity** - The `oxy|entity` selector matches the entity content:

  ```
  oxy|entity {
    display: morph !important;
    editable: false !important;
    color: orange !important;
    background-color: transparent !important;
  }
  ```

To match particular entities, use the `oxy|entity` selector in expressions such as:

```
oxy|entity[name='amp'],
oxy|entity[name='lt'],
oxy|entity[name='gt'],
oxy|entity[name='quot'],
oxy|entity[name='apos'],
oxy|entity[name='#'] {
  -oxy-display-tags: none;
}
```
• **oxy|reference** - The references to entities, *XInclude*, and DITA `@conref` and `@conkeyref` attributes are expanded by default in **Author** mode and the referenced content is displayed. The referenced resources are displayed inside the element or entity that references them.

You can use the `reference` property to customize the way these references are rendered in **Author** mode:

```xml
oxy|reference {
    border:1px solid gray !important;
}
```

In the **Author** mode, content is highlighted when text contains comments ([on page 642](#)) and changes (if *Track Changes* ([on page 642](#)) was active when the content was modified).

If this content is referenced, the **Author** mode does not display the highlighted areas in the new context. If you want to mark the existence of the comments and changes, you can use the `oxy|reference[comments]`, `oxy|reference[changeTracking]`, and `oxy|reference[changeTracking][comments]` selectors.

**Note:**
Two artificial attributes (`comments` and `changeTracking`) are set on the reference node, containing information about the number of comments and tracked changes in the content.

• The following example represents the customization of the reference fragments that contain comments:

```xml
oxy|reference[comments]:before {
    content: "Comments: " attr(comments) !important;
}
```

• To match reference fragments based on the fact that they contain tracked changes inside, use the `oxy|reference[changeTracking]` selector:

```xml
oxy|reference[changeTracking]:before {
    content: "Change tracking: " attr(changeTracking) !important;
}
```

• Here is an example of how you can set a custom color for the reference containing both tracked changes and comments:

```xml
oxy|reference[changeTracking][comments]:before {
    content: "Change tracking: " attr(changeTracking)
    " and comments: " attr(comments) !important;
}
```
Additional CSS Properties

Oxygen XML Editor provides various additional CSS properties to extend the standard CSS properties.

Append Content Properties: -oxy-append-content / -oxy-prepend-content

Used to append specified content.

-oxy-append-content Property

This property appends the specified content to the content generated by other matching CSS rules of lesser specificity. Unlike the content property, where only the value from the rule with the greatest specificity is taken into account, the -oxy-append-content property adds content to that generated by the lesser specificity rules into a new compound content.

Example:
element:before{
    content: "Hello";
}

The content shown before the element will be Hello World!.

-oxy-prepend-content Property

Prepends the specified content to the content generated by other matching CSS rules of lesser specificity. Unlike the content property, where only the value from the rule with the greatest specificity is taken into account, the -oxy-prepend-content prepends content to that generated by the lesser specificity rules into a new compound content.

Example:

element:before{
    content: "Hello!";
}
element:before{
    -oxy-prepend-content: "said: ";
}
element:before{
    -oxy-prepend-content: "I ";
}

The content shown before the element will be I said: Hello!.

Collapse Text: -oxy-collapse-text Property Value

Used to collapse the content of an element.

Oxygen XML Editor allows you to set the value of the visibility property to -oxy-collapse-text, meaning that the content of that element is not rendered. If an element is marked as -oxy-collapse-text you are not able to position the cursor inside it and edit it. The purpose of -oxy-collapse-text is to make the text value of an element editable only through a form control.

Example: visibility Property

The text value of an XML element will be edited using a text field form control. In this case, the text content is not directly present in the Author visual editing mode:
Cyrillic Counters: -oxy-lower-cyrillic Property Values

Used to style lists with Cyrillic counters.

Oxygen XML Editor allows you to set the value of the list-style-type property to Cyrillic counters. For example, -oxy-lower-cyrillic-ru, -oxy-lower-cyrillic-uk, -oxy-upper-cyrillic-ru or -oxy-upper-cyrillic-uk, meaning that you can have Russian and Ukrainian counters.

Example: Cyrillic Counters

Counting list items with Cyrillic symbols:

```html
li{
  display:list-item;
  list-style-type:-oxy-lower-cyrillic-ru;
}
```

Display Tag Markers: -oxy-display-tags Property

Used to specify whether or not tag markers are displayed.

Oxygen XML Editor allows you to choose whether tag markers of an element should never be presented or the current display mode should be respected. This is especially useful when working with :before and :after pseudo-elements, in which case the element range is already visually defined so the tag markers are redundant.

The property is named -oxy-display-tags, with the following possible values:

- **none** - Tags markers will not be presented regardless of the current display mode (on page 594).
- **default** - The tag markers will be created depending on the current display mode (on page 594).
- **inherit** - The value of the property is inherited from an ancestor element.

-oxy-display-tags
---

Value: none | default | inherit
Initial: default
Applies to: all nodes(comments, elements, CDATA, etc.)
Inherited: false
Media: all

Example: -oxy-display-tags Property

In this example, the para element from DocBook uses a :before and :after element and its tag markers will not be visible.
Editable: -oxy-editable Property

Used to inhibit editing the content of a particular element.

If you want to inhibit the editing of the content of a certain element, you can set the -oxy-editable CSS property to false (the deprecated editable property is also supported).

Floating Toolbar: -oxy-floating-toolbar Property

Used to display a configured floating toolbar in Author mode.

The -oxy-floating-toolbar property is used to configure and display a floating toolbar in Author mode. It accepts a space-separated list of the following functions:

- oxy_button
- oxy_combobox
- oxy_label

Note:
The "|" text value can be used to add a separator between elements of the toolbar.

It must be used in conjunction with the -oxy-selected and -oxy-selection-inside pseudo-classes. The -oxy-selected pseudo-class is automatically set on an element that is fully selected and the -oxy-selection-inside pseudo-class is automatically set on an element that has a selection inside.

Example 1:

p:-oxy-selection-inside {
  -oxy-floating-toolbar:
    oxy_button(actionID, 'bold')
    oxy_button(actionID, 'italic')
This results in a floating toolbar that contains bold, italic, and underline actions presented in Author mode every time text inside a paragraph element is selected.

**Example 2:**

```xml
p:-oxy-selected {
    -oxy-floating-toolbar:
        oxy_label(text, "Platform: ")
        oxy_combobox{
            edit, '@platform',
            editable, false,
            values, 'windows, mac, linux',
            labels, 'Windows, MacOS, Linux'
        }
}
```

This results in a floating toolbar that contains a Platform: label and a drop-down menu used to change the value of the @platform profiling attribute. This is presented in Author mode every time a paragraph element is fully selected.

**Example 3:**

```xml
[conref]:-oxy-selected, [conkeyref]:-oxy-selected {
    -oxy-floating-toolbar:
        oxy_button(actionID, 'add_edit_content_reference')
        oxy_button(actionID, 'remove_content_reference')
        "|"
        oxy_button(actionID, 'conref.replace')
}
```

This results in a floating toolbar that contains Edit Content Reference, Remove Content Reference, and Replace Reference with Content actions presented in Author mode every time an element with a @conref or @conkeyref attribute is fully selected.

**Folding Elements: -oxy-foldable Property**

Used to configure whether or not the content of an element can be expanded or collapsed.

Oxygen XML Editor allows you to declare some elements to be foldable (on page 3219). This is especially useful when working with large documents organized in logical blocks, editing a large DocBook article or book, for instance. Oxygen XML Editor marks the foldable content with a small blue triangle. When you hover with your mouse pointer over this marker, a dotted line borders the collapsible content. The following actions are available in the Folding submenu of the contextual menu:
Toggle Fold

Toggles the state of the current fold.

Collapse Other Folds (Ctrl + NumPad/ (Command + NumPad/ on macOS))

Folds all the elements except the current element.

Collapse Child Folds (Ctrl + NumPad. (Command + NumPad. on macOS))

Folds the elements indented with one level inside the current element.

Expand Child Folds

Unfolds all child elements of the currently selected element.

Expand All (Ctrl + NumPad* (Command + NumPad* on macOS))

Unfolds all elements in the current document.

To define the element whose content can be folded by the user, you must use the property: `oxy-foldable:true;`. To define the elements that are folded by default, use the `oxy-folded:true` property.

Note:

The `oxy-folded` property works in conjunction with the `oxy-foldable` property. Thus, the `folded` property is ignored if the `oxy-foldable` property is not set on the same element.

When collapsing an element, it is useful to keep some of its content visible (for example, a short description of the collapsed region). The `oxy-not-foldable-child` property is used to identify the child element that is kept visible. As its value, it accepts an element name or a list of comma-separated element names. The first occurrence of each child element specified in the list of element names will be identified as the `not-foldable` child and displayed. If the element is marked as `foldable` (`oxy-foldable:true;`) but it doesn't have the `oxy-not-foldable-child` property or none of the specified `non-foldable` children exists, then the element is still `foldable`. In this case the element kept visible when `folded` will be the `before` pseudo-element.

Note:

Deprecated properties `foldable`, `not-foldable-child`, and `folded` are also supported.

Example: Folding DocBook Elements

All the elements below can have a `<title>` child element and are considered to be logical sections. You mark them as being `foldable` leaving the `<title>` element visible.

```
set,
book,
part,
reference,
chapter,
preface,
```
Links: -oxy-link Property

Used to specify that a particular element should be considered a link.

Oxygen XML Editor allows you to declare some elements to be *links*. This is especially useful when working with many documents that reference each other. The links allow for an easy way to get from one document to another. Clicking the link marker will open the referenced resource in an editor.

To define the element that should be considered a link, you must use the -oxy-link property on the :before or :after pseudo-element. The value of the property indicates the location of the linked resource. Since links are usually indicated by the value of an attribute in most cases it will have a value similar to `attr(href)`

**Example: DocBook Link Elements**

The following elements are defined to be links on the :before pseudo-element and their values are defined by the value of an attribute.

```xml
*[href]:before{
  -oxy-link:attr(href);
  content: "Click * attr(href) " for opening" ;
}

ulink[url]:before{
  -oxy-link:attr(url);
  content: "Click to open: " attr(url);
}

olink[targetdoc]:before{
  -oxy-link: attr(targetdoc);
  content: "Click to open: " attr(targetdoc);
}
```
Link Navigation: -oxy-link-activation-trigger Property

Used to specify how hyperlinks are handled in Author mode.

The -oxy-link-activation-trigger property is used to specify when hyperlinks are clickable in Author mode. This is helpful for those who are used to the hyperlink activation procedure in other applications (for example, apps that use Ctrl+Click (Command+Click on macOS) to activate hyperlinks.

The possible values are:

- **click** - Hyperlinks are opened when a user mouse-clicks the link icon or text.
- **modifier-click** - Hyperlinks are opened when a user holds down Ctrl (Command on macOS) and mouse-clicks the link icon or text.
- **auto** - The hyperlink strategy is determined automatically, depending on the context.
- **inherit** - The value is inherited from the parent element.

Morph Elements: -oxy-morph Property Value

Used to specify that an element should be displayed inline.

Oxygen XML Editor allows you to specify that an element has an -oxy-morph display type (deprecated morph property is also supported), meaning that the element is inline (on page 3220) if all its children are inline.

**Example: -oxy-morph Property Value**

Suppose you have a wrapper XML element that allows users to set a number of attributes on all sub-elements. This element should have an inline (on page 3220) or block (on page 3216) behavior, depending on the behavior of its child elements:

```xml
wrapper{
  display:-oxy-morph;
}
```

Placeholders for Empty Elements: -oxy-placeholder-content Property

Used to configure placeholders for empty elements.

Oxygen XML Editor displays the element name as pseudo-content for empty elements if the Show placeholders for empty elements option (on page 176) is selected in the Author preferences page and there is no before or after content set in the CSS for this type of element. There are two CSS properties that can be used to control the placeholders (oxy-placeholder-content and oxy-show-placeholder).

**-oxy-placeholder-content CSS Property**

To control the displayed pseudo-content for empty elements, you can use the -oxy-placeholder-content CSS property.

The following example would change the `<keyword>` element to be displayed as `key`:
Note:
This CSS property accepts the $i18n(key) (on page 331) localization editor variable, as in the following example:

```
-oxy-placeholder-content: "$i18n(id)";
```

---

-oxy-show-placeholder CSS Property

The -oxy-show-placeholder property allows you to decide whether or not the placeholder will be shown. The possible values are:

- **always** - Always display placeholders.
- **default** - Always display placeholders if before or after content is not set in the CSS.
- **inherit** - The placeholders are displayed according to the Show placeholders for empty elements option (on page 176) (if before and after content is not declared).
- **no** - Never display placeholders.

Note:
Deprecated properties show-placeholder and placeholder-content are also supported.

Related information
Using Placeholders in Document Templates (on page 377)

Style Elements: -oxy-style Property

Used to configure the style of particular elements.

Oxygen XML Editor allows you to specify the style for an XML element. This is helpful if you want to embed CSS styling to XML elements directly in the XML file you are editing without having to edit the CSS files that are normally attached to the XML files. The property should have an XPath function for the value.

Example: -oxy-style Property

The following code snippet should be added in the CSS file that renders the files for your framework customization:

```
*{
    -oxy-style: attr(style);
}
```
Suppose you want to display the `<title>` elements in your XML document in the color red. You could add the following snippet directly in the XML document:

```xml
<title style="color:red;">My Memoirs</title>
```

**Tip:**
The `@style` attribute is supported by default in HTML5 documents.

**Tags Color: -oxy-tags-color Property**

Used to configure the background or foreground colors of tags.

By default, Oxygen XML Editor does not display element tags. You can use the **Partial Tags** button from the Author toolbar to control the amount of displayed markup (on page 594).

To configure the default background and foreground colors of the tags, open the Preferences dialog box (Options > Preferences) (on page 124), go to Editor > Edit modes > Author, and set the desired colors in the Tags background color (on page 178) and Tags foreground color (on page 178) options.

If you want to be more specific and configure the colors using a CSS, the `-oxy-tags-background-color` and `-oxy-tags-color` properties allow you to control the background and foreground colors for any particular XML element.

**Example:**

```css
para {
    -oxy-tags-color:white;
    -oxy-tags-background-color:green;
}

title {
    -oxy-tags-color:yellow;
    -oxy-tags-background-color:black;
}
```

**Custom CSS Functions**

Oxygen XML Editor provides a wide range of custom CSS extension functions that can be used to customize the visual Author editing mode.

**Arithmetic Functions**

Arithmetic Functions that are supported.

You can use any of the arithmetic functions implemented in the java.lang.Math class (http://download.oracle.com/javase/6/docs/api/java/lang/Math.html).

In addition, the following functions are available:
## Syntax Details

<table>
<thead>
<tr>
<th>Syntax</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>oxy_add (param1, ..., paramN, 'returnType')</code></td>
<td>Adds the values of all parameters from <code>param1</code> to <code>paramN</code>.</td>
</tr>
<tr>
<td><code>oxy_subtract (param1, param2, ..., paramN, 'returnType')</code></td>
<td>Subtracts the values of parameters <code>param2</code> to <code>paramN</code> from <code>param1</code>.</td>
</tr>
<tr>
<td><code>oxy_multiply (param1, ..., paramN, 'returnType')</code></td>
<td>Multiplies the values of parameters from <code>param1</code> to <code>paramN</code>.</td>
</tr>
<tr>
<td><code>oxy_divide (param1, param2, 'returnType')</code></td>
<td>Performs the division of <code>param1</code> to <code>param2</code>.</td>
</tr>
<tr>
<td><code>oxy_modulo (param1, param2, 'returnType')</code></td>
<td>Returns the reminder of the division of <code>param1</code> to <code>param2</code>.</td>
</tr>
</tbody>
</table>

**Note:**

The `returnType` can be 'integer', 'number', or any of the supported CSS measuring types.

### Example: `oxy_multiply` Function

If you have an image with `width` and `height` specified on it, this will compute the number of pixels on it:

```css
image:before{
  content: "Number of pixels: " oxy_multiply(attr(width), attr(height), "px");
}
```

### Actions: `oxy_action()` Function

This function allows you to define actions directly in the CSS, rather than referencing them from the associated framework.

The `oxy_action()` function is frequently used from the `oxy_button()` function (on page 2379) that provides a graphical button for invoking a custom action. The action is normally defined in the associated Document Type (framework configuration) but the `oxy_action()` function allows you to define it directly in the CSS instead of the framework configuration.

The arguments received by the `oxy_action()` function are a list of properties that define an action. The following properties are supported:

- **name** - The name of the action. It will be displayed as the label for the button or menu item.
- **description** (optional) - A short description with details about the result of the action.
- **icon** (optional) - A path relative to the CSS pointing to an image (the icon for the action). The path can point to resources that are packed in Oxygen XML Editor (`oxygen.jar`) by starting its value with `/` (for example, `/images/Remove16.png`). It can also be expressed using an editor variable (on page 323).
- **operation** - The name of the Java class implementing the `ro.sync.ecss.extensions.api.AuthorOperation` interface. There is also a variety of predefined operations (on page 2152) that can be used.
**Note:**

If the name of the operation specified in the CSS is not qualified (has no Java package name), then it is considered to be one of the built-in Oxygen XML Editor operations from `ro.sync.ecss.extensions.commons.operations` package. If the class is not found in this package, then it will be loaded using the specified name.

- **arg-<string>** - All arguments with the `arg-` prefix are passed to the operation (the string that follows the `arg-` prefix is passed).
- **ID** - (optional) - The ID of the action from the framework. If this is specified, all others parameters are disregarded.

**Example: oxy_action function inside an oxy_button form control (on page 2379):**

```xml
oxy_button(
    action, oxy_action{
        name, 'Insert',
        description, 'Insert an element after the current one',
        icon, url('insert.png'),
        operation, 'InsertFragmentOperation',
        arg-fragment, '<element>${caret}</element>',
        arg-insertLocation, '.',
        arg-insertPosition, 'After'),
    showIcon, true)
```

**Example: oxy_action Function**

You can also create a button form control directly from an `oxy_action` function:

```xml
oxy_action(
    name, 'Insert',
    description, 'Insert an element after the current one',
    operation, 'InsertFragmentOperation',
    arg-fragment, '<element>${caret}</element>',
    arg-insertLocation, '.',
    arg-insertPosition, 'After')
```

**Tip:**

A code template is available to make it easy to add the `oxy_action` function with the Content Completion Assistant (on page 3217) by pressing **Ctrl + Space (Command + Space on macOS)** and select the oxy_action code template.
Action Lists: `oxy_action_list()` Function

This function allows you to define a sequential list of actions directly in the CSS, rather than referencing them from the associated framework.

The `oxy_action_list()` function is used from the `oxy_buttonGroup()` function (on page 2381) that provides a graphical group of buttons with multiple custom actions. These actions are normally defined in the associated Document Type (framework configuration) but the `oxy_action_list()` function allows you to define the actions directly in the CSS instead of the framework configuration.

The arguments received by the `oxy_action_list()` function are a list of actions (executed sequentially) that are defined with the `oxy_action()` function (on page 2354). The following properties are supported in the `oxy_action()` function:

- **name** - The name of the action. The name of the first defined action will be displayed as the label for the button or menu item.
- **description** (optional) - A short description with details about the result of the action. The description of the first defined action will be displayed in a tooltip.
- **icon** (optional) - A path relative to the CSS pointing to an image (the icon for the action). The path can point to resources that are packed in Oxygen XML Editor (`oxygen.jar`) by starting its value with `/` (for example, `/images/Remove16.png`). It can also be expressed using an editor variable (on page 323).
- **operation** - The name of the Java class implementing the `ro.sync.ecss.extensions.api.AuthorOperation` interface. There is also a variety of predefined operations (on page 2152) that can be used.

Note:
If the name of the operation specified in the CSS is not qualified (has no Java package name), then it is considered to be one of the built-in Oxygen XML Editor operations from `ro.sync.ecss.extensions.commons.operations` package. If the class is not found in this package, then it will be loaded using the specified name.

- **arg-string** - All arguments with the `arg-` prefix are passed to the operation (the string that follows the `arg-` prefix is passed).
- **ID** - (optional) - The ID of the action from the framework. If this is specified, all others parameters are disregarded.

Example: `oxy_action_list` Function

```css
p:after {
  content: oxy_buttonGroup(
    label, 'A group of actions',
    icon, url('http://www.oxygenxml.com/img/icn_oxy20.png'),
```
actions,
oxy_action_list{
  oxy_action{
    name, 'Insert a new paragraph',
    description, 'Insert an element after the current one',
    operation, 'InsertFragmentOperation',
    arg-fragment, '<p></p>',
    arg-insertLocation, '.',
    arg-insertPosition, 'After'
  },
  oxy_action{
    name, 'Delete',
    description, 'Deletes the current element',
    operation, 'DeleteElementOperation'
  }
}

Tip:
A code template is available to make it easy to add the oxy_action_list function with the Content Completion Assistant (on page 3217) by pressing Ctrl + Space (Command + Space on macOS) and select the oxy_action_list code template.

Related information
Actions: oxy_action() Function (on page 2354)
Button Group Form Control (on page 2381)

Attributes Concatenation: oxy_attributes() Function
This function concatenates the attributes for an element and returns the serialization.

Syntax:

```
oxy_attributes()
```

Example: oxy_attributes Function

```
element{
  content:oxy_attributes();
}
```

For instance, if you have the following XML fragment: `<element att1="x" xmlns:a="2" x="&quot;"/>`, the CSS function will display:
Base URL: oxy_base-uri() Function

This function evaluates the base URL in the context of the current node.

It does not have any arguments and takes into account the xml:base context of the current node. See the XML Base specification for more details.

Example: oxy_base-uri Function

Suppose you have some image references but you want to see other thumbnail images that reside in the same folder (in Author mode):

```xml
image[href]{
    content: oxy_url(oxy_base-uri(), oxy_replace(attr(href), 
        '.jpeg', 'Thumbnail.jpeg'));
}
```

Capitalization: oxy_capitalize() Function

This function capitalizes the first letter of the text received as argument.

Syntax:

```
oxy_capitalize( text )
```

text

The text in which the first letter will be capitalized.

Example: oxy_capitalize Function

```xml
*:before{
    content: oxy_capitalize(oxy_name()) ": ";
}
```

This would insert the capitalized qualified name as static text content before the element.

Compound Actions: oxy_compound_action() Function

This function allows you to define multiple actions that will be executed sequentially.

The oxy_compound_action() function is used from the oxy_button() form control function (on page 2379) or the oxy_buttonGroup() form control function (on page 2381).

The arguments received by the oxy_compound_action() function are a list of actions (executed sequentially) that are defined with the oxy_action() function (on page 2354).
You can use three optional properties (name, description, icon) in the oxy_compound_action() function to provide labels for the compound action. If you do not specify these three properties, those same properties defined in the first oxy_action function will be used for the labels.

- **name** - The name of the action. It will be displayed as the label for the action. If you want to reuse the name of an action already defined in your framework, you can use the oxy_getActionName function.

- **description** - A short description with details about the result of the action. It will be displayed in a tooltip when hovering over the button linked to this action. If you want to reuse the description of an action already defined in your framework, you can use the oxy_getActionDescription function.

- **icon** - A path relative to the CSS pointing to an image (the icon for the action). The path can point to resources that are packed in Oxygen XML Editor (oxygen.jar) by starting its value with / (for example, /images/Remove16.png). It can also be expressed as an editor variable (on page 323). If you want to reuse the icon of an action already defined in your framework, you can use the oxy_getActionIcon function.

The oxy_getActionName, oxy_getActionDescription, and oxy_getActionIcon functions accept the following 2 parameters:

- **framework.defined.action.id** (required) - The ID of an action defined in the current framework that gets the name, description, or icon for that action.

- **fallback** (optional) - A fallback value in case the ID value provided in the framework.defined.action.id parameter is not found.

**Example: oxy_compound_action Function**

```java
oxy_button(
    action,
    oxy_compound_action(
        name, oxy_getActionName('framework.id', 'Fallback'),
        description, 'Inserts a paragraph and uses form controls to edit its @audience attribute',
        icon, url('http://www.oxygenxml.com/img/icn_oxy20.png'),
        oxy_action(
            name, 'Insert',
            description, 'Insert an element after the current one',
            operation, 'ro.sync.ecss.extensions.commons.operations.InsertFragmentOperation',
            icon, url('insert.png'),
            arg_fragment, '<p audience=''></p>','
            arg_insertLocation, '.',
            arg_insertPosition, 'After'
        ),
        oxy_action(
            name, 'Activate edit mode',
            description, 'Sets a pseudo class that will activate a CSS rule that will present a text field form control for the @audience attribute',
```
operation, 'SetPseudoClassOperation',
arg-name, 'edit-mode-on',
arg-elementLocation, '. '
)

, showIcon, true)

Tip:
A code template is available to make it easy to add the oxy_compound_action function with the Content Completion Assistant (on page 3217) by pressing Ctrl + Space (Command + Space on macOS) and select the oxy_action_list code template.

Related information
Actions: oxy_action() Function (on page 2354)
Button Form Control (on page 2379)

Concatenation: oxy_concat() Function
This function concatenates the received string arguments.

Syntax:

```
oxy_concat (str_1, str_2)
```

str_1 ... str_n

The string arguments to be concatenated.

Example: oxy_concat Function
If an XML element has a @padding-left attribute:

```
<p padding-left="20">...</n```

and you want to add a padding before it with that specific amount specified in the attribute value:

```
*p [padding-left] {
  padding-left: oxy_concat(attr(padding-left), "px");
}
```

Get Text: oxy_getSomeText(text, length) Function
This function allows you to truncate a long string and to set a maximum number of displayed characters.

Syntax:
oxy_getSomeText (text, length, endsWithPoints)

text
Displays the actual text.

length
Sets the maximum number of characters that are displayed.

endsWithPoints
Specifies if the truncated text ends with ellipsis.

Example: oxy_getSomeText Function

If an attribute value is very large, you can trim its content before it is displayed as static content:

```xml
*{longdesc}:before{
    content: oxy_getSomeText(attr(longdesc), 200);
}
```

Indexing: oxy_indexof() Function

This function is used to define searches.

The oxy_indexof() function has two signatures:

Syntax 1:

```
oxy_indexof (text, toFind)
```

Returns the index within text string of the first occurrence of the toFind substring.

- text
  Text to search in.

- toFind
  The searched substring.

Syntax 2:

```
oxy_indexof (text, toFind, fromOffset)
```

Returns the index within text string of the first occurrence of the toFind substring. The search starts from fromOffset index.

- text
  Text to search in.

- toFind
  The searched substring.
The searched substring.

fromOffset

The index to start the search from.

**Example: oxy_indexof Function**

```
oxy_indexof('abcd', 'bc') returns 1.
oxy_indexof('abcdabc', 'bc', 2) returns 4.
```

If you only want to display part of an attribute value, for instance the part that comes before an **Appendix** string:

```
image[longdesc] {
    content: oxy_substring(attr(longdesc), 0,
        oxy_indexof(attr(longdesc), "Appendix"));
}
```

**Label: oxy_label() Function**

This function can be used in conjunction with the CSS `content` property to change the style of generated text.

The arguments of the function are `property name - property value` pairs. The following properties are supported:

- **text** - This property specifies the built-in form control you are using.
- **width** - Specifies the width of the content area using relative (em, ex), absolute (in, cm, mm, pt, pc, px), and percentage (followed by the % character) length units. The `width` property takes precedence over the `columns` property (if the two are used together).
- **color** - Specifies the foreground color of the form control. If the value of the `color` property is `inherit`, the form control has the same color as the element that was used to insert it.
- **background-color** - Specifies the background color of the form control. If the value of the `background-color` property is `inherit`, the form control has the same color as the element that was used to insert it.
- **styles** - Specifies styles for the form control. The values of this property are a set of CSS properties:
  - `font-weight`, `font-size`, `font-style`, `font`
  - `text-align`, `text-decoration`
  - `width`
  - `color`, `background-color`
  - `link` - For more information about this property, see the **link property section** *(on page 2350).*

```
element{
    content: oxy_label(text, "Label Text", styles,
        "font-size:2em;color:red;link:attr(href);"));
}
```
Instead of using the values of the `styles` property individually, you can define them in a CSS file as in the following example:

```css
* {
    width: 40%;
    text-align: center;
}
```

Then refer that file with an `@import` directive, as follows:

```css
elem {
    content: oxy_label(text, 'my_label', styles, "@import 'labels.css';")
}
```

**CAUTION:**
Extensive use of the `styles` property may lead to performance issues.

If the text from an `oxy_label()` function contains new lines, for example `oxy_label(text, 'LINE1\A LINE2', width, 100px)`, the text is split in two. Each of the two new lines has the specified width of 100 pixels.

**Note:**
The text is split after `\A`, which represents a new line character.

You can use the `oxy_label()` function together with a built-in form control (on page 2373) function to create a form control based layouts.

**Example: oxy_label Function**

An example of a use case is if you have multiple attributes on a single element and you want use form controls on separate lines and style them differently. Consider the following CSS rule:

```css
person:before {
    content: "Name:*" oxy_textfield(edit, '@name', columns, 20)
    "\A Address:" oxy_textfield(edit, '@address', columns, 20)
}
```

Suppose you only want the *Name* label to be set to **bold**, while you want both labels aligned to look like a table (the first column with labels and the second with a text field). To achieve this, you can use the `oxy_label()` to style each label differently.

```css
person:before {
    content: oxy_label(text, "Name:*", styles, "font-weight:bold;width:200px")
    oxy_textfield(edit, '@name', columns, 20) "\A *
    oxy_label(text, "Address:", styles, "width:200px")
}
oxy_textfield(edit, '@address', columns, 20)
}

Tip:
A code template is available to make it easy to add the oxy_label function with the Content Completion Assistant (on page 3217) by pressing Ctrl + Space (Command + Space on macOS) and select the oxy_label code template.

Last Occurrence: oxy_lastindexOf() Function

This function is used to define last occurrence searches.

The oxy_lastindexOf() function has two signatures:

Syntax 1:

```latex
oxy_lastindexOf(text, toFind)
```

Returns the index within text string of the rightmost occurrence of the toFind substring.

- **text**
  - Text to search in.

- **toFind**
  - The searched substring.

Syntax 2:

```latex
oxy_lastindexOf(text, toFind, fromOffset)
```

The search starts from fromOffset index. Returns the index within text string of the last occurrence of the toFind substring, searching backwards starting from the fromOffset index.

- **text**
  - Text to search in.

- **toFind**
  - The searched substring.

- **fromOffset**
  - The index to start the search backwards from.

Example: oxy_lastindexOf Function

oxy_lastindexOf('abcdbc', 'bc') returns 4.

oxy_lastindexOf('abcdbccbc', 'bc', 2) returns 1.
If you only want to display part of an attribute value, for instance the part that comes before an Appendix string:

```xml
image[longdesc] {
  content: oxy_substring(attr(longdesc), 0,
                   oxy_lastindexof(attr(longdesc), "Appendix"));
}
```

**Link Text: oxy_link-text() Function**

You can use this function on the CSS `content` property to obtain a text description from the source of a reference.

By default, the `oxy_link-text()` function resolves DITA and DocBook references. For further details about how you can also extend this functionality to other frameworks (on page 3219), go to Configuring an Extensions Bundle (on page 2232).

**DITA Support**

For DITA, the `oxy_link-text()` function resolves the `<xref>` element and the elements that have a `@keyref` attribute. The text description is the same as the one presented in the final output for those elements. If you use this function for a `<topicref>` element that has the `@navtitle` and `@locktitle` attributes set, the function returns the value of the `@navtitle` attribute.

**DocBook Support**

For DocBook, the `oxy_link-text()` function resolves the `<xref>` element that defines a link in the same document. The text description is the same as the one presented in the final output for those elements.

**Example: oxy_link-text Function**

For the following XML and associated CSS fragments the `oxy_link-text()` function is resolved to the value of the `@xreflabel` attribute.

```xml
<para><code id="para.id" xreflabel="The reference label">my code</code></para>

<para><xref linkend="para.id"></xref></para>
```

```css
xref {
  content: oxy_link-text();
}
```

If the text from the target cannot be extracted (for instance, if the `@href` is not valid), you can use an optional argument to display fallback text.

```css
*[class="map/topicref"]::before{
  content: oxy_link-text("Cannot find the topic reference");
  link: attr(href);
}
```
Local Name: oxy_local-name() Function

This function evaluates the local name of the current node.

It does not have any arguments.

Example: oxy_local-name Function

To insert the local name as static text content before the element, use this CSS selector:

```css
*:before{
    content: oxy_local-name() "\": ";
}
```

Lowercase: oxy_lowercase() Function

This function transforms the text received as argument to lower case.

Syntax:

```
oxy_lowercase( text )
```

- `text` The text to be lower cased.

Example: oxy_lowercase Function

To insert a lower-cased qualified name as static text content before the element, use this CSS selector:

```css
*:before{
    content: oxy_lowercase(oxy_name()) ": ";
}
```

Name: oxy_name() Function

This function evaluates the qualified name of the current node.

It does not have any arguments.

Example: oxy_name Function

To insert a qualified name as static text content before the element, use this CSS selector:

```css
*:before{
    content: oxy_name() ": ";
}
```

Parent URL: oxy_parent-url() Function

This function evaluates the parent URL of a URL received as string.
Syntax:

```
oxy_parent-url ( URL )
```

**URL**

The URL as string.

**Replace: oxy_replace() Function**

This function is used to replace a string of text.

The `oxy_replace()` function has two signatures:

**Syntax 1:**

```
oxyl_replace ( text, target, replacement )
```

This function replaces each substring of the text that matches the literal target string with the specified literal replacement string.

- **text**
  
  The text in which the replace will occur.

- **target**
  
  The target string to be replaced.

- **replacement**
  
  The string replacement.

**Example:** Suppose that you have image references but you want to see other thumbnail images that reside in the same folder in the visual Author editing mode:

```javascript
image[href] {
    content: oxy_url(oxy_base-uri(), oxy_replace(attr(href), 
                  '.jpeg', 'Thumbnail.jpeg'));
}
```

**Syntax 2:**

```
oxyl_replace ( text, target, replacement, isRegExp )
```

This function replaces each substring of the text that matches the target string with the specified replacement string.

- **text**
  
  The text in which the replace will occur.

- **target**
The target string to be replaced.

replacement

The string replacement.

isRegExp

If true the target and replacement arguments are considered regular expressions, if false they are considered literal strings.

Example: Suppose that you want to use a regular expression to replace all space sequences with an underscore:

```xml
image[title]{
  content:oxy_replace(attr(title), "\s+", "_", true)
}
```

Substring of Text: oxy_substring() Function

This function is used to return a string of text.

The oxy_substring() function has two signatures:

Syntax 1:

```xml
oxy_substring ( text , startOffset )
```

Returns a new string that is a substring of the original text string. It begins with the character at the specified index and extends to the end of text string.

- text
  - The original string.

- startOffset
  - The beginning index, inclusive

Syntax 2:

```xml
substring ( text , startOffset , endOffset )
```

Returns a new string that is a substring of the original text string. The substring begins at the specified startOffset and extends to the character at index endOffset - 1.

- text
  - The original string.

- startOffset
  - The beginning index, inclusive.

- endOffset
  - The end index, exclusive.
The ending index, exclusive.

**Example: oxy_substring Function**

oxy_substring('abcd', 1) returns the string 'bcd'.

oxy_substring('abcd', 4) returns an empty string.

oxy_substring('abcd', 1, 3) returns the string 'bc'.

If you only want to display part of an attribute value, for instance the part that comes before an Appendix string:

```xml
image[longdesc]{
    content: oxy_substring(attr(longdesc), 0, oxy_indexof(attr(longdesc), "Appendix"));
}
```

**Unescape URL Value: oxy_unescapeURLValue(string) Function**

This function returns the unescaped value of a URL-like string given as a parameter.

For example, if the value contains `%20` it will be converted to a simple space character.

**Example: oxy_unescapeURLValue Function**

oxy_unescapeURLValue("http://www.example.com/a%20simple%20example.html") returns the following value:

http://www.example.com/a simple example.html

**Unparsed Entity URI: oxy_unparsed-entity-uri() Function**

This function returns the URI value of an unparsed entity name.

**Syntax:**

```xml
oxy_unparsed-entity-uri (unparsedEntityName)
```

unparsedEntityName

The name of an unparsed entity defined in the DTD.

This function can be useful to display images that are referenced with unparsed entity names.

**Example: oxy_unparsed-entity-uri Function**

CSS for displaying the image in Author for an imagedata with entityref to an unparsed entity:

```xml
imagedata[entityref]{
    content: oxy_url(oxy_unparsed-entity-uri(attr(entityref)));
}
```
### Uppercase: oxy_uppercase() Function

This function transforms the text received as argument to upper case.

**Syntax:**

```plaintext
oxy_uppercase ( text )
```

- **text**
  
  The text to be capitalized.

**Example: oxy_uppercase Function**

To insert the upper-cased qualified name as static text content before the element, use this CSS selector:

```css
*:before{
    content: oxy_uppercase(oxy_name()) "": ";
}
```

### URL: oxy_url() Function

This function extends the standard CSS `url()` function by allowing you to specify additional relative path components (parameters `loc_1` to `loc_n`).

Oxygen XML Editor uses all these parameters to construct an absolute location. Note that any of the parameters that are passed to the function can be either relative or absolute locations. These locations can be expressed as String objects, functions, or editor variables (on page 323) (built-in or custom).

**Syntax:**

```plaintext
oxy_url ( base_location , loc_1 , loc_2 )
```

- **base_location**
  
  String representing the base location. If not absolute, will be solved relative to the CSS file URL.

- **loc_1 ... loc_n (optional)**
  
  Strings representing relative location path components.

**Examples: oxy_url Function**

The following function receives String objects as input parameters:

```plaintext
```

and returns:

`'http://www.oxygenxml.com/dir1/dir4/dir5/test.xml'`
The following function receives the result of the evaluation of two other functions as parameters (for instance, this is useful if you have image references and you want to see thumbnail images stored in the same folder):

```javascript
image[href] {
    content: oxy_url(oxy_base-uri(), oxy_replace(attr(href), 
        '.jpeg', 'Thumbnail.jpeg'));
}
```

The following function uses an editor variable (on page 323) as the first parameter to point to the Oxygen XML Editor installation location:

```javascript
image[href] {
    content: oxy_url('${oxygenHome}', 'logo.png');
}
```

Related information

Editor Variables (on page 323)

**XPath: oxy_xpath() Function**

This function is used to evaluate XPath expressions.

**Syntax:**

```
oxy_xpath ( XPathExpression [, processChangeMarkers , value ] [, evaluate , value ])
```

It evaluates the given XPath 2.0 expression using Saxon 9 and returns the result. XPath expressions that depend on the cursor location can be successfully evaluated only when the cursor is located in the actual XML content. Evaluation fails when the current editing context is inside a referenced `xi:include` section or inside artificially referenced content (for example, DITA `@conref` or `@topicref` references).

The parameters of the function are as follows:

- A required `expression` parameter, which is the XPath expression to be evaluated.
- An optional `processChangeMarkers` parameter, followed by its value, which can be either `true` or `false` (default value). When you set the parameter to `true`, the function returns the resulting text with all the change markers accepted (`delete` changes are removed and `insert` changes are preserved).
- An optional `evaluate` parameter, followed by its value, which can be one of the following:
  - `dynamic`- Evaluates the XPath each time there are changes in the document.
  - `dynamic-once` - Separately evaluates the XPath for each node that matches the CSS selector. It will not re-evaluate the expression when changes are made to other
nodes in the document. This will lead to improved performance, but the displayed content may not be updated to reflect the actual document content.

- **static** - If the same XPath is evaluated on several nodes, the result for the first evaluation will be used for all other matches. Use this only if the XPath does not contain a relationship with the node on which the CSS property is evaluated. This will lead to improved performance, but the static displayed content may not be updated to reflect the actual document content.

**Note:**

When XPath expressions are evaluated, the entities and `<xi:include>` elements are replaced with the actual content that is referenced. For example, consider the following code snippet:

```xml
<article>
    <xi:include href="section1.xml" xmlns:xi="http://www.w3.org/2001/XInclude"/>
</article>
```

where `section1.xml` contains the following content:

```xml
<section>
    <p>Referenced content</p>
</section>
```

The XPath expression will be executed over the actual content in the `section1.xml` file.

**Example: oxy_xpath Function**

The following example counts the number of words from a paragraph (including tracked changes (on page 3224)) and displays the result in front of it:

```xml
para:before{
    content:
    concat("|Number of words:|
    oxy_xpath{
        "count(tokenize(normalize-space(string-join(text(), ' ')), ' '))",
        processChangeMarkers,
        true),
    "| "
    }
}
```

**Note:**

The `oxy_xpath()` function supports editor variables (on page 323), as in the following example:
Form Controls

Oxygen XML Editor provides a variety of built-in form controls that allow users to interact with documents with familiar user interface objects. These form controls are defined in CSS stylesheets that are used to render Author mode. For customization purposes, Oxygen XML Editor also supports custom form controls in Java (on page 2404).

How to Add a Built-in Form Control in Author Mode

Form controls can be added by defining them in the CSS associated with the XML document.

1. Create a custom CSS file.
2. Define the form control in the CSS using its dedicated CSS function. For example, to add a date picker form control (on page 2390), its dedicated function is oxy_datePicker.
3. Associate the CSS file with the XML document in one of the following ways:
   - If you have a framework (document type) already created for this XML vocabulary, create a CSS in the framework directory and associate the CSS with the framework (on page 2145). This approach is recommended if you intend on sharing the customization with others.
   - If you do not have a framework, you can associate the CSS to the XML document through a Processing Instruction (on page 2307).

For more information about form controls, watch our video demonstration:

https://www.youtube.com/embed/-WY3wzkMSLM
Audio File Player Form Control

The oxy_audio built-in form control is used for providing a mechanism to play audio clips.

The oxy_audio form control supports the following properties:

- **href**: The absolute or relative location of a resource. This property is mandatory. Relative values are resolved relative to the CSS. If you have media resources relative to the XML document, you can specify their paths like this:

  ```css
  oxy_audio(href, oxy_url(oxy_base-uri(), 'ex.mp3'), width, 400px)
  ```

- **width**: Specifies the width of the content area using relative (em, ex), absolute (in, cm, mm, pt, pc, px), and percentage (followed by the % character) length units.

Example: oxy_audio Form Control

```xml
object {
  content:
    oxy_audio(
      href, 'resources/audio.mp3',
      width, 200px),
}
```

Tip:

To insert a sample of the oxy_audio form control in a CSS file (or LESS file), invoke the Content Completion Assistant (on page 3217) by pressing Ctrl + Space (Command + Space on macOS) and select the oxy_audio code template.

To see more detailed examples and more information about how form controls work in Oxygen XML Editor, see the sample files in the following directory: `[OXYGEN_INSTALL_DIR]/samples/form-controls`.

For more information about form controls, watch our video demonstration:

https://www.youtube.com/embed/-WY3wzkMSLM
Browser Form Control

The oxy_browser built-in form control is used for providing a mechanism to integrate HTML frames or interact with SVG documents directly in the Author mode editor. It can also be used to load HTML that executes JavaScript and from that JavaScript you can access the Oxygen XML Editor workspace.

The oxy_browser form control supports the following properties:

- **href** - The absolute or relative location of a resource. This property is mandatory. Relative values are resolved relative to the CSS. If you have media resources relative to the XML document, you can specify their paths like this:

  ```
  oxy_browser(href, oxy_url(oxy_base-uri(), 'ex.svg'), width, 50%, height, 50%)
  ```

- **width** - Specifies the width of the content area using relative (em, ex), absolute (in, cm, mm, pt, pc, px), and percentage (followed by the % character) length units.

- **height** - Specifies the height of the form control area using relative (em, ex), absolute (in, cm, mm, pt, pc, px), and percentage (followed by the % character) length units.

**Example: oxy_browser Form Control**

```javascript
object {
  content:
  oxy_browser{
    href, 'http://example.page',
    width, 600px,
    height, 400px,
  }
}
```

**Tip:**
To insert a sample of the oxy_browser form control in a CSS file (or LESS file), invoke the **Content Completion Assistant (on page 3217)** by pressing **Ctrl + Space (Command + Space on macOS)** and select the oxy_browser code template.

To see more detailed examples and more information about how form controls work in Oxygen XML Editor, see the sample files in the following directory: `[OXYGEN_INSTALL_DIR]/samples/form-controls`.
Interacting with the Oxygen XML Editor Workspace

The oxy_browser form control also provides the possibility of creating custom form control without having to use the Java-based API. You can use the oxy_browser form control to load HTML that executes JavaScript. In the JavaScript, you can use some predefined global variables that provide a gateway between the JavaScript and the Oxygen XML Editor Java API. This allows you to perform changes in the document, open resources, and more, solely from the JavaScript.

**Important:**
This will only work if the loaded HTML is located inside a framework or plugin directory (on page 139), such as: \[OXYGEN_INSTALL_DIR\]/frameworks/ or \[OXYGEN_INSTALL_DIR\]/plugins/.

The following global variables can be used:

- **authorAccess** - This object is an instance of ro.sync.ecss.extensions.api.AuthorAccess.
- **contextElement** - An instance of ro.sync.ecss.extensions.api.node.AuthorNode. The form control is added over this node.
- **fcArguments** - A java.util.Map implementation with the properties (name and value pairs) passed on the form control function.
- **apiHelper** - A helper object for creating Java objects. It allows you to create Java objects from within the JavaScript code. These objects can then be passed to the Java methods as in the following example:

```javascript
var newAttrValue = apiHelper.newInstance("ro.sync.ecss.extensions.api.node.AttrValue",
   ["normalizedValue", "rawValue", true]);
authorAccess.getDocumentController().setAttribute("counter", newAttrValue, contextElement);
...
```

You can also specify the constructor signature:

```javascript
var newAttrValue = apiHelper.newInstance("ro.sync.ecss.extensions.api.node.AttrValue
   (java.lang.String, java.lang.String, boolean)",
   ["normalizedValue", "rawValue", true]);
authorAccess.getDocumentController().setAttribute("counter", newAttrValue, contextElement);
...
```

For more information, open the form-controls.xml file in the [OXYGEN_INSTALL_DIR]/samples/form-controls directory and go to section 11.1 - Interacting with the Oxygen Workspace.
Warning:
On macOS, you need to use asynchronous calls to the API, due to the following JDK bug: https://bugs.openjdk.java.net/browse/JDK-8087465. By default, the API is called synchronously, but you can change this behavior for each API object by calling two methods: `sync()` and `async()`.

```javascript
// By default, the methods are invoke synchronously.
var ctrl = authorAccess.getDocumentController();
try {
  // On Mac, methods that change the document must be executed asynchronously.
  ctrl.async();
  ctrl.setAttribute("counter", newAttrValue, contextElement);
} finally {
  ctrl.sync();
}
```

Listening for Changes in the Document

If the form control presents some information from the document (for example, the value of an attribute), then it needs to be notified on changes in the document so that it can update that information. To do this, follow these steps:

1. In the JavaScript, the `bridgeReady()` method is invoked as soon as the form control is loaded and the API bridge is installed. This is where you can add a listener:

```javascript
function bridgeReady () {

  // We declare a member function for each method of the
  // ro.sync.ecss.extensions.api.AuthorListener interface (same function signature)
  var handler = {
    attributeChanged : function(event) {
      var node = event.getOwnerAuthorNode();
      var attrName = event.getAttributeName();

      if (node.equals(contextElement) && attrName === "counter") {
        init();
      }
    },
    contentDeleted : function(event) {},
    contentInserted : function(event) {}
  };

  // We create a proxy over an ro.sync.ecss.extensions.api.AuthorListener that will
  // delegate its methods to the JS object's functions.
  // We assign the listener to a global variable so that we can remove it later on,
```
2. Since a listener was added on the document, it is important to remove it once the form control is not used anymore. When a form control is discarded, the `dispose()` JavaScript function is invoked, so if you have any cleanup to do, make sure you define a function with this name and remove any previously created listeners in it.

```javascript
// on the dispose() method.
authorDocumentListener = apiHelper.createProxyListener(
    "ro.sync.ecss.extensions.api.AuthorListener", handler);

var ctrl = authorAccess.getDocumentController();

// Add the proxy listener.
ctrl.addAuthorListener(authorDocumentListener);
}

/**
 * The form control will not be used anymore. Clean up.
 */
function dispose() {
    // Dispose all added listeners.
    ctrl = authorAccess.getDocumentController();
    ctrl.removeAuthorListener(authorDocumentListener);
}
```

**Debugging JavaScript Used for Custom Form Controls**

If you encounter unexpected results when using the method described above (on page 2376), you can debug the script by using the following guidelines:

- Calls to `alert("message.to.present")` or `console.log("message.to.present")` will be presented in the Results panel (on page 548).
- You can install the Firebug extension by executing the following script:

```javascript
function installFB() {
  if (!document.getElementById('FirebugLite')) {
    E = document['createElement' + 'NS'] && document.documentElement.namespaceURI;
    E = E ? document['createElement' + 'NS'](E, 'script')
    : document['createElement']['script'];
    E['setAttribute']['id', 'FirebugLite'];
    E['setAttribute']['src',
    'https://getfirebug.com/' + 'firebug-lite.js' + '#startOpened']);
    E['setAttribute']['FirebugLite', '4'];
    (document['getElementsByTagName']['head'][0]
    || document['getElementsByTagName']['body'][0]).appendChild(E);
```
E = new Image;
E['setAttribute']('src', 'https://getfirebug.com/' + '#startOpened');
}
}
}

Note:
To force the Browser Form Control to reload after making changes to the JavaScript file, you need to use the Reload page action from the form control's contextual menu.

For more information about form controls, watch our video demonstration:

https://www.youtube.com/embed/-WY3wzkMSLM

Related information
Custom CSS Functions (on page 2353)
URL: oxy_url() Function (on page 2370)

Button Form Control
The oxy_button built-in form control is used for graphical user interface objects that invoke a custom Author mode action (defined in the associated Document Type) referencing it by its ID, or directly in the CSS.

The oxy_button form control supports the following properties:

- **fontInherit** - This value specifies whether or not the form control inherits its font from its parent element. The values of this property can be true or false (default value). To make the form control inherit its font from its parent element, set the fontInherit property to true.
- **color** - Specifies the foreground color of the form control. If the value of the color property is inherit, the form control has the same color as the element that was used to insert it.
- **visible** - Specifies whether or not the form control is visible. The possible values of this property are true (default value) and false.
- **transparent** - Flattens the aspect of the button form control, removing its border and background. The values of this property can be true or false (default value).
- **showText** - Specifies if the action text should be displayed on the button form control. If this property is missing then the button displays the icon only if it is available, or the text if the icon is not available. The values of this property can be true or false.

```xml
element {
  content: oxy_button(actionID, 'remove.attribute', showText, true);
}
```

- **showIcon** - Specifies if the action icon should be displayed on the button form control. If this property is missing then the button displays the icon only if it is available, or the text if the icon is not available. The values of this property can be true or false.
element {
  content: oxy_button(actionID, 'remove.attribute', showIcon, true);
}

- **enableInReadOnlyContext** - To enable button form controls (on page 2379) or groups of buttons form controls (on page 2381) this property needs to be set to true. This property can be used to specify areas as read-only (by setting the oxy-editable property to false). This is useful when you want to use an action that does not modify the context.

- **hoverPseudoclassName** - Allows you to change the way an element is rendered when you hover over a form control. The value is the name of a CSS pseudo-class. When you hover over the form control, the specified pseudo-class will be set on the element that contains the form control.

```html
p:before {
  content: oxy_button(hoverPseudoclassName, 'showBorder')
}
p:showBorder {
  border: 1px solid red;
}
```

- **actionContext** - Specifies the context that the action associated with the form control is executed. Its possible values are **element** (default value) and **caret**. If you select the **element** value, the context is the element that holds the form control. If you select the **caret** value, the action is invoked at the cursor location. If the cursor is not inside the element that holds the form control, the **element** value is selected automatically.

- **actionID** - The ID of the action, specified in the document type association (on page 148), that is invoked when you click the button.

**Note:**
The element that contains the form control represents the context where the action is invoked.

- **action** - Defines an action directly, rather than using the actionID parameter to reference an action from the document type association (on page 148). This property is defined using the oxy_action function (on page 2354).

**Tip:**
You can also create a button form control directly from an oxy_action function (on page 2355).
Tip:
To execute multiple actions sequentially, you can use the `oxy_compound_action` function (on page 2358).

Example: `oxy_button` Form Control

```css
button:before {
  content: "Label:"
  oxy_button{
    /* This action is declared in the document type associated with the XML document. */
    actionID, "insert.popupWithMultipleSelection";
  }
}
```

Tip:
To insert a sample of the `oxy_button` form control in a CSS file (or LESS file), invoke the `Content Completion Assistant (on page 3217)` by pressing Ctrl + Space (Command + Space on macOS) and select the `oxy_button` code template. Also, an `oxy_button_in_place_action` code template is available that inserts an `oxy_button` function that includes an `action` parameter.

To see more detailed examples and more information about how form controls work in Oxygen XML Editor, see the sample files in the following directory: `[OXYGEN_INSTALL_DIR]/samples/form-controls`

For more information about form controls, watch our video demonstration:

https://www.youtube.com/embed/-WY3wzkMSLM

Related information
- Custom CSS Functions (on page 2353)
- Actions: `oxy_action()` Function (on page 2354)

**Button Group Form Control**

The `oxy_buttonGroup` built-in form control is used for a graphical user interface group of buttons that invokes one of several custom Author mode actions (defined in the associated Document Type) referencing it by its ID, or directly in the CSS.

The `oxy_buttonGroup` form control supports the following properties:
- **label** - Specifies the label to be displayed on the button. This label can be translated using the \$\{i18n\} editor variable (on page 331).
- **icon** - The path to the icon to be displayed on the button.
- **visible** - Specifies whether or not the form control is visible. The possible values of this property are true (default value) and false.
- **tooltip** - Specifies a tooltip to be displayed when you hover over the form control.
- **transparent** - Makes the button transparent without any borders or background colors. The values of this property can be true or false.
- **fontInherit** - This value specifies whether or not the form control inherits its font from its parent element. The values of this property can be true or false (default value). To make the form control inherit its font from its parent element, set the fontInherit property to true.
- **enableInReadOnlyContext** - To enable button form controls (on page 2379) or groups of buttons form controls (on page 2381) this property needs to be set to true. This property can be used to specify areas as read-only (by setting the -oxy-editable property to false). This is useful when you want to use an action that does not modify the context.
- **hoverPseudoclassName** - Allows you to change the way an element is rendered when you hover over a form control. The value is the name of a CSS pseudo-class. When you hover over the form control, the specified pseudo-class will be set on the element that contains the form control.

```html
p:before {
    content: oxy_buttonGroup(hoverPseudoclassName, 'showBorder')
}
p:showBorder {
    border: 1px solid red;
}
```

- **actionIDs** - The IDs of the actions that will be presented in the group of buttons.
- **actionID** - The ID of the action, specified in the document type association (on page 148), that is invoked when you click the button.

---

**Note:**

The element that contains the form control represents the context where the action is invoked.

- **actions** - Defines a sequential list of actions directly, rather than using the actionID parameter to reference actions from the document type association (on page 148). This property is defined using the oxy_action_list function (on page 2356).

```javascript
oxy_buttonGroup(
    label, 'A group of actions',
    icon, url('http://www.oxygenxml.com/img/icn_oxy20.png'),
    actions,
    oxy_action_list(
        oxy_action(
            name, 'Insert',
```
Tip:
To execute multiple actions sequentially, you can use the `oxy_compound_action` function (on page 2358) in the `oxy_action_list` function (on page 2356).

```xml
p:before {
    content:
    oxy_buttonGroup(
        label, 'A group of actions',
        icon, url('http://www.oxygenxml.com/img/icn_oxy20.png'),
        actions,
        oxy_action_list(
            oxy_compound_action(
                name, oxy_getActionName('framework.id', 'Fallback'),
                description, 'Inserts a paragraph and uses form controls to edit its @audience attribute',
                icon, url('http://www.oxygenxml.com/img/icn_oxy20.png'),
                oxy_action(
                    name, 'Insert',
                    description, 'Insert an element after the current one',
                    operation, 'ro.sync.ecss.extensions.commons.operations.InsertFragmentOperation',
                    icon, url('insert.png'),
                    arg-fragment, '<p audience='''></p>',
                    arg-insertLocation, '.',
                    arg-insertPosition, 'After'
                )
            )
        )
    )
}
```
description, 'Sets a pseudo class that will activate a CSS rule that will present a text field form control for the @audience attribute',
operation, 'SetPseudoClassOperation',
arg-name, 'edit-mode-on',
arg-elementLocation, '.',
}
},
 oxy_action{
 name, 'Delete',
description, 'Deletes the current element',
operation, 'DeleteElementOperation'
}
)
)
)
)
}

• **actionContext** - Specifies the context that the action associated with the form control is executed. Its possible values are `element` (default value) and `caret`. If you select the `element` value, the context is the element that holds the form control. If you select the `caret` value, the action is invoked at the cursor location. If the cursor is not inside the element that holds the form control, the `element` value is selected automatically.

• **actionStyle** - Specifies what to display for an action in the form control. The values of this property can be `text` (default value), `icon`, or `both`.

**Example: oxy_buttonGroup Form Control**

```css
t buttongroup:before {
    content:
 oxy_label(text, "Button Group:", width, 150px, text-align, left)
 oxy_buttonGroup(
    label, 'A group of actions',
    /* The action IDs are declared in the document type associated with the XML document. */
    actionIDs,
    "insert.popupWithMultipleSelection,insert.popupWithSingleSelection",
    actionStyle, "both")
}
```

**Tip:**
To insert a sample of the oxy_buttonGroup form control in a CSS file (or LESS file), invoke the **Content Completion Assistant (on page 3217)** by pressing Ctrl + Space (Command + Space on macOS) and select the oxy_buttonGroup code template. Also, an oxy_buttonGroup_in_place_action code
A template is available that inserts an `oxy_buttonGroup` function that includes an `oxy_action_list` function.

To see more detailed examples and more information about how form controls work in Oxygen XML Editor, see the sample files in the following directory: `{OXYGEN_INSTALL_DIR}/samples/form-controls`.

For more information about form controls, watch our video demonstration:

https://www.youtube.com/embed/-WY3wzkMSLM

Related information

- Custom CSS Functions ([on page 2353])
- Actions: `oxy_action()` Function ([on page 2354])
- Action Lists: `oxy_action_list()` Function ([on page 2356])
- Compound Actions: `oxy_compound_action()` Function ([on page 2358])
- Label: `oxy_label()` Function ([on page 2362])

**Checkbox Form Control**

The `oxy_checkbox` built-in form control is used for a graphical user interface box that you can click to enable or disable an option. A single checkbox or multiple checkboxes can be used to present and edit the value on an attribute or element.

The `oxy_checkbox` form control supports the following properties:

- **edit** - Lets you edit the value of an attribute, the text content of an element, or Processing Instructions (PI). This property can have the following values:
  - `@attribute_name` - The name of the attribute whose value is being edited. If the attribute is in a namespace, the value of the property must be a `QName` ([on page 3223]) and the CSS must have a namespace declaration for the prefix.
  - `#text` - Specifies that the presented/edited value is the simple text value of an element.

  **Note:** You can set the value of the `visibility` property to `-oxy-collapse-text` ([on page 2345]) to render the text only in the form control that the `oxy_editor` function specifies.

- **resultSeparator** - If multiple check-boxes are used, the separator is used to compose the final result. If not specified, the `space` character is used.

- **tooltips** - Associates tooltips to each value in the `values` property. The value of this property is a list of tooltip messages separated by commas. If you want the tooltip to display a comma, use the `$[comma]` variable ([on page 329]).
• **visible** - Specifies whether or not the form control is visible. The possible values of this property are `true` (default value) and `false`.

• **values** - Specifies the values that are committed when the check-boxes are selected. If these values are not specified in the CSS, they are collected from the associated XML Schema.

![Note:](image)

Typically, when you use a comma in the values of a form control, the content that follows a comma is considered a new value. If you want to include a comma in the values, precede the comma with two backslashes. For example, `{values, '1\, 2\, 3, 4, edit, false}` will display a form control that has `1, 2, 3` for the first value and `4` for the second value.

• **fontInherit** - This value specifies whether or not the form control inherits its font from its parent element. The values of this property can be `true` or `false` (default value). To make the form control inherit its font from its parent element, set the `fontInherit` property to `true`.

• **uncheckedValues** - Specifies the values that are committed when check-boxes are not selected.

• **labels** - This property must have the same number of items as the `values` property. Each item provides a literal description of the items listed in the `values` property. These labels can be translated using the `$i18n()` editor variable (on page 331). If this property is not specified, the `values` property is used as the label.

• **columns** - Controls the layout of the form control. The check boxes will be grouped in a number of columns equal to the given value.

• **color** - Specifies the foreground color of the form control. If the value of the `color` property is `inherit`, the form control has the same color as the element that was used to insert it.

• **hoverPseudoclassName** - Allows you to change the way an element is rendered when you hover over a form control. The value is the name of a CSS pseudo-class. When you hover over the form control, the specified pseudo-class will be set on the element that contains the form control.

```css
p:before {
    content: oxy_checkbox(hoverPseudoclassName, 'showBorder')
}
p:showBorder {
    border: 1px solid red;
}
```

**Example: Single oxy_checkbox Form Control**

```css
checkBox[attribute]:before {
    content: "A check box editor that edits a two valued attribute (On/Off)
    The values are specified in the CSS:
    
    oxy_checkbox{
        edit, "@attribute",
        values, "On",
        uncheckedValues, "Off",
    }
```
Example: Multiple oxy_checkbox Form Controls

```css
multipleCheckBox[attribute]:before {
    content: "Multiple checkboxes editor that edits an attribute value.
    Depending whether the check-box is selected,
    a different value is committed:"
    oxy_checkbox{
        edit, "@attribute",
        values, "true, yes, on",
        uncheckedValues, "false, no, off",
        resultSeparator, ",",
        labels, "Present, Working, Started";}
}
```

Tip:
To insert a sample of the oxy_checkbox form control in a CSS file (or LESS file), invoke the Content Completion Assistant (on page 3217) by pressing Ctrl + Space (Command + Space on macOS) and select the .oxy_checkbox code template.

To see more detailed examples and more information about how form controls work in Oxygen XML Editor, see the sample files in the following directory: [OXYGEN_INSTALL_DIR]/samples/form-controls.

For more information about form controls, watch our video demonstration:

https://www.youtube.com/embed/-WY3wzkMSLM

Related information
Custom CSS Functions (on page 2353)
Collapse Text: -oxy-collapse-text Property Value (on page 2345)

Combo Box Form Control

The oxy_combobox built-in form control is used for providing a graphical user interface object that is a drop-down menu of proposed values. This form control can also be used for a combination of a drop-down menu and an editable single-line text field.

The oxy_combobox form control supports the following properties:
• **edit** - Lets you edit the value of an attribute, the text content of an element, or Processing Instructions (PI). This property can have the following values:
  ◦ **@attribute_name** - The name of the attribute whose value is being edited. If the attribute is in a namespace, the value of the property must be a QName (on page 3223) and the CSS must have a namespace declaration for the prefix.
  ◦ **#text** - Specifies that the presented/edited value is the simple text value of an element.

  ![Note:](image)
  You can set the value of the **visibility** property to **-oxy-collapse-text** (on page 2345) to render the text only in the form control that the **oxy_editor** function specifies.

• **columns** - Controls the width of the form control. The unit size is the width of the w character.

• **width** - Specifies the width of the content area using relative (em, ex), absolute (in, cm, mm, pt, pc, px), and percentage (followed by the % character) length units. The **width** property takes precedence over the **columns** property (if the two are used together).

• **visible** - Specifies whether or not the form control is visible. The possible values of this property are **true** (default value) and **false**.

• **editable** - This property accepts the **true** and **false** values. In addition to a drop-down menu, the **true** value also generates an editable text field box that allows you to insert other values than the proposed ones. The **false** value generates a drop-down menu that only accepts the proposed values.

• **tooltips** - Associates tooltips to each value in the **values** property. The value of this property is a list of tooltip messages separated by commas. If you want the tooltip to display a comma, use the **${comma}** variable (on page 329).

• **values** - Specifies the values that populate the list of proposals. If these values are not specified in the CSS, they are collected from the associated XML Schema.

  ![Note:](image)
  Typically, when you use a comma in the values of a form control, the content that follows a comma is considered a new value. If you want to include a comma in the values, precede the comma with two backslashes. For example, **(values, '1\, 2\, 3, 4, edit, false)** will display a form control that has **1, 2, 3** for the first value and **4** for the second value.

• **fontInherit** - This value specifies whether or not the form control inherits its font from its parent element. The values of this property can be **true** or **false** (default value). To make the form control inherit its font from its parent element, set the **fontInherit** property to **true**.

• **labels** - This property must have the same number of items as the **values** property. Each item provides a literal description of the items listed in the **values** property. These labels can be translated using the **${i18n()}** editor variable (on page 331).

  ![Note:](image)
  This property is only available for read-only combo boxes (the **editable** property is set to **false**).
• **color** - Specifies the foreground color of the form control. If the value of the color property is *inherit*, the form control has the same color as the element that was used to insert it.

• **hoverPseudoclassName** - Allows you to change the way an element is rendered when you hover over a form control. The value is the name of a CSS pseudo-class. When you hover over the form control, the specified pseudo-class will be set on the element that contains the form control.

```css
p:before {
    content: oxy_combobox(hoverPseudoclassName, 'showBorder')
}
p:showBorder {
    border: 1px solid red;
}
```

• **canRemoveValue** - If the value is set to true and the combo box is not editable, then a new `<Empty>` value is added in that combo box. This clears or removes the value being edited, depending on if it edits an element or attribute.

• **onChange** - Can be used to invoke an action when the value of the combo box changes. The action can be created in the CSS using the `oxy_action()` function (on page 2354) or referenced from the framework (on page 3219) by its ID. After the action is executed, the cursor remains in the combo box. Note that this property does not support actions defined by JavaScript code.

**Example: oxy_combobox Form Control**

```xml
comboBox:before {
    content: "A combo box that edits an attribute value. The possible values are provided from CSS:"
    oxy_combobox{
        edit, "@attribute",
        editable, false,
        values, "value1, value2, value3",
        labels, "Value no1, Value no2, Value no3",
        onChange, oxy_action{
            name, 'Insert',
            operation, 'XQueryUpdateOperation',
            arg-script, "insert node <product>{xs:string(@attribute)}</product>
            as last into "."};
    }
}
```

**Tip:**
To insert a sample of the oxy_combobox form control in a CSS file (or LESS file), invoke the **Content Completion Assistant** (on page 3217) by pressing Ctrl + Space (Command + Space on macOS) and select the oxy_combobox code template.
To see more detailed examples and more information about how form controls work in Oxygen XML Editor, see the sample files in the following directory: 

[OXYGEN_INSTALL_DIR]/samples/form-controls.

For more information about form controls, watch our video demonstration:

https://www.youtube.com/embed/-WY3wzkMSLM

Related information

- Custom CSS Functions (on page 2353)
- Actions: oxy_action() Function (on page 2354)
- Collapse Text: -oxy-collapse-text Property Value (on page 2345)

Date Picker Form Control

The oxy_datePicker built-in form control is used for offering a text field with a calendar browser that allows the user to choose a certain date in a specified format.

The oxy_datePicker form control supports the following properties:

- **edit** - Lets you edit the value of an attribute, the text content of an element, or Processing Instructions (PI). This property can have the following values:
  - @attribute_name - The name of the attribute whose value is being edited. If the attribute is in a namespace, the value of the property must be a QName (on page 3223) and the CSS must have a namespace declaration for the prefix.
  - #text - Specifies that the presented/edited value is the simple text value of an element.

  **Note:**
  You can set the value of the visibility property to -oxy-collapse-text (on page 2345) to render the text only in the form control that the oxy_editor function specifies.

- **columns** - Controls the width of the form control. The unit size is the width of the w character.
- **width** - Specifies the width of the content area using relative (em, ex), absolute (in, cm, mm, pt, pc, px), and percentage (followed by the % character) length units. The width property takes precedence over the columns property (if the two are used together).
- **color** - Specifies the foreground color of the form control. If the value of the color property is inherit, the form control has the same color as the element that was used to insert it.
- **fontInherit** - This value specifies whether or not the form control inherits its font from its parent element. The values of this property can be true or false (default value). To make the form control inherit its font from its parent element, set the fontInherit property to true.
- **format** - This property specifies the format of the inserted date, if a specific format is not detected from the associated document schema. The pattern value must be a valid Java date (or date-time) format. If this property is missing, the format of the date is determined from the associated schema.
- **visible** - Specifies whether or not the form control is visible. The possible values of this property are `true` (default value) and `false`.

- **validateInput** - Specifies if the form control is validated. If you introduce a date that does not respect the format, the `datePicker` form control is rendered with a red foreground. By default, the input is validated. To disable the validation, set this property to `false`.

- **hoverPseudoclassName** - Allows you to change the way an element is rendered when you hover over a form control. The value is the name of a CSS pseudo-class. When you hover over the form control, the specified pseudo-class will be set on the element that contains the form control.

```css
p:before {
  content: oxy_datePicker(edit, "@attribute", hoverPseudoclassName, 'showBorder')
}
p:showBorder {
  border: 1px solid red;
}
```

**Example: oxy_datePicker Form Control**

```xml
date {
  content:
    oxy_label(text, "Date time attribute with format defined in CSS: ", width, 300px)
    oxy_datePicker(
      columns, 16,
      edit, "@attribute",
      format, "yyyy-MM-dd");
}
```

**Tip:**

To insert a sample of the `oxy_datePicker` form control in a CSS file (or LESS file), invoke the Content Completion Assistant (on page 3217) by pressing `Ctrl + Space` (Command + Space on macOS) and select the `oxy_datePicker` code template.

To see more detailed examples and more information about how form controls work in Oxygen XML Editor, see the sample files in the following directory: `[OXYGEN_INSTALL_DIR]/samples/form-controls`.

For more information about form controls, watch our video demonstration:

https://www.youtube.com/embed/-WY3wzkMSLM

**Related information**

- Custom CSS Functions (on page 2353)
- Label: oxy_label() Function (on page 2362)
## HTML Content Form Control

The `oxy_htmlContent` built-in form control is used for rendering HTML content. This HTML content is displayed as a graphical element shaped as a box. The shape of the box is determined by a given width and the height is computed based upon the length of the text.

The `oxy_htmlContent` form control supports the following properties:

- **href** - The absolute or relative location of a resource. The resource needs to be a well-formed HTML file.
- **id** - The unique identifier of an item. This is a `<div>` element that has a unique `@id` and is a child of the `<body>` element. The `<div>` element is the container of the HTML content to be rendered by the form control.
- **content** - An alternative to the `@href` and `@id` pair of elements. It provides the HTML content that will be displayed in the form control.
- **width** - Specifies the width of the content area using relative (`em`, `ex`), absolute (`in`, `cm`, `mm`, `pt`, `pc`, `px`), and percentage (followed by the `%` character) length units. The `width` property takes precedence over the `columns` property (if the two are used together).
- **hoverPseudoclassName** - Allows you to change the way an element is rendered when you hover over a form control. The value is the name of a CSS pseudo-class. When you hover over the form control, the specified pseudo-class will be set on the element that contains the form control.

```html
p:before {
  content: oxy_htmlContent(hoverPseudoclassName, 'showBorder')
}
p:showBorder {
  border: 1px solid red;
}
```

You can customize the style of the content using CSS that is either referenced by the file identified by the `href` property or is defined inline. If you change the HTML content or CSS and you want your changes to be reflected in the XML that renders the form control, then you need to refresh the XML file. If the HTML does not have an associated style, then a default text and background color will be applied.

### Example: oxy_htmlContent Form Control

In the following example, the form control collects the content from the `p_description` `<div>` element found in the `descriptions.html` file. The box is 400 pixels wide and is displayed before a paragraph identified by the `@intro_id` attribute value.

```html
p#intro_id:before {
  content: oxy_htmlContent(
    href, "descriptions.html",
    id, "p_description",
    width, 400px)
}
```
An alternative example, using the `content` property:

```css
p#intro_id:before {
  content: oxy_htmlContent(
    content, "<div style='font-weight:bold;'>My content</div>",
    width, 400px);
}
```

**Tip:**
To insert a sample of the `oxy_htmlContent` form control in a CSS file (or LESS file), invoke the *Content Completion Assistant* by pressing `Ctrl + Space (Command + Space on macOS)` and select the `oxy_htmlContent` code template.

To see more detailed examples and more information about how form controls work in Oxygen XML Editor, see the sample files in the following directory: `[/OXYGEN_INSTALL_DIR]/samples/form-controls`.

For more information about form controls, watch our video demonstration:

https://www.youtube.com/embed/-WY3wzkMSLM

**Related information**
Custom CSS Functions (on page 2353)

---

**Pop-up Form Control**

The `oxy_popup` built-in form control is used to offer a contextual menu that provides quick access to various actions. A pop-up form control can display single or multiple selections.

The `oxy_popup` form control supports the following properties:

- **edit** - Lets you edit the value of an attribute, the text content of an element, or Processing Instructions (PI). This property can have the following values:
  - @attribute_name - The name of the attribute whose value is being edited. If the attribute is in a namespace, the value of the property must be a `QName` and the CSS must have a namespace declaration for the prefix.
  - #text - Specifies that the presented/edited value is the simple text value of an element.

  **Note:**
  You can set the value of the `visibility` property to `oxy-collapse-text` to render the text only in the form control that the `oxy_editor` function specifies.

- **rows** - This property specifies the number of rows that the form control presents.
Note: If the value of the `rows` property is not specified, the default value of 12 is used.

- **color** - Specifies the foreground color of the form control. If the value of the `color` property is `inherit`, the form control has the same color as the element that was used to insert it.

Note: This property is used for rendering in the **Author** mode.

- **visible** - Specifies whether or not the form control is visible. The possible values of this property are `true` (default value) and `false`.

- **tooltips** - Associates tooltips to each value in the `values` property. The value of this property is a list of tooltip messages separated by commas. If you want the tooltip to display a comma, use the `$[comma]$` variable (on page 329).

Example:

```css
link:before{
  content: oxy_popup(
    edit, '@href',
    values, "Spring, Summer, Autumn, Winter",
    tooltips, "Iris${comma}Snowdrop, Gardenia${comma}Liliac, Chrysanthemum${comma}Salvia, Gerbera",
    selectionMode, single);
}
```

- **values** - Specifies the values that populate the list of proposals. If these values are not specified in the CSS, they are collected from the associated XML Schema.

Note: Typically, when you use a comma in the values of a form control, the content that follows a comma is considered a new value. If you want to include a comma in the values, precede the comma with two backslashes. For example, `(values, '1\\, 2\\, 3, 4, edit, false)` will display a form control that has 1, 2, 3 for the first value and 4 for the second value.

- **resultSeparator** - If multiple check-boxes are used, the separator is used to compose the final result. If not specified, the `space` character is used.

Note: The value of the `resultSeparator` property cannot exceed one character.
• **selectionMode** - Specifies whether the form control allows the selection of a single value or multiple values. The predefined values of this property are `single` (default value) and `multiple`.

• **labels** - Specifies the label associated with each entry used for presentation. If this property is not specified, the `values` property is used instead.

• **columns** - Controls the width of the form control. The unit size is the width of the `w` character. This property is used for the visual representation of the form control.

• **width** - Specifies the width of the content area using relative (em, ex), absolute (in, cm, mm, pt, pc, px), and percentage (followed by the `%` character) length units. The `width` property takes precedence over the `columns` property (if the two are used together).

• **rendererSort** - Specifies the sorting of the values (labels) displayed on the form control before clicking the popup control. The possible values of this property are `ascending` and `descending`.

• **editorSort** - Specifies the sorting of the values displayed after clicking the popup control (for example, clicking a drop-down arrow button). The possible values of this property are `ascending` and `descending`.

• **rendererSeparator** - Defines a separator used when multiple values are rendered. If not specified, the value of the `resultSeparator` property is used.

• **sort** - Specifies the default sorting of the form control values (the values displayed before and after clicking the popup control). However, the `editorSort` and `rendererSort` properties have a higher priority. The possible values of this property are `ascending` and `descending`.

• **fontInherit** - This value specifies whether or not the form control inherits its font from its parent element. The values of this property can be `true` or `false` (default value). To make the form control inherit its font from its parent element, set the `fontInherit` property to `true`.

• **hoverPseudoclassName** - Allows you to change the way an element is rendered when you hover over a form control. The value is the name of a CSS pseudo-class. When you hover over the form control, the specified pseudo-class will be set on the element that contains the form control.

```html
defining a separator used when multiple values are rendered. If not specified, the value of the `resultSeparator` property is used.

• **sort** - Specifies the default sorting of the form control values (the values displayed before and after clicking the popup control). However, the `editorSort` and `rendererSort` properties have a higher priority. The possible values of this property are `ascending` and `descending`.

• **fontInherit** - This value specifies whether or not the form control inherits its font from its parent element. The values of this property can be `true` or `false` (default value). To make the form control inherit its font from its parent element, set the `fontInherit` property to `true`.

• **hoverPseudoclassName** - Allows you to change the way an element is rendered when you hover over a form control. The value is the name of a CSS pseudo-class. When you hover over the form control, the specified pseudo-class will be set on the element that contains the form control.

```html
Example: **oxy_popup** Form Control

```html
defining a separator used when multiple values are rendered. If not specified, the value of the `resultSeparator` property is used.

• **sort** - Specifies the default sorting of the form control values (the values displayed before and after clicking the popup control). However, the `editorSort` and `rendererSort` properties have a higher priority. The possible values of this property are `ascending` and `descending`.

• **fontInherit** - This value specifies whether or not the form control inherits its font from its parent element. The values of this property can be `true` or `false` (default value). To make the form control inherit its font from its parent element, set the `fontInherit` property to `true`.

• **hoverPseudoclassName** - Allows you to change the way an element is rendered when you hover over a form control. The value is the name of a CSS pseudo-class. When you hover over the form control, the specified pseudo-class will be set on the element that contains the form control.

```html
Tip:
To insert a sample of the oxy_popup form control in a CSS file (or LESS file), invoke the Content Completion Assistant (on page 3217) by pressing Ctrl + Space (Command + Space on macOS) and select the .oxy_popup code template.

To see more detailed examples and more information about how form controls work in Oxygen XML Editor, see the sample files in the following directory: [OXYGEN_INSTALL_DIR]/samples/form-controls.

For more information about form controls, watch our video demonstration:

https://www.youtube.com/embed/-WY3wzkMSLM

Related information
Custom CSS Functions (on page 2353)
Collapse Text: .oxy-collapse-text Property Value (on page 2345)

Text Area Form Control

The oxy_textArea built-in form control is used for entering multiple lines of text in a graphical user interface box. A text area may include optional syntax highlight capabilities to present the form control.

The oxy_textArea form control supports the following properties:

- **edit** - Lets you edit the value of an attribute, the text content of an element, or Processing Instructions (PI). This property can have the following values:
  - `@attribute_name` - The name of the attribute whose value is being edited. If the attribute is in a namespace, the value of the property must be a QName (on page 3223) and the CSS must have a namespace declaration for the prefix.
  - `#text` - Specifies that the presented/edited value is the simple text value of an element.

**Note:**
You can set the value of the visibility property to `oxy-collapse-text` (on page 2345) to render the text only in the form control that the oxy_editor function specifies.

- `#content` - This parameter is useful when an element has mixed or element-only content and you want to edit its content inside a text area form control.
For example, if you have the following XML content:

```xml
<codeblock outputclass="language-xml">START_TEXT<ph>phase</ph><apiname><text>API</text></apiname></codeblock>
```

and your CSS includes the following snippet:

```css
codeblock:before{
  content:
    oxy_textArea(
      edit, '#content',
      contentType, 'text/xml');
}
```

then the text area form control will edit the following fragment:

```xml
START_TEXT<ph>phase</ph><apiname><text>API</text></apiname>
```

Note: When the value of the `edit` property is `#content`, the text area form control will also offer content completion proposals.

• `#content` - This parameter is useful when an element has mixed or element-only content and you want to edit its content inside a text area form control.

For example, if you have the following XML content:

```xml
<codeblock outputclass="language-xml">START_TEXT<ph>phase</ph><apiname><text>API</text></apiname></codeblock>
```

and your CSS includes the following snippet:

```css
codeblock:before{
  content:
   oxy_textArea(
      edit, '#content',
      contentType, 'text/xml');
}
```

then the text area form control will edit the following fragment:

```xml
START_TEXT<ph>phase</ph><apiname><text>API</text></apiname>
```

Note: When the value of the `edit` property is `#content`, the text area form control will also offer content completion proposals.
• **columns** - Controls the width of the form control. The unit size is the width of the w character.

• **width** - Specifies the width of the content area using relative (em, ex), absolute (in, cm, mm, pt, pc, px), and percentage (followed by the % character) length units. The width property takes precedence over the columns property (if the two are used together).

• **fontInherit** - This value specifies whether or not the form control inherits its font from its parent element. The values of this property can be true or false (default value). To make the form control inherit its font from its parent element, set the fontInherit property to true.

• **visible** - Specifies whether or not the form control is visible. The possible values of this property are true (default value) and false.

• **rows** - This property specifies the number of rows that the form control presents. If the form control has more lines, you can scroll and see them all.

• **contentType** - Specifies the type of content that the form control will format with syntax highlighting. The following values are supported: text/css; text/shell; text/cc; text/xquery; text/xml; text/plain; text/python; text/xsd; text/c; text/xpath; text/javascript; text/xsl; text/wsdl; text/html; text/xproc; text/properties; text/sql; text/rng; text/sch; text/json; text/perl; text/php; text/java; text/batch; text/rnc; text/dtd; text/nvdl; text/plain.

• **indentOnTab** - Specifies the behavior of the Tab key. If the value of this property is set to true (default value), the Tab key inserts characters. If it is set to false, Tab is used for navigation, jumping to the next editable position in the document.

• **white-space** - CSS property that influences the value that you edit, as well as the form control size:
  - **pre** - The whitespaces and new lines of the value are preserved and edited. If the rows and columns properties are not specifies, the form control calculates its size on its own so that all the text is visible.
  - **pre-wrap** - The long lines are wrapped to avoid horizontal scrolling.
  - **normal** - The white spaces and new lines are normalized.

• **hoverPseudoclassName** - Allows you to change the way an element is rendered when you hover over a form control. The value is the name of a CSS pseudo-class. When you hover over the form control, the specified pseudo-class will be set on the element that contains the form control.

```css
p:before {
  content: oxy_textArea(hoverPseudoclassName, 'showBorder')
}
p:showBorder {
  border: 1px solid red;
}
```

**Example:** oxy_textArea Form Control
The following example presents a text area with CSS syntax highlighting that calculates its own dimension, and a second one with XML syntax highlighting with defined dimension.

```css
textArea {
  visibility: -oxy-collapse-text;
  white-space: pre;
}

textArea[language="CSS"]:before {
  content: oxy_textArea{
    edit, '#text',
    contentType, 'text/css'};
}

textArea[language="XML"]:before {
  content: oxy_textArea{
    edit, '#text',
    contentType, 'text/xml',
    rows, 10,
    columns, 30};
}
```

**Tip:**
To insert a sample of the `oxy_textArea` form control in a CSS file (or LESS file), invoke the [Content Completion Assistant](on page 3217) by pressing Ctrl + Space (Command + Space on macOS) and select the `oxy_textArea` code template.

To see more detailed examples and more information about how form controls work in Oxygen XML Editor, see the sample files in the following directory: `[OXYGEN_INSTALL_DIR]/samples/form-controls`.

For more information about form controls, watch our video demonstration:

[https://www.youtube.com/embed/-WY3wzkMSLM](https://www.youtube.com/embed/-WY3wzkMSLM)

**Related information**
- [Custom CSS Functions](on page 2353)
- [Collapse Text: -oxy-collapse-text Property Value](on page 2345)

**Text Field Form Control**

The `oxy_textfield` built-in form control is used for entering a single line of text in a graphical user interface box. A text field may include optional content completion capabilities, used to present and edit the value of an attribute or an element.
The oxy_textfield form control supports the following properties:

- **edit** - Lets you edit the value of an attribute, the text content of an element, or Processing Instructions (PI). This property can have the following values:
  - **@attribute_name** - The name of the attribute whose value is being edited. If the attribute is in a namespace, the value of the property must be a *QName (on page 3223)* and the CSS must have a namespace declaration for the prefix.
  - **#text** - Specifies that the presented/edited value is the simple text value of an element.

  **Note:** You can set the value of the visibility property to -oxy-collapse-text *(on page 2345)* to render the text only in the form control that the oxy_editor function specifies.

- **columns** - Controls the width of the form control. The unit size is the width of the w character.
- **width** - Specifies the width of the content area using relative (em, ex), absolute (in, cm, mm, pt, pc, px), and percentage (followed by the % character) length units. The width property takes precedence over the columns property (if the two are used together).
- **fontInherit** - This value specifies whether or not the form control inherits its font from its parent element. The values of this property can be **true** or **false** (default value). To make the form control inherit its font from its parent element, set the fontInherit property to **true**.
- **visible** - Specifies whether or not the form control is visible. The possible values of this property are **true** (default value) and **false**.
- **values** - Specifies the values that populate the list of proposals. If these values are not specified in the CSS, they are collected from the associated XML Schema.
- **tooltips** - Associates tooltips to each value in the values property. The value of this property is a list of tooltip messages separated by commas. If you want the tooltip to display a comma, use the $comma variable *(on page 329)*.
- **tooltip** - Specifies a tooltip to be displayed when you hover over the form control.
- **color** - Specifies the foreground color of the form control. If the value of the color property is **inherit**, the form control has the same color as the element that was used to insert it.
- **hasMultipleValues** - Specifies if the text field allows multiple values separated by spaces or just a single value.

  **Note:** If the value is **false**, the Content Completion Assistant *(on page 3217)* considers the entire text as the prefix for its proposals. If the value is **true** (the default value), the space is the delimiter for the values and thus it is not included in the prefix (the prefix will be whatever comes after the space).

  For example, suppose the possible values for your text field are: value a, value b, and other values. If the hasMultipleValues property is set to **true** and the user enters "value " (notice the space character after 'value') in the text field, the Content Completion Assistant will suggest all three values because the prefix is whatever comes after the space, and in this case the user
did not enter anything after the space. If the `hasMultipleValues` property was set to `false`, the Content Completion Assistant would only suggest `value a` and `value b` because the space is considered part of the prefix.

- **hoverPseudoclassName** - Allows you to change the way an element is rendered when you hover over a form control. The value is the name of a CSS pseudo-class. When you hover over the form control, the specified pseudo-class will be set on the element that contains the form control.

```html
p:before {
    content: oxy_textfield(hoverPseudoclassName, 'showBorder')
}
p:showBorder {
    border: 1px solid red;
}
```

**Example: oxy_textfield Form Control**

```html
element {
    content: "Label: "
    oxy_textfield(
        edit, "@my_attr",
        values, "value1, value2",
        color, "red",
        columns, 40);
}
```

**Tip:**

To insert a sample of the `oxy_textfield` form control in a CSS file (or LESS file), invoke the Content Completion Assistant (on page 3217) by pressing **Ctrl + Space (Command + Space on macOS)** and select the `oxy_textfield` code template.

To see more detailed examples and more information about how form controls work in Oxygen XML Editor, see the sample files in the following directory: `[OXYGEN_INSTALL_DIR]/samples/form-controls`.

For more information about form controls, watch our video demonstration:

https://www.youtube.com/embed/-WY3wzkMSLM

**Related information**

- Custom CSS Functions (on page 2353)
- Collapse Text: -oxy-collapse-text Property Value (on page 2345)
URL Chooser Form Control

The oxy_urlChooser built-in form control is used for a dialog box that allows you to select the location of local or remote resources. The inserted reference is made relative to the URL of the currently open editor.

The oxy_urlChooser editor supports the following properties:

- **edit** - Lets you edit the value of an attribute, the text content of an element, or Processing Instructions (PI). This property can have the following values:
  - @attribute_name - The name of the attribute whose value is being edited. If the attribute is in a namespace, the value of the property must be a QName (on page 3223) and the CSS must have a namespace declaration for the prefix.
  - #text - Specifies that the presented/edited value is the simple text value of an element.

  **Note:** You can set the value of the `visibility` property to `oxy-collapse-text` (on page 2345) to render the text only in the form control that the `oxy_editor` function specifies.

- **columns** - Controls the width of the form control. The unit size is the width of the w character.
- **width** - Specifies the width of the content area using relative (em, ex), absolute (in, cm, mm, pt, pc, px), and percentage (followed by the % character) length units. The width property takes precedence over the columns property (if the two are used together).
- **color** - Specifies the foreground color of the form control. If the value of the color property is inherit, the form control has the same color as the element that was used to insert it.
- **visible** - Specifies whether or not the form control is visible. The possible values of this property are true (default value) and false.
- **fontInherit** - This value specifies whether or not the form control inherits its font from its parent element. The values of this property can be true or false (default value). To make the form control inherit its font from its parent element, set the fontInherit property to true.
- **fileFilter** - string value that holds comma-separated file extensions. The URL chooser uses these extensions to filter the displayed files. A value such as "jpg,png,gif" is mapped to a single filter that will display all jpg, png, and gif files.
- **hoverPseudoclassName** - Allows you to change the way an element is rendered when you hover over a form control. The value is the name of a CSS pseudo-class. When you hover over the form control, the specified pseudo-class will be set on the element that contains the form control.

```html
p:before {
    content: oxy_urlChooser(hoverPseudoclassName, 'showBorder')
}
p:showBorder {
    border: 1px solid red;
}
```

**Example: oxy_urlChooser Form Control**
urlChooser[&file]:before {
    content: "A URL chooser editor that allows browsing for a URL. The selected URL is made relative to the currently edited file:
    oxy_urlChooser(
        edit, "@file",
        columns 25);
}

Tip:
To insert a sample of the oxy_urlChooser form control in a CSS file (or LESS file), invoke the Content Completion Assistant (on page 3217) by pressing Ctrl + Space (Command + Space on macOS) and select the oxy_urlChooser code template.

To see more detailed examples and more information about how form controls work in Oxygen XML Editor, see the sample files in the following directory: [OXYGEN_INSTALL_DIR]/samples/form-controls.

For more information about form controls, watch our video demonstration:

https://www.youtube.com/embed/-WY3wzkMSLM

Related information
Custom CSS Functions (on page 2353)
Collapse Text: -oxy-collapse-text Property Value (on page 2345)

Video Player Form Control
The oxy_video built-in form control is used for providing a mechanism to play videos.

The oxy_video form control supports the following properties:

- **href** - The absolute or relative location of a resource. This property is mandatory. Relative values are resolved relative to the CSS. If you have media resources relative to the XML document, you can specify their paths like this:
  
  oxy_video(href, oxy_url(oxy_base-uri(), 'ex.mp4'), width, 400px, height, 300px)

- **width** - Specifies the width of the content area using relative (em, ex), absolute (in, cm, mm, pt, pc, px), and percentage (followed by the % character) length units.

- **height** - Specifies the height of the form control area using relative (em, ex), absolute (in, cm, mm, pt, pc, px), and percentage (followed by the % character) length units.

Example: oxy_video Form Control

```object {
    content:
```
oxy_video(
    href, 'resources/video.mp4',
    width, 400px,
    height, 300px),
)

Tip:
To insert a sample of the oxy_video form control in a CSS file (or LESS file), invoke the Content Completion Assistant (on page 3217) by pressing Ctrl + Space (Command + Space on macOS) and select the oxy_video code template.

To see more detailed examples and more information about how form controls work in Oxygen XML Editor, see the sample files in the following directory: [OXYGEN_INSTALL_DIR]/samples/form-controls.

For more information about form controls, watch our video demonstration:

https://www.youtube.com/embed/-WY3wzkMSLM

Related information
Custom CSS Functions (on page 2353)
URL: oxy_url() Function (on page 2370)

Implementing Custom Form Controls

If the built-in form controls are not sufficient for your needs, you can implement custom form controls in Java.

Custom Form Controls Implementation

You can specify custom form controls using the following properties:

- **rendererClassName** - The name of the class that draws the edited value. It must be an implementation of ro.sync.ecss.extensions.api.editor.InplaceRenderer. The renderer has to be a SWING implementation and can be used both in the standalone and Eclipse distributions.

- **swingEditorClassName** - You can use this property for the standalone (Swing-based) distribution to specify the name of the class used for editing. It is a Swing implementation of ro.sync.ecss.extensions.api.editor.InplaceEditor.

- **swtEditorClassName** - You can use this property for the Eclipse plugin distribution to specify the name of the class used for editing. It is a SWT implementation of the ro.sync.ecss.extensions.api.editor.InplaceEditor.

Note:
If the custom form control is intended to work in the Oxygen XML Editor standalone distribution, the declaration of swtEditorClassName is not required. The renderer (the class
that draws the value) has different properties from the editor (the class that edits the value) because you can present a value in one way and edit it in another.

- **classpath** - You can use this property to specify the location of the classes used for a custom form control. The value of the `classpath` property is an enumeration of URLs separated by comma.

- **edit** - If your form control edits the value of an attribute or the text value of an element, you can use the `@attribute_name` and `#text` predefined values and Oxygen XML Editor will perform the commit logic by itself. You can use the `custom` value to perform the commit logic yourself.

- **saHeavyFormControlClassName** - This type of form control is effectively present at all times at its allocated bounds. This is useful if you need a form control that renders dynamic or interactive SVG documents (for example, if you have an SVG document that displays tooltips when hovering over certain areas). It is also helpful if you want to use JavaFX, since JavaFX-based form controls are not compatible with the classic form control architecture.

  The value of this property is a class name that must implement the `ro.sync.ecss.extensions.api.editor.InplaceHeavyEditor` method. The JAR (on page 3220) that contains this implementation can either be added in the Classpath tab in the Document Type Configuration dialog box (on page 144) for your particular framework (on page 3219) or specified with the `classpath` property (on page 2405).

**Example: Java Code**

The following is a sample Java code for implementing a custom combo box form control that inserts an XML element in the content when the editing stops:

```java
public class ComboBoxEditor extends AbstractInplaceEditor {

    /**
     * @see ro.sync.ecss.extensions.api.editor.InplaceEditor#stopEditing()
     */
    @Override
    public void stopEditing() {
        Runnable customCommit = new Runnable() {
            @Override
            public void run() {
                AuthorDocumentController documentController =
                context.getAuthorAccess().getDocumentController();
                documentController.insertXMLFragment("<custom/>", offset);}
        };
        EditingEvent event = new EditingEvent(customCommit, true);
        fireEditingStopped(event);
    }
}
```
The custom form controls can use any of the predefined properties of the built-in form controls (on page 2373), as well as specified custom properties.

**Example: CSS**

The following is an example of how to specify a custom form control in the CSS:

```css
myElement {
  content: oxy_editor(
    rendererClassName, "com.custom.editors.CustomRenderer",
    swingEditorClassName, "com.custom.editors.SwingCustomEditor",
    swtEditorClassName, "com.custom.editors.SwtCustomEditor",
    edit, "@my_attr",
    customProperty1, "customValue1",
    customProperty2, "customValue2"
  )
}
```

**How to Implement Custom Form Controls**

To implement a custom form control, follow these steps:

1. Download the Oxygen XML Editor SDK at: [https://www.oxygenxml.com/oxygen_sdk.html](https://www.oxygenxml.com/oxygen_sdk.html).
2. Implement the custom form control by extending `ro.sync.ecss.extensions.api.editor.InplaceEditorRendererAdapter`. You could also use `ro.sync.ecss.extensions.api.editor.AbstractInplaceEditor`, which offers some default implementations and listeners management.
3. Pack the previous implementation in a Java JAR (on page 3220) library.
4. Copy the JAR library to the `[OXYGEN_INSTALL_DIR]/frameworks/[FRAMEWORK_DIR]` directory.
5. In Oxygen XML Editor, open the Preferences dialog box (Options > Preferences) (on page 124), go to Document Type Association, edit the appropriate framework, and add the JAR library in the Classpath tab.
6. Specify the custom form control in your CSS, as described above.

**Tip:**

To see more detailed examples and more information about how form controls work in Oxygen XML Editor, see the sample files in the following directory: `[OXYGEN_INSTALL_DIR]/samples/form-controls`.

For more information about form controls, watch our video demonstration:

[https://www.youtube.com/embed/-WY3wzkMSLM](https://www.youtube.com/embed/-WY3wzkMSLM)
Editing Processing Instructions Using a Form Control

Oxygen XML Editor allows you to edit *processing instructions, comments, and CDATA* by using CSS extensions.

Note:
You can edit both the content and the attribute value from a processing instruction.

Example: Editing an Attribute from a Processing Instruction

PI content:

```xml
<?pi_target attr="val"/>
```

CSS:

```css
@namespace oxy "http://www.oxygenxml.com/extensions/author";

oxy|processing-instruction:before {
    display:inline;
    content:
        "EDIT attribute: " oxy_textfield(edit, '@attr', columns, 15);
    visibility:visible;
}
oxy|processing-instruction{
    visibility:-oxy-collapse-text;
}
```

Related information

- Text Field Form Control *(on page 2399)*
- Collapse Text: -oxy-collapse-text Property Value *(on page 2345)*
- Displaying Processing Instructions from Other XML Editors *(on page 2310)*

Custom CSS Pseudo-classes

You can set your custom CSS pseudo-classes on the nodes from the *AuthorDocument* model. These are similar to the normal XML attributes, with the important difference that they are not serialized, and by changing them, the document does not create undo and redo edits (the document is considered unmodified).

You can use custom pseudo-classes for changing the style of an element (and its children) without altering the document.

In Oxygen XML Editor they are used to hide/show the `colspec` elements from CALS tables. To take a look at the implementation, see:
1. \( [OXYGEN\_INSTALL\_DIR]/frameworks/docbook/css/cals_table.css \) (Search for \(-oxy-\)visible-colspecs)

2. The definition of action \texttt{table.toggle.colspec} from the DocBook framework \textit{(on page 3219)} makes use of the pre-defined \textit{TogglePseudoClassOperation Author} mode operation.

Here are some examples:

**Example: Controlling the visibility of a section using a pseudo-class**

You can use a non standard (custom) pseudo-class to impose a style change on a specific element. For instance, you can have CSS styles matching the custom pseudo-class \texttt{access-control-user}, like the one below:

```
section {
    display:none;
}

section:access-control-user {
    display:block;
}
```

By setting the pseudo-class \texttt{access-control-user}, the element section will become visible by matching the second CSS selector.

**Example: Coloring the elements at the current cursor location**

You could create an \textit{AuthorCaretListener} that sets the \texttt{caret-visited} pseudo-class to the element at the cursor location. The effect will be that all the elements traversed by the cursor become red.

```
*:caret-visited {
    color:red;
}
```

The API that you can use from the \textit{CaretListener}.

\begin{verbatim}
ro.sync.ecss.extensions.api.AuthorDocumentController#setPseudoClass(java.lang.String, ro.sync.ecss.extensions.api.node.AuthorElement)
ro.sync.ecss.extensions.api.AuthorDocumentController#removePseudoClass(java.lang.String, ro.sync.ecss.extensions.api.node.AuthorElement)
\end{verbatim}

**Predefined Pseudo-Class Author Mode Operations**

Pre-defined \textit{Author mode operations} can be used directly in your \textit{framework} to work with custom pseudo-classes:
1. **TogglePseudoClassOperation**
2. **SetPseudoClassOperation**
3. **RemovePseudoClassOperation**

**Using the :before(n) and :after(n) CSS Pseudo-Elements**

Although not standard, this extension may be useful if you want to style sections by adding multiple levels of static content. To add static content to an element, you would normally use a :before or :after pseudo-element.

This example adds static text before the title ("Chapter 1", "Chapter 2", etc.):

```css
h1:before {
    content: "Chapter " counter(chapter) ".";
    color: blue;
}
```

All of this is styled with the same color (blue in this example). Using standard CSS, it is impossible to style specific aspects of it (for example, just the chapter number with a larger font and with red). However, you can do it using multiple :before(n) or :after(n) pseudo-elements:

```css
h1:before(3) {
    content: "Chapter ";
    color: blue;
}

h1:before(2) {
    content: counter(chapter);
    color: red;
    font-size: large;
}

h1:before(1) {
    content: ".";
    color: blue;
}
```

**Notes:**

- The bigger the level, the more distant the pseudo-element is.
- Level 1 corresponds to normal :before or :after pseudo-elements.

**Debugging CSS Stylesheets**

To assist you with debugging and customizing CSS stylesheets the Author mode includes a **CSS Inspector view (on page 640)** to examine the CSS rules that match the currently selected element.
This tool is similar to the Inspect Element development tool that is found in most browsers. The CSS Inspector view allows you to see how the CSS rules are applied and the properties defined. Each rule that is displayed in this view includes a link to the line in the CSS file that defines the styles for the element that matches the rule. You can use the link to open the appropriate CSS file and edit the style rules. Once you have found the rule you want to edit, you can click the link in the top-right corner of that rule to open the CSS file in the editor.

There are two ways to open the CSS Inspector view:

1. Select CSS Inspector from the Window > Show View menu.
2. Select the Inspect Styles action from the contextual menu in Author mode.

Related Information:

CSS Inspector View (on page 640)
18. Extending Oxygen With the SDK

Oxygen XML Editor has an SDK that can be used as a base to develop frameworks (on page 3219) and plugins (on page 3222). It can be also used to create projects that use the Oxygen XML Author Component or Oxygen XML Web Author. The SDK is a Java library available under the Oxygen XML SDK licensing terms and is delivered with a set of examples that demonstrate how to extend Oxygen XML functionality through API calls. The SDK is available at https://www.oxygenxml.com/oxygen_sdk.html.

Important:
From a legal standpoint, you can freely develop and share extensions using the Oxygen SDK ONLY if you have a legal, active license to use Oxygen XML Editor and ONLY if such extensions are used from inside Oxygen XML Editor. To use such extensions outside of Oxygen XML Editor (for example, a 3rd-party application that has Oxygen XML Editor built in to it), an additional license must be purchased to use the SDK according the Oxygen XML SDK Licensing Policy.

Extending Oxygen XML Editor with Plugins

A plugin (on page 3222) is a software component that adds extra functionality to the standalone version of the application using a series of application-provided extension points.

This chapter explains how to write and install a plugin for the standalone version of Oxygen XML Editor. The Plugins Development Kit contains sample plugins (source and compiled Java code) and the Javadoc API necessary for developing custom plugins.

If you want to customize the Oxygen XML Editor Eclipse plugin you can look at the Eclipse IDE Integration Sample Project to see how an Eclipse plugin can interact with the Oxygen XML Editor APIs.

General Configuration of an Oxygen XML Editor Plugin

The Oxygen XML Editor functionality can be extended with plugins (on page 3222) that implement a clearly specified API. On the Oxygen XML Editor website, there is an SDK with sample plugins (source and compiled Java code) and the Javadoc API necessary for developing custom plugins.

The minimal implementation of a plugin must provide:

- A Java class that extends the `ro.sync.exml.plugin.Plugin` class.
- A Java class that implements the `ro.sync.exml.plugin.PluginExtension` interface.
- A plugin descriptor file called `plugin.xml`. 
A `ro.sync.exml.plugin.PluginDescriptor` object is passed to the `Constructor` of the subclass of the `ro.sync.exml.plugin.Plugin` class. It contains the following data items about the **plugin**:

- **basedir** (`File` object) - The base directory of the plugin.
- **description** (`String` object) - The description of the plugin.
- **name** (`String` object) - The name of the plugin.
- **vendor** (`String` object) - The vendor name of the plugin.
- **version** (`String` object) - The plugin version.
- **id** (`String` object) - A unique identifier.
- **classLoaderType** - You can choose between `preferOxygenResources` (default value) and `preferReferencedResources`. When choosing `preferOxygenResources`, the libraries that are referenced in the Oxygen XML Editor `lib` directory will have precedence over those referenced in the `plugin.xml` configuration file, if they have the same package names. When choosing `preferReferencedResources`, the libraries that are referenced in the `plugin.xml` configuration file will have precedence over those found in the Oxygen XML Editor `lib` directory, if they have the same package names.

The **plugin** descriptor is an XML file that defines how the **plugin** is integrated in Oxygen XML Editor and what libraries are loaded. The structure of the **plugin** descriptor file is fully described in a DTD grammar located in `{OXYGEN_INSTALL_DIR}/plugins/plugin.dtd`. Here is a sample **plugin** descriptor used by the Capitalize Lines sample **plugin**:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plugin SYSTEM "../plugin.dtd">
<plugin
    name="Capitalize Lines"
    description="Capitalize the first character on each line"
    version="1.0.0"
    vendor="SyncRO"
    class="ro.sync.sample.plugin.caplines.CapLinesPlugin">
    <runtime>
        <library name="lib/caplines.jar"/>
    </runtime>
    <extension type="selectionProcessor"
        class="ro.sync.sample.plugin.caplines.CapLinesPluginExtension"
        keyboardShortcut="ctrl shift EQUALS"/>
</plugin>
```

If your **plugin** is of the **Selection**, **Document** or **General** types, and thus contributes an action either to the contextual menu or to the main menu of the **Text** editing mode, then you can assign a keyboard shortcut for it. You can use the `@keyboardShortcut` attribute for each `<extension>` element to specify the desired shortcut.
Tip:
To compose string representations of the desired shortcut keys you can go to Options > Menu Shortcut Keys, select an action, and click Edit. Then choose the desired key sequence and use the representation that appears in the resulting dialog box.

Referencing Libraries

To reference libraries, use either of the following elements:

- `<library name="path/libraryName" scope="global">` - To point to specific libraries. Notice that the value of library name includes the path (relative or absolute) to the library.

Note:
You can use the `${oxygenInstallDir}` editor variable (on page 331) as part of the value of the @name attribute. You can also use a system variable (`${system(var.name)}`) or environment variable (`${env(VAR_NAME)}`).

- `<librariesFolder name="path/libraryFolderPath" scope="global">` - To point to multiple libraries located in the specified folder. Notice that the value of libraryFolder name includes the path (relative or absolute) to the library folder.

Both elements support the @scope attribute that defines the loading priority. It can have one of the following two values:

- `local` - The library is loaded in the plugin's own class loader. This is the default behavior.
- `global` - The library is loaded in the main application class loader as the last library in the list (as if it would be present in the application lib directory).

Installing an Oxygen XML Editor Plugin

Choose one of the following methods to install a plugin (on page 3222) in Oxygen XML Editor:

Manual Method

To manually install a plugin in Oxygen XML Editor, follow these steps:

1. Go to the Oxygen XML Editor installation directory and locate the plugins directory.

Note:
The plugins directory contains all the plugins available to Oxygen XML Editor.

2. In the plugins directory, create a subfolder to store the plugin files (for example, `{OXYGEN_INSTALL_DIR}/plugins/myPlugin`).
3. In the new folder, place the plugin descriptor file (plugin.xml), the Java classes of the plugin, and the other files that are referenced in the descriptor file.
4. Restart Oxygen XML Editor.

**Automatic Method**

To install an add-on that is hosted on a remote update site, follow these steps:

1. Go to Help > Install new add-ons.
2. In the displayed dialog box, enter or paste the update site that hosts the add-on in the Show add-ons from field (or select it from the drop-down menu, if applicable). The default add-ons are hosted on https://www.oxygenxml.com/InstData/Addons/default/updateSite.xml. If you want to see a list of all the default and sample add-ons that are available on the Oxygen remote update sites, choose ALL AVAILABLE SITES from the drop-down menu. The add-ons list contains the name, status, update version, Oxygen XML Editor version, and the type of the add-on (either framework, or plugin). A short description of each add-on is presented under the add-ons list.

**Note:**
To see all the versions of the add-ons, deselect Show only compatible add-ons and Show only the latest version of the add-ons. Incompatible add-ons are shown only to acknowledge their presence on the remote update site, but you cannot install an incompatible add-on.

3. Choose the add-ons you want to install, click the Next button, then follow the on-screen instructions.

**Note:**
Accepting the license agreement of the add-on is a mandatory step in the installation process.

**Note:**
All add-ons are installed in the extensions directory inside the Oxygen XML Editor preferences directory (on page 125).

**Tip:**
As an alternate approach, you can add an Install button to a web page that links to a URL that has the syntax https://host/path/to/updateSite.xml?oxygenAddonId=addOnIDValue and drop the button into the application's main editing area.

**Types of Plugin Extensions Available with the SDK**

A plugin (on page 3222) can have one or more defined plugin extensions that provide functionality to the application. This section presents the plugin extensions that are available.
Workspace Access Plugin Extension

This type of plugin (on page 3222) allows you to contribute actions to the main menu and toolbars, create custom views, interact with the application workspace, make modifications to opened documents, and add listeners for various events.

Many complex integrations (such as integrations with Content Management Systems) usually require access to some workspace resources such as toolbars, menus, views, and editors. This type of plugin is also useful because it allows you to make modifications to the XML content of an open editor.

The plugin must implement the `ro.sync.exml.plugin.workspace.WorkspaceAccessPluginExtension` interface. The callback method `applicationStarted` of this interface allows access to a parameter of the `ro.sync.exml.workspace.api.standalone.StandalonePluginWorkspace` type (allows for API access to the application workspace).

The `StandalonePluginWorkspace` interface has three methods that can be called to customize toolbars, menus, and views:

**addToolbarComponentsCustomizer**

Contributes to or modifies existing toolbars. You can specify additional toolbar IDs in the associated `plugin.xml` descriptor file using the following construct:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plugin SYSTEM "../plugin.dtd">
<plugin name="CustomWorkspaceAccess" .............>
 <runtime>........</runtime>
</plugin>

<extension type="WorkspaceAccess" ............./>

<toolbar id="SampleID" initialSide="NORTH" initialRow="1"/>
</extension>
```

The `<toolbar>` element adds a toolbar in the Oxygen XML Editor interface and allows you to contribute your own plugin-specific actions. The following attributes are supported:

- **id** - Unique identifier for the toolbar.
- **initialSide** - Specifies the place where the toolbar is initially displayed. The allowed values are `NORTH` and `SOUTH`.
- **initialRow** - Specifies the initial row on the specified side where the toolbar is displayed. For example, the first toolbar has an initial row of `0` and the next toolbar has an initial row of `1`. 
The `ro.sync.exml.workspace.api.standalone.ToolbarInfo` toolbar component information with the specified ID will be provided to you by the customizer interface. Therefore, you will be able to provide Swing components that will appear on the toolbar when the application starts.

`addViewComponentCustomizer`

Contributes to or modifies existing views, or contributes to the reserved custom view. You can specify additional view IDs in the associated `plugin.xml` descriptor using the following construct:

```xml
<view id="SampleID" initialSide="WEST" initialRow="0"/>
```

The `<view>` element adds a view in the Oxygen XML Editor interface and allows you to contribute your own plugin-specific UI components. The following attributes are supported:

- **id** - Unique identifier of the view component.
- **initialSide** - Specifies the place where the view is initially displayed. The allowed values are: NORTH, SOUTH, EAST, and WEST.
- **initialRow** - Specifies the initial row on the specified side where the view is displayed. For example, in Oxygen XML Editor, the Project view (on page 402) has an initial row of 0 and the Outline view (on page 539) has an initial row of 1. Both views are in the WEST part of the workbench.
- **initialState** - Specifies the initial state of the view. The allows values are: hidden, docked, autohide, and floating. By default, the view is visible and docked.

The `<view>` element also supports an optional `<perspective>` child element so that you can show or hide a certain view for a specified perspective. The `<perspective>` element supports the following attributes:

- **id** (required) - Unique identifier for the perspective. The possible values are: editor, dita, xslt_debugger, xquery_debugger, and database.
- **initState** (optional) - Specifies the initial state of the perspective. The allows values are: hidden, docked, autohide, and floating. By default, the view is visible and docked.
The `ro.sync.exml.workspace.api.standalone.ViewInfo` view component information with the specified ID will be provided to you by the customizer interface. Therefore, you will be able to provide Swing components that will appear on the view when the application starts.

**addMenuBarCustomizer**

Contributes to or modifies existing menu components.

Access to the open editors can be done by first getting access to all URLs opened in the workspace using the `StandalonePluginWorkspace.getAllEditorLocations(int editingArea)` API method. There are two available editing areas: the DITA Maps Manager editing area and the main editing area. Using the URL of an open resource, you can gain access to it using the `StandalonePluginWorkspace.getEditorAccess(URL location, int editingArea)` API method. A `ro.sync.exml.workspace.api.editor.WSEditor` then allows access to the current editing page.

A special editing API is supported for the **Text** mode (`ro.sync.exml.workspace.api.editor.page.text.WSTextEditorPage`) and the **Author** mode (`ro.sync.exml.workspace.api.editor.page.author.WSAuthorEditorPage`).

To be notified when editors are opened, selected, and closed, you can use the `StandalonePluginWorkspace.addEditorChangeListener` API method to add a listener.

**Examples:**

- A simple Maven-based sample of a workspace access plugin is available here: [https://github.com/oxygenxml/sample-plugin-workspace-access](https://github.com/oxygenxml/sample-plugin-workspace-access).
- A more complex sample of a workspace access plugin mimicking a CMS integration is available in the Author SDK: [https://www.oxygenxml.com/oxygen_sdk.html](https://www.oxygenxml.com/oxygen_sdk.html).

**Example: Adding a Custom View in Oxygen XML Editor**

To add a custom view in Oxygen XML Editor, follow this procedure:

1. Locate the `plugin.xml` descriptor file for your plugin (should be located inside the `plugins` folder, for example, `[OXYGEN_INSTALL_DIR]/plugins/myPlugin`). Define the ID of the view you want to add and specify the location where it will be placed:

   ```
   <view id="SampleWorkspaceAccessID" initialSide="WEST" initialRow="0"/>
   ```

2. In your `Workspace Access Plugin Extension (on page 2415)` implementation, where the `applicationStarted` callback is received, add a view component customizer like this:

   ```java
   pluginWorkspaceAccess.addViewComponentCustomizer(new ViewComponentCustomizer() {
       public void customizeView(ViewInfo viewInfo) {
           if (viewInfo.getViewID().equals("SampleWorkspaceAccessID")) {
               cmsMessagesArea = new JTextArea("CMS Session History:");
           }
   }
   ```
3. Define the `cmsMessagesArea` as a static field (if you can access the messages area from anywhere in your code).

Related Information:
https://www.oxygenxml.com/oxygen_sdk/oxygen_standalone_plugins.html

**Workspace Access Plugin Extension (JavaScript-Based)**

This is a JavaScript-based plugin (on page 3222) extension that allows you to contribute actions to the main menu and toolbars, create custom views, interact with the application workspace, make modifications to opened documents, and add listeners for various events.

This extension can use the same API as the Workspace Access plugin extension (on page 2415), but the implementation is JavaScript-based and uses the bundled Rhino library to create and work with Java API from the JavaScript code.

First, you need to create a custom plugin folder inside the plugins folder (for example, `[OXYGEN_INSTALL_DIR]/plugins/myPlugin`). This folder will contain your custom plugin descriptor file (`plugin.xml`) and all other resources for the plugin.

The plugin descriptor file (named `plugin.xml`) needs to reference a JavaScript file, as in the following example:

```xml
<!DOCTYPE plugin PUBLIC "-//Oxygen Plugin" "../plugin.dtd">
<plugin
    id="unique.id.value"
    name="Add Action To DITA Maps Manager popup-menu"
    description="Plugin adds action to DITA Maps Manager contextual menu."
    version="1.0"
    vendor="Syncro Soft"
    class="ro.sync.exml.plugin.Plugin"
    classLoaderType="preferReferencedResources">
    <extension type="WorkspaceAccessJS" href="wsAccess.js"/>
</plugin>
```

In the example above, the JavaScript file `wsAccess.js` (located in your custom plugin folder (on page 2418)) will be called. This JavaScript file needs to have two JavaScript methods defined inside. Methods that will be called when the application starts and when it ends:
With regard to the `applicationStarted` callback, besides using the `ro.sync.exml.workspace.api.standalone.StandalonePluginWorkspace` API with the `pluginWorkspaceAccess` parameter, you can also use a globally defined field called `jsDirURL` that points to the folder where the JavaScript file is located.

Below is a much larger example with a JavaScript Workspace Access plugin extension implementation that adds a new action in the contextual menu of the DITA Maps Manager view ([on page 2897](on page 2897)). The action starts the `notepad.exe` application and passes the reference to the currently selected `<topicref>` to it.
Packages.java.lang.System.err.println("Computed absolute reference "+ absoluteRef);

mi = new Packages.javaw.swing.JMenuItem("Run notepad");
        popUp.add(mi);
        actionPerfObj = {
            actionPerformed: function (e) {
                try {
                    Packages.java.lang.Runtime.getRuntime().exec("notepad.exe "+ pluginWorkspaceAccess.getUtilAccess().locateFile(absoluteRef));
                } catch (e1) {
                    e1.printStackTrace();
                }
            }
        }
        mi.addActionListener(new JavaAdapter(Packages.java.awt.event.ActionListener, actionPerfObj));
        }
        catch (el) {
            Packages.java.lang.System.err.println(el);
        }
    }
}

ditaMapPage.setPopUpMenuCustomizer(new Packages.ro.sync.exml.workspace.api.editor.page.ditamap.DITAMapPopupMenuCustomizer(customizerObj));

edChangedListener = new JavaAdapter(Packages.ro.sync.exml.workspace.api.listeners.WSEditorChangeListener, edChangedListener);
pluginWorkspaceAccess.addEditorChangeListener(edChangedListener,
    Packages.ro.sync.exml.workspace.api.PluginWorkspace.DITA_MAPS_EDITING_AREA);

function applicationClosing(pluginWorkspaceAccess) {
    Packages.java.lang.System.err.println("Application closing "+ pluginWorkspaceAccess);
}
Declaring Multiple Modules

JavaScript-based plugins can include multiple modules of JavaScript files in the plugin. In those files, you can declare functions that can be used in the main WorkspaceAccessJS JavaScript file. Thus, you can use those external script files as a library of functions. The modules must be declared in the plugin descriptor file (plugin.xml).

For example:

```xml
<!DOCTYPE plugin PUBLIC "-//Oxygen Plugin" "../plugin.dtd">
<plugin
  id="unique.id.value"
  name="Add Action To DITA Maps Manager popup-menu"
  description="Plugin adds action to DITA Maps Manager contextual menu."
  version="1.0"
  vendor="Syncro Soft"
  class="ro.sync.exml.plugin.Plugin"
  classLoaderType="preferReferencedResources">
  <extension type="WorkspaceAccessJS" href="wsAccess.js"/>
  <extension type="WorkspaceAccessJSModule" href="wsAccessModule1.js"/>
  <extension type="WorkspaceAccessJSModule" href="wsAccessModule2.js"/>
</plugin>
```

For more information and some samples, see GitHub Project with Multiple Workspace Access JavaScript-Based Plugin Samples.

Trusted Hosts Plugin Extension

This type of plugin (on page 3222) can be used by developers to automatically allow or reject remote connections that Oxygen XML Editor would normally ask the user for confirmation.

The name of the plugin extension is TrustedHosts. For security reasons, Oxygen XML Editor intercepts all connections to remote hosts and displays a dialog box that asks the user for confirmation. By implementing this plugin extension, the application will automatically allow or deny connections from websites you consider and configure as trusted or untrusted.

To develop an integration project, follow these steps:

- Copy the oxygen.jar file from [OXYGEN_INSTALL_DIR]/lib to the lib folder of your project.
- Implement the ro.sync.exml.plugin.workspace.security.TrustedHostsProviderExtension extension point.
- In the plugin descriptor file, define the <extension> element that points to the implementation of your classes:

  ```xml
  <extension type="TrustedHosts" class="my.trusted.hosts.provider.class.qualified.name"/>
  ```

Detailed information regarding the accepted or rejected connections from plugins are logged in the Information view (on page 512).
Example implementation:

```java
import ro.sync.exml.plugin.workspace.security.Response;
import ro.sync.exml.plugin.workspace.security.TrustedHostsProviderExtension;

public class DummyTrustedHostsProviderImpl implements TrustedHostsProviderExtension {
    @Override
    public Response isTrusted(String hostName) {
        // Connections from this website will always be
        // considered safe and always accepted.
        if ("trusted.website:80".equals(hostName)) {
            return TRUSTED;
        } else if("malicious.website:80".equals(hostName)) {
            // Always reject connections from malicious website
            return UNTRUSTED;
        }
        // All other connections are unknown, so a dialog will
        // appear and ask user's confirmation
        // to allow or deny the connection to this website.
        return UNKNOWN;
    }
}
```

**Author Stylesheet Plugin Extension**

This type of plugin (on page 3222) allows you to add a stylesheet (CSS or LESS) that renders elements in Author mode.

To specify additional stylesheets, edit the plugin descriptor and add `<extension>` elements that point to them, as in the following example:

```xml
<extension type="AuthorStylesheet" href="showTables.css"/>
<extension type="AuthorStylesheet" href="hideButtons.css"/>
```

Using this mechanism, you can add one or more CSS stylesheets to merge with the existing ones. Whenever you add a new stylesheet using this plugin, it will have priority over all other stylesheets applied on the file edited in Author mode.

If your implementation requires more flexibility (such as a dynamic change of the stylesheet), you should consider using the StylesFilter plugin extension (on page 2430).

**Additional Framework Plugin Extension**

This type of plugin (on page 3222) allows you to add a new framework straight from the plugin.
To specify additional frameworks, edit the *plugin* descriptor and add `<extension>` elements that point to them, as in the following example:

```xml
<extension type="AdditionalFrameworks" path="framework_directory"/>
```

The path attribute should be a sub-directory of the plugin. If the *plugin* is installed as an add-on (on page 120), the new framework will be set as read-only and editing it will only be possible if you duplicate it (on page 138). If the *plugin* is installed in the `[OXYGEN_INSTALL_DIR]/plugins` directory, the new frameworks will be editable.

**Additional XProc Engine Plugin Extension**

This type of *plugin* (on page 3222) contributes a folder that contains an external XProc engine.

The name of the *plugin* extension is *AdditionalXProcEngine* and it makes it easier to integrate an external XProc engine (on page 1534). After the plugin is installed, when you run an XProc transformation scenario, your external XProc engine can be selected from the *Processor* drop-down menu in the XProc tab.

An example of the *plugin.xml* file looks like this:

```xml
<plugin
  id="morgana.xproc.addon"
  name="Contribute Morgana XProc"
  description="Contribute Morgana XProc"
  version="1.0"
  vendor="Syncro Soft"
  class="ro.sync.exml.plugin.Plugin"
  classLoaderType="preferReferencedResources">...
  <extension type="AdditionalXProcEngine" path="MorganaXProcEngine/"/>
</extension>
</plugin>
```

The *path* attribute points to the XProc engine folder that contains the *engine.xml* and all the libraries necessary to run it.

**Components Validation Plugin Extension**

This type of *plugin* (on page 3222) allows you to customize the menus, toolbars, and other components by enabling or filtering them from the user interface.

This *plugin* provides the following API:

*ComponentsValidatorPluginExtension* interface

There is one method that must be implemented:

```
getComponentsValidator()
```

Returns a `ro.sync.exml.ComponentsValidator` implementation class used for validating the menus, toolbars, and their actions.
ComponentsValidator interface

Provides methods to filter various features from being added to the GUI of Oxygen XML Editor:

validateMenuOrTaggedAction(String[] menuOrActionPath)
Checks if a menu or a tag action from a menu is allowed and returns a boolean value. A tag is used to uniquely identifying an action. The String[] argument is the tag of the menu / action and the tags of its parent menus if any.

validateToolbarTaggedAction(String[] toolbarOrAction)
Checks if an action from a toolbar is allowed and returns a boolean value. The String[] argument is the tag of the action from a toolbar and the tag of its parent toolbar if any.

validateComponent(String key)
Checks if the given component is allowed and returns a boolean value. The String argument is the tag identifying the component. You can remove toolbars entirely using this callback.

validateAccelAction(String category, String tag)
Checks if the given accelerator action is allowed to appear in the GUI and returns a boolean value. An accelerator action can be uniquely identified so it will be removed both from toolbars or menus. The first argument represents the action category, the second is the tag of the action.

validateContentType(String contentType)
Checks if the given content type is allowed and returns a boolean value. The String argument represents the content type. You can instruct Oxygen XML Editor to ignore content types such as text / xsl or text / xquery.

validateOptionPane(String optionPaneKey)
Checks if the given options page can be added in the preferences option tree and returns a boolean value. The String argument is the option pane key.

validateOption(String optionKey)
Checks if the given option can be added in the option page and returns a boolean value. The String argument is the option key. This method is mostly used for internal use and it is not called for each option in a preferences page.

validateLibrary(String library)
Checks if the given library is allowed to appear listed in the About dialog box and returns a boolean value. The String argument is the library. This method is mostly for internal use.

validateNewEditorTemplate(EditorTemplate editorTemplate)
Checks if the given template for a new editor is allowed and returns a boolean value. The EditorTemplate argument is the editor template. An EditorTemplate is used to create an editor for a given extension. You can thus filter what appears in the list of the New dialog box.

isDebuggerPerspectiveAllowed()

Checks if the debugger perspective (on page 3221) is allowed and returns a boolean value.

validateSHMarker(String marker)

Checks if the given marker is allowed and returns a boolean value. The String argument represents the syntax highlight marker to be checked. If you decide to filter certain content types, you can also filter the syntax highlight options so that the content type is no longer present in the Preferences options tree.

validateToolbarComposite(String toolbarCompositeTag)

Checks if the toolbar composite is available. A toolbar composite is a toolbar component such as a drop-down menu.

Tip:
The best way to decide what to filter is to observe the values that Oxygen XML Editor passes when these callbacks are called. You have to create an implementation for this interface that lists in the console all values received by each function. Then you can decide on the values to filter and act accordingly.

Contribute Additional Languages Plugin Extension

This type of plugin (on page 3222) allows you to contribute new translation languages to the Oxygen XML Editor UI.

The AdditionalUITranslation plugin extension provides the ability to contribute new translation languages to the interface in Oxygen XML Editor.

A sample plugin.xml file looks like this:

```xml
<plugin
  id="romanian.i18n.provider"
  name="Add Romanian as an user interface language"
  description="Add Romanian as an user interface language"
  version="1.0"
  vendor="Syncro Soft"
  class="ro.sync.exml.plugin.Plugin">
  <extension type="AdditionalUITranslation" href="translation.xml"/>
</plugin>
```

where the translation.xml has a structure like this:
If all error keys are not translated in the custom translation.xml contributed by the plugin, the fallback is the default English translation. Once this plugin is installed, the Languages drop-down menu in the Options > Preferences > Global (on page 126) will be updated to include the newly added languages. The end-user will still need to select that language in the drop-down menu to use it.

Contribute External DITA-OT Distribution Plugin Extension

This type of plugin (on page 3222) allows you to contribute an external DITA-OT distribution to Oxygen XML Editor.

Oxygen XML Editor comes bundled with DITA-OT version 3.7.0. If you want to use a different DITA-OT version, the AdditionalDITAOT plugin extension provides the ability to contribute an external distribution of the DITA Open Toolkit to Oxygen XML Editor.

Example

For instance, if you wanted to use a DITA-OT version 1.8, your plugin.xml file might look like this:

```xml
<plugin
   id="dita-ot-18"
   name="Contribute DITA-OT 1.8"
   description="Contributes DITA-OT 1.8"
   version="1.0"
   vendor="Syncro Soft"
   class="ro.sync.exml.plugin.Plugin">  
   <extension type="AdditionalDITAOT" path="DITA-OT1.8.5" description="DITA-OT 1.8"/>
</plugin>
```

The @path attribute points to a folder located relative to the plugin.xml file and this folder is where the additional distribution of DITA-OT would be located.

When Oxygen XML Editor is started with this plugin enabled, that addition DITA-OT version can now be selected in the DITA Open Toolkit option in the DITA preferences page (on page 269).
Custom Protocol Plugin Extension

This type of plugin (on page 3222) allows you to work with a custom designed protocol for retrieving and storing files.

It provides the following API:

**URLStreamHandlerPluginExtension interface**

There is one method that must be implemented:

*getURLStreamHandler(String protocol)*

It takes as an argument the name of the protocol and returns a URLStreamHandler object, or null if there is no URL handler for the specified protocol.

This type of plugin extension can be usually combined with a Workspace Access plugin extension (on page 2415) that can add a custom toolbar with custom actions for opening documents from a certain source.

As an alternative, two older plugin extensions can also be used to add a toolbar action for showing a custom URL chooser:

**URLChooserPluginExtension2 interface**

Makes it possible to create your own dialog box that works with the custom protocol. This interface provides two methods:

*chooseURLs(StandalonePluginWorkspace workspaceAccess)*

Returns a URL[] object that contains the URLs the user decided to open with the custom protocol. You can invoke your own URL chooser dialog box here and then return the chosen URLs having your own custom protocol. You have access to the workspace of Oxygen XML Editor.

*getMenuName()*

Returns a String object that is the name of the entry added in the File menu.

**URLChooserToolbarExtension interface**

Makes it possible to provide a toolbar entry that is used for launching the custom URLs chooser from the URLChooserPluginExtension implementation. This interface provides two methods:

*getToolbarIcon()*

Returns the javax.swing.Icon image used on the toolbar.

*getToolbarTooltip()*

Returns a String that is the tooltip used on the toolbar.

Lock Handler Plugin Extension

This type of plugin extension (on page 3222) is used for locking resources from a specific protocol.
It provides the following API:

**LockHandlerFactoryPluginExtension interface**

You need to implement the following two methods:

**LockHandler getLockHandler()**

Gets the lock handler for the current handled protocol. Might be `null` if not supported.

**boolean isLockingSupported(String protocol)**

Checks if a lock handler can be provided for a specific protocol.

To use this type of extension in your plugin, create an extension of `LockHandlerFactory` type in your `plugin.xml` file and specify the class implementing `LockHandlerFactoryPluginExtension`:

```xml
<extension type="LockHandlerFactory"
    class="LockHandlerFactoryPluginExtensionImpl"/>
```

**Open Redirect Plugin Extension**

This type of plugin (on page 3222) is useful for opening multiple files with only one open action.

For example, when a zip archive or an ODF file or an OOXML file is open in the Archive Browser view (on page 2008) a plugin of this type can decide to open a file also from the archive in an XML editor panel. This file can be the `document.xml` main file from an OOXML file archive or a specific XML file from a zip archive.

The plugin must implement the interface `OpenRedirectExtension`. It only has one callback: `redirect(URL)` that receives the URL of the file opened by the Oxygen XML Editor user. If the plugin decides to open also other files it must return an array of information objects (`OpenRedirectInformation[]`) that correspond to these files. Such an information object must contain the URL that is opened in a new editor panel and the content type (for example, `text/xml`). The content type is used for determining the type of editor panel. A `null` content type allows auto-detection of the file type.
Option Page Plugin Extension

This type of plugin extension (on page 3222) allows you to add custom Preferences pages.

The extension must implement the ro.sync.exml.plugin.option.OptionPagePluginExtension class. The provided callbacks allow you to create a custom Swing component that will be added to the page and to react to various calls to persistently save the page settings using the OptionsStorage API.

All preferences pages that are contributed by a plugin are listed in the Preferences dialog box in the Plugins category. As long as the added preferences page has the same name as its plugin, it will be promoted to the first level of the hierarchy within the Plugins category.

The plugin.xml configuration file can specify one or more such extensions using constructs like this:

```xml
<extension type="OptionPage" class="my.pack.CustomOptionPagePluginExtension"/>
```

Sharing Options Through Project Files

To share options that are configured in certain plugin preferences pages, you can store them in a project file (.xpr file extension) that can easily be shared with others. To do this you must:

1. Override ro.sync.exml.plugin.option.OptionPagePluginExtension.getProjectLevelOptionKeys() and return a set of options that need to be saved inside the project.
2. Install the plugin in an Oxygen XML Editor instance. (on page 2413)
3. In the Project view (on page 402), create a project or open an existing one.
4. Open the Preferences dialog box (Options > Preferences) (on page 124).
5. Configure the options in each preferences page that you want to be included in the project file and switch the storage preference to Project Options (on page 3222) in each page.

Note:
Some pages do not have the Project Options button, since the options they host might contain sensitive data (such as passwords, for example) that is unsuitable for sharing with other users.

6. Click OK and close the Preferences dialog box.

All explicitly set values are now saved in the project file. You can then share the project file so that your team will have the same option configuration that you stored in the project file.

Note:
The project file extension (.xpr) must be preserved when the file is distributed to others.

Notice:
When a project is opened for the first time, a confirmation dialog box will be displayed that asks you to confirm that the project came from a trusted source. This is meant to help prevent potential security issues.
**Resource Locking Custom Protocol Plugin Extension**

This type of plugin (on page 3222) allows you to work with a custom designed protocol for retrieving and storing files and it can lock a resource when opening it in Oxygen XML Editor.

This type of plugin extends the custom protocol plugin type with resource locking support and provides the following API:

* **URLStreamHandlerWithLockPluginExtension interface**

  The plugin receives callbacks following the simple protocol for resource locking and unlocking imposed by Oxygen XML Editor.

  There are two additional methods that must be implemented:

  - `getLockHandler()`
    - Returns a `LockHandler` implementation class with the implementation of the lock specific methods from the plugin.

  - `isLockingSupported(String protocol)`
    - Returns a `boolean` that is `true` if the plugin accepts to manage locking for a certain URL protocol scheme (such as `ftp`, `http`, `https`, or `customName`).

**Styles Filter Plugin Extension**

This type of plugin (on page 3222) allows you to dynamically modify the CSS styles used to render elements in the Author mode.

The plugin must extend the `ro.sync.exml.plugin.author.css.filter.GeneralStylesFilterExtension` class. This class has a callback on which you can alter the styles for an Author mode element.

This extension point is similar with the Styles Filter that you set at the framework (on page 3219) level. The only difference is that the plugin filters styles are used for any open XML document, regardless of the document type. The changes made by this plugin are prioritized over the changes made by the framework-level filter.

**Note:**

Alternatively, you can use the AuthorStylesheet plugin extension (on page 2422), which does not require any additional Java development and is compatible with Oxygen XML Web Author.

Related Information:

Customizing the CSS Styles Filter (on page 2262)
Targeted URL Stream Handler Plugin Extension

This type of plugin (on page 3222) can be used when it is necessary to impose custom URL stream handlers for specific URLs.

This plugin extension can handle the following protocols: http, https, ftp or sftp. Oxygen XML Editor usually provides specific fixed URL stream handlers. If it is set to handle connections for a specific protocol, this extension prompts for the URL stream handler for each open connection of a URL that has that protocol.

To use this type of plugin, you have to implement the ro.sync.exml.plugin.urlstreamhandler.TargetedURLStreamHandlerPluginExtension interface that provides the following methods:

boolean canHandleProtocol(String protocol)

This method checks if the plugin can handle a specific protocol. If this method returns true for a specific protocol, the getURLStreamHandler(URL) method will be called for each open connection of a URL having this protocol.

URLStreamHandler getURLStreamHandler(URL url)

This method provides the URL handler for the specified URL and it is called for each open connection of a URL with a protocol that has the canHandleProtocol(String) method return true.

If this method returns null, the default Oxygen XML Editor URLStreamHandler is used.

To use this type of extension in your plugin, create an extension of TargetedURLHandler type in your plugin.xml file and specify the class that implements TargetedURLStreamHandlerPluginExtension:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plugin SYSTEM "../plugin.dtd">
<plugin name="CustomTargetedURLStreamHandlerPlugin" .............>
  <runtime>
  ........
  </runtime>

  <extension type="TargetedURLHandler"
              class="CustomTargetedURLStreamHandlerPluginExtension"/>
  .............

</plugin>
```

This extension can be useful in situations when connections opened from a specific host must be handled in a particular way. For example, the Oxygen XML Editor HTTP URLStreamHandler may not be compatible for sending and receiving SOAP using the SUN Web Services implementation. In this case, you can override the stream handler (set by Oxygen XML Editor) to use the default SUN URLStreamHandler, since it is more compatible with sending and receiving SOAP requests.
public class CustomTargetedURLStreamHandlerPluginExtension
    implements TargetedURLStreamHandlerPluginExtension {

    @Override
    public boolean canHandleProtocol(String protocol) {
        boolean handleProtocol = false;
        if ("http".equals(protocol) || "https".equals(protocol)) {
            // This extension handles both HTTP and HTTPS protocols
            handleProtocol = true;
        }
        return handleProtocol;
    }

    @Override
    public URLStreamHandler getURLStreamHandler(URL url) {
        // This method is called only for the URLs with a protocol
        // where canHandleProtocol(String) method returns true (HTTP and HTTPS)

        URLStreamHandler handler = null;

        String host = url.getHost();
        String protocol = url.getProtocol();
        if ("some_host".equals(host)) {
            // When there are connections opened from some_host, the SUN HTTP(S)
            // handlers are used
            if ("http".equals(protocol)) {
                handler = (URLStreamHandler) Class.forName("sun.net.www.protocol.http.Handler")
                        .getConstructor(new Class[0]).newInstance(new Object[0]);
            } else {
                handler = (URLStreamHandler) Class.forName("sun.net.www.protocol.https.Handler")
                        .getConstructor(new Class[0]).newInstance(new Object[0]);
            }
        }
        return handler;
    }
}

XML Refactoring Operations Plugin Extension

This type of plugin (on page 3222) allows you to specify one or more directories where the XML Refactoring operation resources are loaded.
The `RefactoringOperationsProvider` extension can be used to specify the location where custom XML Refactoring operation resources (XQuery Update script or XSLT stylesheet and Operation Descriptor files) are stored. Oxygen XML Editor will scan the specified locations to load the custom operations when the XML Refactoring tool is opened, and allows you to share your custom refactoring operations.

**Example: XML Refactoring Operations Plugin Extension**

```xml
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plugin PUBLIC "-//Oxygen Plugin" ".//plugin.dtd">

<plugin
   id="refactoring.operations"
   name="Refactoring operations plugin"
   description="Contains operation descriptors and related scripts"
   version="1.0">
   <extension type="RefactoringOperationsProvider">
     <folder path="customDir/"/>
     <folder path="customDir2"/>
   </extension>
</plugin>
```

**XSLT Transformer Plugin Extension**

This type of plugin (on page 3222) allows you to add an external XSLT transformer plugin.

The name of the plugin is `Transformer` and it makes it easier to contribute your own implementation of the XSLT Processor. After the plugin is installed, you can create a new XML transformation with XSLT scenario (on page 1464) and select your external XSLT engine from `Transformer` drop-down menu in the XSLT tab.

To create an XSLT integration project, follow these steps:

- Copy the `oxygen.jar` file from `[OXYGEN_INSTALL_DIR]/lib` to the `lib` folder of your project.
- Copy the jars of your transformer to the `lib` folder of your project.
- Implement the `ro.sync.exml.plugin.transform.XSLTTransformerPluginExtension` interface.
- In the `plugin` descriptor file, define the `<extension>` element that points to the implementation of your classes:

```xml
<extension type="Transformer" class="my.xslt.plugin.extension"/>
```

**Validator Plugin Extension**

This type of plugin (on page 3222) allows you to add an external validation engine from a plugin.

The name of the plugin extension is `DocumentValidator` and it makes it possible to contribute your own implementation of a validation engine. After the plugin is installed, if you create a new validation scenario or edit an existing validation scenario to add a validation stage, you will find the name of the new engine contributed by the plugin in the `Validation Engine` combo box.
To create a plugin that implements the validator extension:

1. Implement the `ro.sync.exml.plugin.validator.ValidatorPluginExtension` interface in your plugin's libraries.
2. In the plugin descriptor file, define the `<extension>` element that points to the implementation of your classes:

   ```xml
   <extension type="DocumentValidator" class="my.validator.plugin.extension"/>
   ```

### Saxon XSLT Transformer Plugin Extension

This type of plugin (on page 3222) allows you to add an external Saxon XSLT transformer plugin.

The name of the plugin is **Transformer** and it makes it easier to contribute your own implementation of the Saxon XSLT Processor. After the plugin is installed, you can create a new XML transformation with XSLT scenario (on page 1464) and select your external Saxon engine from **Transformer** drop-down menu in the XSLT tab.

To create an XSLT integration project, follow these steps:

- Copy the `oxygen.jar` file from `{OXYGEN_INSTALL_DIR}/lib` to the `lib` folder of your project.
- Copy the Saxon jars to the `lib` folder of your project.
- Implement the `ro.sync.exml.plugin.transform.SaxonXSLTTransformerPluginExtension` interface.
- In the plugin descriptor file, define the `<extension>` element that points to the implementation of your classes, following example:

  ```xml
  <extension type="Transformer" class="my.saxon.xslt.plugin.extension"/>
  ```

An add-on that implements the Saxon XSLT transformer can be found here: Saxon XSLT and XQuery Transformer Add-on (on page 2563). For more information, see the Oxygen XML SDK Add-on Repositories web page.

### XQuery Transformer Plugin Extension

This type of plugin (on page 3222) allows you to add an external XQuery transformer plugin.

The name of the plugin is **XQueryTransformer** and it makes it easier to contribute your own implementation of the XQuery Processor. After the plugin is installed, you can create a new XQuery transformation scenario (on page 1536) and select your external XQuery engine from **Transformer** drop-down menu in the XQuery tab.

To create an XQuery integration project, follow these steps:

- Copy the `oxygen.jar` file from `{OXYGEN_INSTALL_DIR}/lib` to the `lib` folder of your project.
- Copy the jars of your transformer to the `lib` folder of your project.
- Implement the `ro.sync.exml.plugin.transform.XQueryTransformerPluginExtension` interface.
- In the plugin descriptor file, define the `<extension>` element that points to the implementation of your classes:

  ```xml
  <extension type="XQueryTransformer" class="my.xquery.plugin.extension"/>
  ```
Saxon XQuery Transformer Plugin Extension

This type of plugin (on page 3222) allows you to add the Saxon external XQuery transformer plugin.

The name of the plugin is XQueryTransformer and it makes it easier to contribute your own implementation of the Saxon XQuery Processor. After the plugin is installed, you can create a new XQuery transformation scenario (on page 1536) and select your Saxon external XQuery engine from Transformer drop-down menu in the XQuery tab.

To create an XQuery integration project, follow these steps:

• Copy the oxygen.jar file from [OXYGEN_INSTALL_DIR]/lib to the lib folder of your project.
• Copy the Saxon jars to the lib folder of your project.
• Implement the ro.sync.exml.plugin.transform.SaxonXQueryTransformerPluginExtension interface.
• In the plugin descriptor file, define the <extension> element that points to the implementation of your classes:

```
<extension type="XQueryTransformer" class="my.saxon.xquery.plugin.extension"/>
```

An add-on that implements the Saxon XQuery transformer can be found here: Saxon XSLT and XQuery Transformer Add-on (on page 2563). For more information, see the Oxygen XML SDK Add-on Repositories web page.

Plugin Extensions Designed to Work only in the Text Editing Mode

These plugin (on page 3222) extensions operate only when editing documents in the Text mode. They are mounted automatically by the application on the contextual menu in the Plugins submenu.

The Workspace Access Plugin Extension (on page 2415) offers an API that can be used to implement similar functionality for both Text and Author mode.

General Plugin Extension

This type of plugin (on page 3222) allows you to invoke custom code to interact with the workspace in Text mode.

This plugin is the most general plugin type and provides a limited API:

GeneralPluginExtension interface

Intended for general-purpose plugins, kind of external tools but triggered from the Plugins menu. The implementing classes must provide the method process(GeneralPluginContext), which must provide the plugin processing. This method takes as a parameter a GeneralPluginContext object.

GeneralPluginContext class

Represents the context in which the general plugin extension does its processing. The getPluginWorkspace() method allows you access to the workspace of Oxygen XML Editor.
Selection Plugin Extension

This type of plugin (on page 3222) allows you to manage selections of text.

A selection plugin can be applied to both XML and non-XML documents. The plugin is started by making a selection in the editor, then selecting the corresponding menu item from the Plugins submenu in the contextual menu of Text mode.

This plugin type provides the following API:

SelectionPluginExtension interface

The context containing the selected text is passed to the extension and the processed result is going to replace the initial selection. The process(GeneralPluginContext) method must return a SelectionPluginResult object that contains the result of the processing. The String value returned by the SelectionPluginResult object can include editor variables (on page 323) such as ${caret} and ${selection}.

SelectionPluginContext object

Represents the context and provides four methods:

- getSelection() - Returns a String that is the current selection of text.
- getFrame() - Returns a Frame that is the editing frame.
- getPluginWorkspace() - Returns access to the workspace of Oxygen XML Editor.
- getDocumentURL() - Returns the URL of the currently edited document.

Example - Uppercase Plugin

The following plugin (on page 3222) is called UppercasePlugin and is an example of a Selection plugin. (on page 2436). It is used in Oxygen XML Editor for capitalizing the characters in the current selection. This example consists of two Java classes and the plugin descriptor file (plugin.xml):

- UppercasePlugin.java:

```java
package ro.sync.sample.plugin.uppercase;

import ro.sync.exml.plugin.Plugin;
import ro.sync.exml.plugin.PluginDescriptor;

public class UppercasePlugin extends Plugin {
    /**
     * Plugin instance.
     */
```
private static UppercasePlugin instance = null;

/**
 * UppercasePlugin constructor.
 *
 * @param descriptor Plugin descriptor object.
 */
public UppercasePlugin(PluginDescriptor descriptor) {
    super(descriptor);

    if (instance != null) {
        throw new IllegalStateException("Already instantiated!");
    }
    instance = this;
}

/**
 * Get the plugin instance.
 *
 * @return the shared plugin instance.
 */
public static UppercasePlugin getInstance() {
    return instance;
}

• UppercasePluginExtension.java:

package ro.sync.sample.plugin.uppercase;

import ro.sync.exml.plugin.selection.SelectionPluginContext;
import ro.sync.exml.plugin.selection.SelectionPluginExtension;
import ro.sync.exml.plugin.selection.SelectionPluginResult;
import ro.sync.exml.plugin.selection.SelectionPluginResultImpl;

public class UppercasePluginExtension implements SelectionPluginExtension {
    /**
     * Convert the text to uppercase.
     *
     * @param context Selection context.
     * @return Uppercase plugin result.
     */

public SelectionPluginResult process(SelectionPluginContext context) {
    return new SelectionPluginResultImpl(
        context.getSelection().toUpperCase());
}

• plugin.xml:

```xml
<!DOCTYPE plugin SYSTEM "../plugin.dtd">
<plugin
    name="UpperCase"
    description="Convert the selection to uppercase"
    version="1.0.0"
    vendor="SyncRO"
    class="ro.sync.sample.plugin.uppercase.UppercasePlugin">
    <runtime>
        <library name="lib/uppercase.jar"/>
    </runtime>
    <extension type="selectionProcessor"
        class="ro.sync.sample.plugin.uppercase.UppercasePluginExtension"/>
</plugin>
```

Document Plugin Extension

This type of plugin (on page 3222) allows you to manage the current document.

The document plugin type can only be applied to an XML document. It can modify the current document that is received as a callback parameter.

The plugin is started by selecting the corresponding menu item from the Plugins submenu in the contextual menu of Text mode. It provides the following API:

**DocumentPluginExtension interface**

Receives the context object containing the current document. The process(GeneralPluginContext) method can return a DocumentPluginResult object containing a new document.

**DocumentPluginContext object**

Represents the context and provides three methods:

- **getDocument()** - Returns a javax.swing.text.Document object that represents the current document.
- **getFrame()** - Returns a java.awt.Frame object that represents the editing frame.
- **getPluginWorkspace()** - Returns access to the workspace of Oxygen XML Editor.
How to Write a CMS Integration Plugin

To have a complete integration between Oxygen XML Editor and a CMS, you usually have to write a plugin (on page 3222) that combines the following two available plugin extensions:

- Workspace Access (on page 2415)
- Custom protocol (on page 2444)

The usual set of requirements for an integration between Oxygen XML Editor and the CMS are as follows:

1. Contribute to the Oxygen XML Editor toolbars and main menu with your custom Check Out and Check In actions:
   - **Check Out** triggers your custom dialog boxes that allow you to browse the remote CMS and choose the resources you want to open.
   - **Check In** allows you to send the modified content back to the server.

   You can use the Workspace Access plugin extension (and provided sample Java code) for all these operations.

2. When Check Out is called, use the Oxygen XML Editor API to open your custom URLs (URLs created using your custom protocol). It is important to implement and use a Custom Protocol extension to be notified when the files are opened and saved and to be able to provide the content for the relative references the files may contain to Oxygen XML Editor. Your custom `java.net.URLStreamHandler` implementation checks out the resource content from the server, stores it locally and provides its content. Sample Check Out implementation:

```java
/**
 * Sample implementation for the "Check Out" method.
 *
 * @param pluginWorkspaceAccess (Workspace Access plugin).
 * @throws MalformedURLException
 */
private void checkOut(StandalonePluginWorkspace pluginWorkspaceAccess)
throws MalformedURLException {
    //TODO Show the user a custom dialog box for browsing the CMS
    //TODO after user selected the resource create a URL with a custom protocol
    //which will uniquely map to the resource on the CMS using the URLHandler
    //something like:
    URL customURL = new URL("mycms://host/path/to/file.xml");
    //Ask Oxygen to open the URL
    pluginWorkspaceAccess.open(customURL);
    //Oxygen will then your custom protocol handler to provide the contents for
    //the resource "mycms://host/path/to/file.xml"
    //Your custom protocol handler will check out the file in a temporary
```
directory, for example, and provide the content from it.
//Oxygen will also pass through your URLHandler if you have any relative
//references which need to be opened/obtained.

Figure 604. Check Out Process Diagram

The phases are:

- 1 - Browse CMS repository
- 2 - User chooses a resource
- 3 - Use API to open custom URL: mycms://path/to/file.xml
- 4 - Get content of URL: mycms://path/to/file.xml
- 5 - Get content of resource
- 6 - Store on disk for faster access
- 7 - Retrieve content from disk if already checked out
- 8 - Retrieved content

3. Contribute a special **Browse CMS** action to every dialog box in Oxygen XML Editor where a URL can be chosen to perform a special action (such as the **Reuse Content** or **Insert Image** action). Sample code:
```java
public void customizeBrowseActions(List<Action> existingBrowseActions, final InputURLChooser chooser) {

    //IMPORTANT, you also need to set a custom icon on the action
    //for situations when its text is not used for display.
    Action browseCMS = new AbstractAction("CMS") {
        public void actionPerformed(ActionEvent e) {
            URL chosenResource = browseCMSAndChooseResource();
            if (chosenResource != null) {
                try {
                    //Set the chosen resource in the combo box chooser.
                    chooser.urlChosen(chosenResource);
                } catch (MalformedURLException e1) {
                    //
                }
            }
        }
    }

    existingBrowseActions.add(browseCMS);
}
```

When inserting references to other resources using the actions already implemented in Oxygen XML Editor, the reference to the resource is made by default relative to the absolute location of the edited XML file. You can gain control over the way that the reference is made relative for a specific protocol like this:

```java
//Add a custom relative reference resolver for your custom protocol.
//Usually when inserting references from one URL to another Oxygen
//makes the inserted path relative.
//If your custom protocol needs special relativization techniques then
//it should set up a custom relative
//references resolver to be notified when resolving needs to be done.
pluginWorkspaceAccess.addRelativeReferencesResolver(
    //Your custom URL protocol that you already have a
    //custom URLStreamHandlerPluginExtension set up.
    "mycms",
    //The relative references resolver
    new RelativeReferenceResolver() {
        public String makeRelative(URL baseURL, URL childURL) {
            //Return the referenced path as absolute for example.
            return childURL.toString();
            //Or return null for the default behavior.
    ```
4. Write the plugin.xml descriptor file. Your plugin combines the two extensions using a single set of libraries. The descriptor would look like this:

```xml
<!DOCTYPE plugin SYSTEM "../plugin.dtd">
<plugin
  name="CustomCMSAccess"
  description="Test"
  version="1.0.0"
  vendor="ACME"
  class="custom.cms.CMSAccessPlugin">
  <runtime>
    <library name="lib/cmsaccess.jar"/>
  </runtime>
  <!--Access to add actions to the main menu and toolbars or to add custom views.-->  
  <!--See the "CustomWorkspaceAccessPluginExtension" Java sample for more details-->
  <extension type="WorkspaceAccess"
    class="custom.cms.CustomWorkspaceAccessPluginExtension"/>
  <!--The custom URL handler that will communicate with the CMS implementation-->
  <!--See the "CustomProtocolURLHandlerExtension" Java sample for more details-->
  <extension type="URLHandler"
    class="custom.cms.CustomProtocolURLHandlerExtension"/>
</plugin>
```

5. Create a cmsaccess.jar JAR (on page 3220) archive containing your implementation classes.

6. Copy your new plugin directory in the plugins subfolder of the Oxygen XML Editor install folder (for example, [OXYGEN_INSTALL_DIR]/plugins/myPlugin) and start Oxygen XML Editor.

Related Information:
https://github.com/oxygenxml/oxygen-cmis-plugin
https://github.com/axxepta/project-argon

Class Loading Issues
It is possible that the Java libraries you have specified in the plugin libraries list conflict with the ones already loaded by Oxygen XML Editor. To instruct the plugin to prefer its libraries over the ones used by Oxygen XML Editor, you can add the following attribute on the <plugin> root element:

```xml
<plugin classLoaderType="preferReferencedResources">
```
A **Late Delegation Class Loader** (the main class loader in Oxygen XML Editor) is a `java.net.URLClassLoader` extension that prefers to search classes in its own libraries list and only if a class is not found there to delegate to the parent class loader.

The main Oxygen XML Editor Class Loader uses as libraries all JARS specified in the `{OXYGEN_INSTALL_DIR}/lib` directory. Its parent class loader is the default JVM Class loader. For each plugin instance, a separate class loader is created having as parent the Oxygen XML Editor Class Loader.

The **plugin** class loader can be either a standard `java.net.URLClassLoader` or a `LateDelegationClassLoader` (depending on the attribute `classLoaderType` in the `plugin.xml`). Its parent class loader is always the Oxygen XML Editor `LateDelegationClassLoader`.

If you experience additional problems, such as:

```java
java.lang.LinkageError: ClassCastException:
    attempting to cast
    jar:file:/C:/jdk1.6.0_06/jre/lib/rt.jar!/javax/xml/ws/spi/Provider.class
    to jar:file:/D:/Program Files/Oxygen XML Editor 12/plugins/wspaccess/../../xdocs/lib/jaxws/jaxws-api.jar!/javax/xml/ws/spi/Provider.class
    at javax.xml.ws.spi.Provider.provider(Provider.java:94) at javax.xml.ws.Service.<init>(Service.java:56)
    ............................................
```

The cause could be the fact that some classes are instantiated using the context class loader of the current thread. The most straightforward fix is to write your code in a `try/finally` statement:

```java
ClassLoader oldClassLoader =
    Thread.currentThread().getContextClassLoader();
try {
    //This is the implementation of the
    //WorkspaceAccessPluginExtension plugin interface.
    Thread.currentThread().setContextClassLoader{
        CustomWorkspaceAccessPluginExtension.
            this.getClass().getClassLoader();
    } //WRITE YOUR CODE HERE
} finally {
    Thread.currentThread().
        setContextClassLoader(oldClassLoader);
}
```
How to Write A Custom Protocol Plugin

To create a custom protocol plugin (on page 3222), follow these steps:

1. Write the handler class for your protocol that implements the java.net.URLStreamHandler interface. Be careful to provide ways to encode and decode the URLs of your files.
2. Write the plugin class by extending ro.sync.exml.plugin.Plugin.
3. Write the plugin extension class that implements the ro.sync.exml.plugin=urlstreamhandler.URLStreamHandlerPluginExtension interface.

   It is necessary that the plugin extension for the custom protocol implements the URLStreamHandlerPluginExtension interface. Without it, you cannot use your plugin, because Oxygen XML Editor is not able to find the protocol handler.

   You can choose also to implement the URLChooserPluginExtension interface. It allows you to write and display your own customized dialog box for selecting resources that are loaded with the custom protocol.

   An implementation of the extension URLHandlerReadOnlyCheckerExtension allows you to:
   ◦ Mark a resource as read-only when it is opened.
   ◦ Switch between marking the resource as read-only and read-write while it is edited.

   It is useful when opening and editing CMS resources.

4. Write the plugin.xml descriptor file.

   Remember to set the name of the plugin class to the one from the second step and the plugin extension class name with the one you have chosen at step 3.

5. Create a JAR archive with all these files.

6. Create a custom plugin folder inside the plugins folder (for example, \OXYGEN_INSTALL_DIR\plugins/myPlugin) that contains your new plugin.

How to Share a Class Loader Between a Framework and Plugin

In some cases you may need to extend the functionality of Oxygen XML Editor both through a framework (on page 3219) and through a plugin (on page 3222). Normally, a framework and a plugin both run in their own private classloader. If the framework and the plugin use the same JAVA extensions/classes, it is recommended that they share the same classloader. This way, the common classes are loaded by only one Class Loader and they will both use the same static objects and have the ability to cast objects between one another.

To do this, open the Preferences dialog box (Options > Preferences) (on page 124), go to Document Type Association, select the document type, go to the Classpath tab, and in the Use parent classloader from plugin with ID fields introduce the ID of the plugin. This ID is declared in the configuration file of the plugin (on page 2412).
If you have created the framework using a *Framework Extension Script*, then edit the script and specify the plugin ID on the classpath (*on page 2138*).

**Important:**
The shared classes must be specified only in the configuration files of the *plugin*, and not in the configuration file and the document type class path at the same time.

## Packing and Deploying Plugins as Add-ons

In Oxygen XML Editor, a *plugin* can be packed and deployed as an *add-on* to provide additional functionality to the application.

### Packing a Plugin as an Add-on

This procedure is suitable for developers who want a better control over the *add-on* (*on page 3222*) package or those who want to automate some of the steps:

1. Pack the *plugin* (*on page 3222*) as a ZIP file or a *Java Archive* (*on page 3220*). Note that you should pack the entire root directory not just its contents.
2. **[Optional]** If you created a *Java Archive* at the previous step, digitally sign the package. You will need a certificate signed by a trusted authority. To sign the *JAR*, you can either use the *jarsigner* command-line tool inside Oracle’s Java Development Kit ([JDK_DIR]/bin/jarsigner.exe) or if you are working with *Apache Ant* (*on page 3216*), you can use the *signjar* task (a front for the *jarsigner* command-line tool). The benefit of having a signed add-on is that you can verify the integrity of the add-on issuer. If you do not have such a certificate, you can generate one yourself using the *keytool* command-line utility.

**Note:**
This approach is recommended for tests since anyone can create a self-signed certificate.

3. Create a descriptor file. You can use a template that Oxygen XML Editor provides by going to **File > New** and selecting the *Oxygen add-ons update site* template. The products the add-on is compatible with can be specified in the template. Once deployed, this descriptor file is referenced as *update site*.

### Alternate Packing Method: Add-ons Packager

Alternatively, you can use the *Add-ons Packager* plugin by following this procedure:

1. Install the *Add-ons Packager* plugin following the procedure described in **Installing Add-ons** (*on page 120*). In the displayed dialog box, enter or paste [https://www.oxygenxml.com/InstData/Addons/optional/updateSite.xml](https://www.oxygenxml.com/InstData/Addons/optional/updateSite.xml) in the *Show add-ons from* field.
2. Restart Oxygen XML Editor. If the add-on is correctly installed, the *Add-ons packager* toolbar action is available.
3. Invoke the **Add-ons packager** toolbar action and input the required information in the displayed dialog box.
4. Click **OK** to complete the packaging process.

**Deploying an Add-on**

To deploy an add-on, copy the **ZIP** or **Java Archive** file and the descriptor file to an HTTP server. The URL to this location serves as the **Update Site URL**.

**Testing Plugins and Java Extensions**

In the various procedures for creating a **plugin** (on page 3222), you are usually instructed to copy your plugin folder to the `{OXYGEN_INSTALL_DIR}/plugins/` directory. If you want to test the code in your plugin without copying it to that folder, follow this procedure:

1. Create a file called `plugin.redirect` that contains the full file path references to your project (for example, `C:\Users\john_doe\Documents\sample-plugin-folder`).
2. Save that file in any folder (for example, called `sample_test_folder`) inside the `{OXYGEN_INSTALL_DIR}/plugins/` directory.

**Step Result:** Oxygen XML Editor will automatically load the plugin from your project location.
3. Now you can modify the Java code, the IDE will automatically compile it, and if the **plugin.xml** file has a classpath reference to the compiled classes folder, you can restart Oxygen XML Editor and test your changes.

**Creating and Running Automated Tests**

If you have developed complex custom **plugins** (on page 3222) or **frameworks** (on page 3219) (document types), the best way to test your implementation and ensure that further changes will not interfere with the current behavior is to make automated tests for your customization.

An Oxygen XML Editor standalone installation includes a main **oxygen.jar** library located in the `{OXYGEN_INSTALL_DIR}`. That **JAR** (on page 3220) library contains a base class for testing developer customizations named: `ro.sync.exml.workspace.api.PluginWorkspaceTCBase`.

To develop **JUnit** tests for your customizations using the Eclipse workbench, follow these steps:
1. Create a new Eclipse Java project and copy the entire contents of the \{OXYGEN_INSTALL_DIR\} folder to the new project.

2. Add all JAR libraries present in the \{OXYGEN_INSTALL_DIR\}/lib directory to the Java Build Path -> Libraries tab. Make sure that the main JAR library oxygen.jar or oxygenAuthor.jar is the first one in the Java classpath by moving it up in the Order and Export tab.

3. Click Add Library and add the JUnit and JFCUnit libraries.

4. Create a new Java class that extends ro.sync.exml.workspace.api.PluginWorkspaceTCBase.

5. Pass the following parameters to the constructor of the super class:
   - **File installationFolder** - The file path to the main application installation directory. If not specified, it defaults to the folder where the test is started.
   - **File frameworksFolder** - The file path to the frameworks directory. It can point to a custom framework directory where it resides.
   - **File pluginsFolder** - The file path to the plugins directory. It can point to a custom plugin directory where it resides.
   - **File optionsFolder** - The folder that contains the application options. If not specified, the application will auto-detect the location based on the started product ID.
   - **String licenseKey** - The license key used to license the test class.
   - **int productId** - The ID of the product and should be one of the following:

     - PluginWorkspaceTCBase.XML_AUTHOR_PRODUCT
     - PluginWorkspaceTCBase.XML_EDITOR_PRODUCT
     - PluginWorkspaceTCBase.XML_DEVELOPER_PRODUCT

6. Create test methods that use the API in the base class to open XML files and perform various actions on them. Your test class could look something like this:

   ```java
   public class MyTestClass extends PluginWorkspaceTCBase {

   /**
   * Constructor.
   */
   public MyTestClass() throws Exception {
       super(null, new File("frameworks"), new File("plugins"), null,
               "------START-LICENSE-KEY------\n" +
               "\n" +
               "Registration_Name=Developer\n" +
               "\n" +
               "Company=\n" +
               "\n" +
               "Category=Enterprise\n" +
               "\n" +
               "Component=XML-Editor, XSLT-Debugger, Saxon-SA\n" +
               "\n" +
               "Version=14\n" +
               "\n" +
               "Number_of_Licenses=1\n" +
   ```
/**
 * <p><b>Description:</b> TC for opening a file and using a bold operation</p>
 * <p><b>Bug ID:</b> EXM-20417</p>
 *
 * @author radu_coravu
 *
 * @throws Exception
 */

public void testOpenFileAndBoldEXM_20417() throws Exception {
    WSEditor ed = open(new File("D:/projects/eXml/test/authorExtensions/dita/sampleSmall.xml").toURL());

    //Move caret
    moveCaretRelativeTo("Context", 1, false);

    //Insert <b>
    invokeAuthorExtensionActionForID("bold");
    assertEquals("<?xml version="1.0" encoding="utf-8"?>
    <!DOCTYPE task PUBLIC "+//OASIS//DTD DITA Task//EN"
    "task.dtd">
    "
    "<task id="taskId">
    "    <title>Task <b>title</b></title>
    "    <prolog/>
    "    <taskbody>
    "        <context>
    "            <p>Context for the current task</p>
    "        </context>
    "        <steps>
    "            <step>
    "                <cmd>Task step.</cmd>
    "            </step>
    "        </steps>
    "    </taskbody>
    "</task>
    "</task>
    
    "</task.dtd>\n" +
"    "<title>Task <b>title</b></title>
"    "<prolog/>
"    "<taskbody>
"    "    <context>
"    "    <p>Context for the current task</p>
"    "</context>
"    "<steps>
"    "    <step>
"    "    <cmd>Task step.</cmd>
"    "    </step>
"    "</steps>
" +
Debugging a Plugin Using IntelliJ IDEA

To use IntelliJ IDEA to debug problems in the code of a plugin (on page 3222) without having to re-bundle the plugin's Java classes in a JAR (on page 3220) library, follow these steps:

1. Download and install Oxygen XML Editor.
2. Set up the Oxygen SDK following this set of instructions.
3. Create a Java Project (for example, MyPluginProject) from one of the sample plugins (for example, the Workspace Access plugin).

4. In the MyPluginProject folder, create a folder called myPlugin. In this new folder, copy the plugin.xml file from the sample plugin. Modify the added plugin.xml to add a library reference to the directory where IntelliJ IDEA copies the compiled output. To find out where this directory is located, go to File > Project Structure. Then select the Modules category and inspect the value of the Output path text box from the Path tab.

Example: If the output path is C:/Users/myUser/Documents/MyPluginProject/target/classes, then in the plugin.xml, you need to add the following library reference in the runtime element:

```xml
<library name="..../target/classes"/>
```

5. Copy the plugin.dtd from the [OXYGEN_INSTALL_DIR]/plugins folder in the root MyPluginProject folder.
6. In the MyPluginProject dependences (File > Project Structure > Modules > Dependencies), add external JAR references to all the JAR libraries in the [OXYGEN_INSTALL_DIR]/lib folder. Now your MyPluginProject should compile successfully.
7. In IntelliJ IDEA, create a new Java Application configuration for debugging (Run > Edit Configurations... > + > Application). Set the Main class box to ro.sync.exml.Oxygen and add the following code snippet in the VM options input box, making sure that the path to the plugins directory is the correct one:

```
-Dcom.oxygenxml.app.descriptor=ro.sync.exml.EditorFrameDescriptor -Xmx1024m
-XX:MaxPermSize=384m -Dcom.oxygenxml.editor.plugins.dir=D:\projects\MyPluginProject
```

8. Add a breakpoint (on page 2126) in the source of one of your Java classes.
9. Debug the created configuration. When the code reaches your breakpoint (on page 2125), the IntelliJ IDEA debugging view should take over.
Debugging a Plugin Using the Eclipse Workbench

To use the Eclipse workbench to debug problems in the code of a plugin (on page 3222) without having to re-bundle the plugin's Java classes in a JAR (on page 3220) library, follow these steps:

1. Download and install Oxygen XML Editor.
2. Set up the Oxygen SDK following this set of instructions.
3. Create an Eclipse Java Project (for example, MyPluginProject) from one of the sample plugins (for example, the Workspace Access plugin).
4. In the MyPluginProject folder, create a folder called myPlugin. In this new folder, copy the plugin.xml file from the sample plugin. Modify the added plugin.xml to add a library reference to the directory where Eclipse copies the compiled output. To find out where this directory is located, invoke the contextual menu of the project (in the Project view (on page 402)), and go to Build Path > Configure Build Path. Then inspect the value of the Default output folder text box. Example: If the compiled output folder is classes, then in the plugin.xml, you need to add the following library reference:

   `<library name="../classes"/>

5. Copy the plugin.dtd from the \{OXYGEN_INSTALL_DIR\}/plugins folder in the root MyPluginProject folder.
6. In the MyPluginProject build path, add external JAR references to all the JAR libraries in the \{OXYGEN_INSTALL_DIR\}/lib folder. Now your MyPluginProject should compile successfully.
7. In the Eclipse IDE, create a new Java Application configuration for debugging. Set the Main class box to ro.sync.exml.Oxygen. Click the Arguments tab and add the following code snippet in the VM arguments input box, making sure that the path to the plugins directory is the correct one:

   ```
   -Dcom.oxygenxml.app.descriptor=ro.sync.exml.EditorFrameDescriptor -Xmx1024m
   -XX:MaxPermSize=384m -Dcom.oxygenxml.editor.plugins.dir=D:\projects\MyPluginProject
   ```
8. Add a breakpoint (on page 2126) in the source of one of your Java classes.
9. Debug the created configuration. When the code reaches your breakpoint (on page 2125), the Eclipse IDE debugging perspective should take over.

Debugging an Oxygen SDK Extension Using the Eclipse Workbench

To use the Eclipse workbench to debug problems in the code of an extension (on page 3222) without having to bundle its Java classes in a JAR (on page 3220) library, perform the following steps:

1. Download and install Oxygen XML Editor.
2. Create an Eclipse Java Project (for example, MySDKProject) with the corresponding Java sources (for example, a custom implementation of the ro.sync.ecss.extensions.api.StylesFilter interface).
3. In the Project build path, add external JAR references to all the JAR libraries in the \{OXYGEN_INSTALL_DIR\}/lib folder. In the build path Order and Export panel, make sure that the oxygen.jar entry is before all other libraries. Now your Project should compile successfully.
4. Start the standalone version of Oxygen XML Editor from the \[\text{OXYGEN\_INSTALL\_DIR}\] and in the Document Type Association preferences page (on page 138), edit the document type (for example, DITA) to open the Document Type configuration dialog box (on page 140). In the Classpath tab, add a reference to your Project’s classes directory and in the Extensions tab, select your custom StylesFilter extension as a value for the CSS styles filter property. Close the application to save your changes.

5. Create a new Java Application configuration for debugging. The Main Class should be ro.sync.exml.Oxygen. The given VM Arguments should be:

```
-Dcom.oxygenxml.app.descriptor=ro.sync.exml.EditorFrameDescriptor -Xmx1024m
```

6. Add a breakpoint (on page 2126) in one of the source Java classes.

7. Debug the created configuration. When the code reaches your breakpoint (on page 2125), the Eclipse IDE debugging perspective should take over.

Disabling a Plugin

To disable a plugin (on page 3222), use one of the following two methods:

- Open the Preferences dialog box (Options > Preferences) (on page 124), go to Plugins, and deselect the plugin that you want to disable.
- Create an empty file called plugin.disable next to the plugin configuration file (plugin.xml). The plugin will be disabled and will no longer be loaded by the application on startup.

**Note:**

This is useful if you want to temporarily stop work on a plugin and use the application without it.

Oxygen XML Author Component

The Oxygen XML Author Component was designed as a subset of Oxygen XML Editor that can be integrated into another application under the terms of the Oxygen XML Editor SDK agreement to provide functionality for editing and authoring XML documents. The component can be embedded in a third-party standalone Java application to provide WYSIWYG-like XML editing directly in your application.

More information about the setup for the Oxygen XML Author Component can be found on the Oxygen SDK page.

Licensing

The licensing terms and conditions for the Oxygen XML Author Component are defined in the Oxygen SDK License Agreement. To obtain the licensing terms and conditions and other licensing information as well, you can also contact the support team at support@oxygenxml.com. You may also obtain a free of charge evaluation license key for development purposes, subject to registration. Any deployment of an application developed using the Oxygen XML Author Component is also subject to the terms of the SDK agreement.
There are two main categories of Oxygen XML Author Component integrations:

- **Integration for internal use:**
  
  You develop an application that embeds the *Author Component* to be used internally (in your company or by you). You can buy and use previously purchased Oxygen XML Editor floating licenses to enable the runtime usage of the Oxygen XML Author Component as it was integrated into the application.

- **Integration for external use:**
  
  Using the Oxygen XML Author Component, you create an application that you distribute to other users outside your company (with a CMS for example). In this case, you should apply for a Value Added Reseller (VAR) partnership by contacting the *Oxygen Sales Team* (https://www.oxygenxml.com/sales_support.html).

From a technical point of view, the Oxygen XML Author Component provides the Java API to:

- **Inject floating license server details in the Java code:**
  
  The following link provides details about how to configure an HTTP floating license server: https://www.oxygenxml.com/license_server.html#floating_license_servlet.

```java
AuthorComponentFactory.getInstance().init(frameworkZips, 
   optionsZipURL, codeBase, appID,
   //The servlet URL
   "http://www.host.com/servlet",
   //The HTTP credentials user name
   "userName",
   //The HTTP credentials password
   "password");
...
```

**Related Information:**
https://www.oxygenxml.com/sdk_agreement.html

**Installation Requirements**

Running the Oxygen XML Author Component embedded in a third-party Java/Swing application requires:

- Oracle (Sun) Java JRE version 1.8.
- At least 100 MB disk space and 100MB free memory.

**Customization**

For a special type of XML, you can create a custom *framework* (on page 3219) (which also works in a standalone version of Oxygen XML Editor). Oxygen XML Editor already has *frameworks* for editing DocBook,
DITA, TEI, and so on. Their sources are available in the Oxygen SDK. This custom framework is then packed in a zip archive and used to deploy the component.

**Figure 605. Components of a Custom Framework**

![Diagram showing components of a custom framework]

Multiple frameworks can coexist in the same component and can be used at the same time for editing XML documents.
You can add on your custom toolbar all actions available in the standalone Oxygen XML Editor application for editing in the Author mode. You can also add custom actions defined in the framework customized for each XML type.

The Oxygen XML Author Component can also provide the Outline (on page 539), Model (on page 545), Elements (on page 633), and Attributes (on page 627) views, which can be added to your own developed containers.

The main entry point for the Oxygen XML Author Component Java API is the AuthorComponentFactory class.

Related Information:
Creating and Configuring Custom Frameworks (on page 2133)
Oxygen XML Author Component (on page 2451)
AuthorComponentFactory API

Example - Customizing the DITA Framework

If you look inside the bundle-frameworks\oxygen-frameworks folder distributed with the Oxygen XML Author Component sample project, it contains a frameworks folder. Customizations that affect the
framework configuration for the component should first be done in a standalone installation of Oxygen XML Editor.

The Oxygen XML Editor installation also includes a frameworks folder that contains the dita framework located in \{OXYGEN_INSTALL_DIR\}/frameworks\dita. The dita framework contains a bundled DITA-OT distribution that contains the DTDs used for DITA editing. If your DTD specialization is a DITA-OT plugin, it should be installed (on page 2413) in the DITA-OT-DIR\plugins folder.

To make changes to the DITA framework configuration, open the Preferences dialog box (Options > Preferences) (on page 124), go to Document Type Association, and edit or extend the framework. These changes will affect the \{OXYGEN_INSTALL_DIR\}/frameworks\dita\dita.framework configuration file.

After you do this, you can re-pack the Oxygen XML Author Component following the instructions from the README.html file located in the Oxygen XML Author Component sample project. The Author Component sample project and the Oxygen XML Editor standalone installation should be of the same version.

Related Information:
Framework and Author Mode Customization (on page 2133)

Packing a Fixed Set of Options

The Oxygen XML Author Component shares a common internal architecture with the standalone application, although it does not have Preferences dialog boxes. However, the Author Component can be configured to use a fixed set of user options on startup.

The sample project contains a module called bundle-options. The module contains a file called options.xml in the oxygen-options folder. Such an XML file can be obtained by exporting the options to an XML format from an installation of Oxygen XML Editor.

To create an options file in the Oxygen XML Editor:

- Make sure the options that you want to set are not stored at project level (on page 311).
- Set the values you want to impose as defaults in the Preferences pages (on page 124).
- Select Options > Export Global Options.

Adding MathML support in the Oxygen XML Author Component

By default, the Oxygen XML Author Component does not come with the libraries necessary for viewing and editing MathML equations in the Author mode. You can view and edit MathML equations either by adding support for JEuclid (on page 2455) or by adding support for MathFlow (on page 2456).

Adding MathML Support Using JEuclid

By default, the JEuclid library is excluded from the Oxygen SDK artifact dependencies. To enable it, comment the following lines in the pom.xml file:
To edit specialized DITA Composite documents with MathML content, include the entire MathML2 framework directory (for example, the default location is: \texttt{[OXYGEN\_INSTALL\_DIR]/frameworks/mathml2}) in the frameworks (on page 3219) bundled with the component in the bundle-frameworks module. This directory is used to solve references to MathML DTDs.

### Adding MathML Support Using MathFlow

In the \texttt{pom.xml} file, add dependencies to the following additional libraries used by the MathFlow library to parse MathML equations:

1. MFComposer.jar
2. MFExtraSymFonts.jar
3. MFSimpleEditor.jar
4. MFStructureEditor.jar
5. MFStyleEditor.jar

**Note:**

For MathFlow 2.1, all of these JAR (on page 3220) files are packaged into one file called MathFlow.jar.

You can reference these additional libraries from the MathFlow SDK as in the example below:

```xml
<dependency>
    <groupId>com.dessci</groupId>
    <artifactId>MFComposer</artifactId>
    <version>1.0.0</version>
    <scope>system</scope>
    <systemPath>${MathFlowSDKDir}/lib/MFComposer.jar</systemPath>
</dependency>
```

In addition, you must obtain fixed MathFlow license keys for editing and composing MathML equations and register them using these API methods: \texttt{AuthorComponentFactory.setMathFlowFixedLicenseKeyForEditor} and \texttt{AuthorComponentFactory.setMathFlowFixedLicenseKeyForComposer}.

To edit specialized DITA Composite with MathML content, include the entire \texttt{[OXYGEN\_INSTALL\_DIR]/frameworks/mathml2} Mathml2 framework directory in the frameworks (on page 3219) bundled with the component in the bundle-frameworks module. This directory is used to solve references to MathML DTDs.
Adding Support to Insert References from a WebDAV Connection

Predefined actions that insert references, such as the Insert Image action, includes a URL chooser field with a drop-down menu that allows you to select a Browse Data Source Explorer action. This action opens the Data Source Explorer (on page 2015) that allows you to view a WebDAV connection.

To use a WebDAV connection in the Oxygen XML Author Component, follow these steps:

1. Open a standalone Oxygen XML Editor 24.1 and configure a WebDAV connection (on page 2068).
2. Pack the fixed set of options (on page 2455) from the standalone application to use them with the Oxygen XML Author Component project.
3. In the Oxygen XML Author Component, the defined connection still does not work when expanded because the additional JAR libraries used to browse the WebDAV repository are missing. By default, the httpclient dependency of the Oxygen SDK artifact is excluded. You can enable it by commenting the following lines:

```xml
<exclusion>
  <artifactId>httpclient</artifactId>
  <groupId>org.apache.httpcomponents</groupId>
</exclusion>
```

If you want to have multiple WebDAV connection URLs, user names, and passwords (depending on the user who started the component), you can use a more flexible approach by using the following API:

```java
//DBConnectionInfo(String id, String driverName, String url, String user, String passwd, String host, String port)
DBConnectionInfo info = new DBConnectionInfo("WEBDAV", "WebDAV FTP", "http://host/webdav-user-root", "userName", "password", null, null);
AuthorComponentFactory.getInstance().setObjectProperty("database.stored.sessions1", new DBConnectionInfo[] {info});
```

Using Plugins with the Oxygen XML Author Component

To bundle Workspace Access plugins (on page 3222) that are developed for the standalone application with the Oxygen XML Author Component, follow these steps:

- The bundle-plugins module must contain the additional plugin directories in the dropins subdirectory. The content must also contain a plugin.dtd file. Copy the plugin.dtd file from an [OXYGEN_INSTALL_DIR]\plugins folder.
- In the class that instantiates the AuthorComponentFactory (for example the ro.sync.ecss.samples.AuthorComponentSample class), call the methods AuthorComponentFactory.getPluginToolbarCustomizers(), AuthorComponentFactory.getPluginViewCustomizers(), and AuthorComponentFactory.getMenubarCustomizers(), obtain the customizers that have been added
by the plugins and call them to obtain the custom swing components that they contribute. There is a commented-out example for this in the `AuthorComponentSample.reconfigureActionsToolbar()` method for adding the toolbar from the Acrolinx plugin.

**Important:**
As the Oxygen XML Author Component is just a subset of the entire application, there is no guarantee that all the functionality of the plugin will work.

### Frequently Asked Questions

#### Installation and Licensing

1. Are there any client requirements beyond the Java VM?

   Oracle (formerly Sun) Java JRE version 1.8. At least 200 MB disk space and 200MB free memory is necessary for the Oxygen XML Author Component.

2. Does the Oxygen XML Author Component support multiple documents being open simultaneously? What are the licensing ramifications?

   A single `AuthorComponentFactory` instance can create multiple `EditorComponentProvider` editors that can then be added and managed by the developer who customizes the component in a Swing `JTabbedPane`. A single license (floating or user-based) is enough for this.

   If you need to run multiple distinct Java processes using the Oxygen XML Author Component, the current floating license model allows for only two concurrent components from the same computer when using the HTTP floating license server. An additional started component will take an extra license seat.

#### Functionality

1. What graphic formats can be directly rendered in the Oxygen XML Author Component?

   GIF, JPEG, PNG, BMP and SVG.

2. Can links be embedded to retrieve (from the server) and "play" other types of digital assets, such as audio or video files?

   You could add listeners to intercept clicks and open the clicked links. This would require a good knowledge of the Oxygen SDK. The Oxygen XML Author Component can only render static images (no GIF animations).

3. Does the Oxygen XML Author Component provide methods for uploading ancillary files (new graphics, for instance) to the hosting server?

   No.
4. Does the Oxygen XML Author Component provide any type of autosave functionality?

By default no, but you could customize it to save its content periodically to a file on disk.

5. Assuming multiple documents can be edited simultaneously, can content be copied, cut, and pasted from one Oxygen XML Author Component "instance" to another?

Yes.

6. Does the Oxygen XML Author Component support pasting content from external sources (such as a web page or a Microsoft Word document and, if so, to what extent?

If no customizations are available, the content is pasted as simple text. Customizations are provided for the major frameworks (DITA, DocBook, TEI, etc.) that use a conversion XSLT stylesheet to convert HTML content from clipboard to the target XML.

7. Can UTF-8 characters (such as Greeks, mathematical symbols, etc.) be inserted and rendered?

Any UTF-8 character can be inserted and rendered, provided that the font used for editing supports rendering the characters. The font can be changed by developers but not by the users. When using a logical font (by default, Serif for the Oxygen XML Author Component), the JVM will know how to map all characters to glyphs. There is no character map available but you could implement one.

Customization

1. Describe, in general terms, the menus, toolbars, contextual menu options, helper panes, and so on, that are available for the Oxygen XML Author Component out-of-the-box.

You can mount all actions available in the standalone Oxygen XML Editor application on your custom toolbar. This includes custom actions defined in the framework customized for each XML type.

The Oxygen XML Author Component also can provide the Outline (on page 539), Model (on page 545), Elements (on page 633), and Attributes (on page 627) views that can be added to your own panels.

2. Describe, in general terms, the actions, project resources (for example, DTD/Schema for validation purposes, CSS/XSL for styling, etc.) and typical level of effort that would be required to deploy a Oxygen XML Author Component solution for a customer with a proprietary DTD.

The Author mode internal engine uses CSS to render XML.

For a special type of XML, you can create a custom framework (which also works in an Oxygen XML Editor standalone version) that would also contain default schemas and custom actions. A simple framework would probably need 2-3 weeks development time. For a complex framework with many custom actions, it could take a long time. Oxygen XML Editor has built-in frameworks for editing (DocBook, DITA, TEI, etc.) and sources for them are available in the Oxygen SDK.

Multiple frameworks can co-exist in the same Oxygen XML Editor instance and can be used at the same time for editing XML documents.
3. Many customers desire a very simplistic interface for contributors (with little or no XML expertise) but a more robust XML editing environment for editors (or other users with more advanced XML expertise). How well does the Oxygen XML Author Component support varying degrees of user interface complexity and capability?

   ◦ Showing/hiding menus, toolbars, helpers, etc.

   You assemble all the UI parts from the Oxygen XML Author Component. For example, you could provide two implementations: one for advanced users and one for content authors.

   ◦ Forcing behaviors (for example, ensuring change tracking (on page 3224) is on and preventing it from being shut down).

   You could avoid placing the change tracking toolbar actions in the UI. You could also use the API to turn change tracking ON when the content has been loaded.

   ◦ Preventing access to “privileged” editor processes (for example, accept/reject changes).

   You can remove the change tracking actions completely in a custom implementation, including the ones from the contextual menu.

   ◦ Presenting and/or describing XML constructs (for example, tags) in "plain-English".

   Using our API, you can customize what the Outline view or Breadcrumb displays for each XML tag. You can also customize the in-place content completion list.

   ◦ Presenting a small subset of the overall XML tag set (rather than the full tag set) for use by contributors (for example, allowing an author to only insert Heading, Para, and inline emphasis).

   The API allows for a content completion filter that also affects the Elements view.

4. Does the Oxygen XML Author Component API provide access to the XML document, for manipulation purposes, using common XML syntax (such as DOM, XPath, etc.)?

   Yes, using the Oxygen XML Author Component API.

5. Can custom dialog boxes be developed and launched to collect information in a "form" (with scripting behind to tag the collection information and embed it in the XML document)?

   Yes.

6. Can project resources and customizations be readily shared between the desktop and component versions of your Oxygen XML Author Component product line?

   A framework developed for the standalone version of the Oxygen XML Editor application can then be bundled with the Oxygen XML Author Component. For example, you could use the same framework that you use in the Oxygen XML Editor standalone distribution.

   A custom editing solution can deploy one or more frameworks that can be used at the same time.
Print Document Within the Oxygen XML Author Component

Question
Can a document be printed within the Oxygen XML Author Component?

Answer
You can use the following API method to either print the document content to the printer or to show the Print Preview dialog box, depending on the `preview` parameter value:

```java
AuthorComponentProvider.print(boolean preview)
```

Here is the online Javadoc for this method: https://www.oxygenxml.com/InstData/Editor/SDK/javadoc/ro/sync/ecss/extensions/api/component/AuthorComponentProvider.html#print(boolean)

Oxygen XML Web Author Component

The Oxygen SDK provides the ability to integrate the Oxygen XML Web Author into your existing content ecosystem and it allows anyone in your organization to access your content from anywhere they have an internet connection. Oxygen XML Web Author is highly versatile and can be customized to work with any XML vocabulary, most file repository systems, and virtually any type of workflow.

Web Author Component Integration

For information about integrating Oxygen XML Web Author into your environment, see Web Author Integration.

Web Author Customization

For detailed information about customizing Web Author, see the Oxygen XML Web Author Customization Guide.

Using Web Author

For information about using the Web Author product, see the Oxygen XML Web Author User Manual.

Web Author vs. Web Author Component

The purpose of this topic is to help you choose which distribution of Oxygen XML Web Author is appropriate for your particular use-case.

Oxygen XML Web Author has two distributions:

1. **Oxygen XML Web Author** (product) - This version can be downloaded from the Oxygen website and is licensed according to the Web Author License Agreement.

2. **Web Author Component** - Provided as a Maven artifact (according to the SDK Agreement) that is used in the Web Author Component integration project. This project can be used as a starting point for your integration.
Note:
The formal definition for Web Author Component inside the SDK agreement is:

“Web Author Component” is a subset of Software composed of a server component operating as a service application and a client component deployed to a web browser, such as an HTML5 based application, where the client component is not installed on the client machine but is in use by the client machine while the browser is connected to the server component.”

For End-Users

For end-users, the recommended distribution is Oxygen XML Web Author (product).

For Plugin and Framework Developers

If you develop frameworks or plugins that provides enhanced functionality for Oxygen XML Web Author, it is recommended to distribute the plugin or framework separately from Web Author. They will work with both types of distributions and the end-users can choose:

- To install your plugin or framework in Web Author.
- To use a solution based on Web Author that has the plugin or framework installed by default.
- To use a solution based on Web Author and install the plugin or framework themselves.

Example of such plugins/frameworks:

- Integration with a terminology database.
- Adding support to edit embedded SVG snippets.
- Providing support for a specific XML language.

For Integration Developers

If you want to integrate Oxygen XML Web Author into another application (for example, a CMS), that you distribute to your end-users, there are several aspects to consider when choosing between the two distributions:

Functionality

There are some features that are available in the installable Oxygen XML Web Author product requires an additional plugin to be available in the Web Author Component. For example, the File Comparison Tool is available by default in the Oxygen XML Web Author product, but requires an additional plugin to be installed for it to be available in the Web Author Component version.

Legal

If you want to use the Web Author Component, you must have your own application that is not called Oxygen XML Web Author, but will contain Web Author as a software component provided by Syncro Soft. Hence,
Syncro Soft does not have a direct legal link with your users, you will handle the licensing of your application to your end-users.

For the official details, you can consult the license agreements for both distributions:

2. **Web Author Component**: [https://www.oxygenxml.com/sdk_agreement.html](https://www.oxygenxml.com/sdk_agreement.html)

### Financial

The prices for **Oxygen XML Web Author** (product) are listed on the **Oxygen** website: [https://www.oxygenxml.com/xml_web_author/buy_oxygen_xml_web_author.html](https://www.oxygenxml.com/xml_web_author/buy_oxygen_xml_web_author.html).

If you choose to use the **Web Author** product to distribute as a plugin or framework, then you act as a channel reseller, basically reselling Oxygen XML Web Author to your end-users and you must pay Syncro Soft the corresponding cost of the license.

The **Web Author Component** has more flexible pricing alternatives for resellers that are negotiated for each contract and include:

- Subscription packages similar to the ones used for the Web Author product, but you may get a discount depending on the partnership level (Gold, Silver, Bronze).
- Solution OEM license.

If you choose to integrate the **Web Author Component** into your application, the cost is based on royalties, rather than a cost for the license. That means you do not need to pay for the ability to include the component in your application, but when you distribute your application you must pay a royalty to Syncro Soft.

### Customization

It is strongly recommended to implement any customization as a combination of plugins, frameworks, and options. If you are using **Web Author Component**, you can also alter the Web Author files, but this is discouraged since those files are not considered API and may change in a future version.

### Deployment

If you want to use **Oxygen XML Web Author** (product), you can install it using the installation kits, then configure the plugins, frameworks, and options using a browser-based UI. Alternatively, you can configure Oxygen XML Web Author to load the configuration from a custom directory (**Oxygen Data Directory**) along with desired plugins, frameworks, and options.

If you want to use **Web Author Component**, you can use the **Web Author Component integration project** to build a custom WAR file with plugins, frameworks, and options bundled. This WAR file can be deployed in any Servlet container (for example, Tomcat).
Distribution

If you want to use **Oxygen XML Web Author** (product), your users must download Oxygen XML Web Author from the [Oxygen website](https://www.oxygenxml.com), while you will distribute your plugins/frameworks to your users. You should provide instructions on how to deploy and use Oxygen XML Web Author. Alternatively, you can install and configure it for them.

If you want to use **Web Author Component**, you will distribute the customized WAR file.

Developer Quick Start Guide

Oxygen XML Editor allows you to develop add-ons to customize the editing experience. Such customizations can be achieved through a plugin or a framework configuration. This section is meant to provide guidance to developers who are getting started with these types of customizations and to offer links to various resources to help with customizations.

- A **plugin** can be used to customize the behavior of the entire application no matter what XML document is currently being edited. Once created, a plugin can be **deployed and installed as an add-on** (on page 2445). For more information, see the [The Oxygen SDK (Part 1: Plugins)](https://www.oxygenxml.com/doc/ug-editor/topics/plugins.html) blog post.

- A **framework** configuration provides validation, content completion, and editing support for a specific XML vocabulary. See: [https://blog.oxygenxml.com/topics/oxygenFrameworks.html](https://blog.oxygenxml.com/topics/oxygenFrameworks.html). Once created, a framework can be deployed and installed as an add-on. See: [https://www.oxygenxml.com/doc/ug-editor/topics/packing-and-deploying-addons.html](https://www.oxygenxml.com/doc/ug-editor/topics/packing-and-deploying-addons.html).

From a legal point of view, you can freely develop and share such extensions as long as they are only used from inside Oxygen XML Editor. For details, see: [https://www.oxygenxml.com/sdk_agreement.html](https://www.oxygenxml.com/sdk_agreement.html).

Plugins

A **plugin** can be used to customize the behavior of the entire application no matter what XML document is currently being edited. Since Oxygen XML Editor is a Java-based application, most of the allowed plugin types are Java-based but some JavaScript-based plugin types are also supported.

There are lots of **plugin types** (on page 2414) but the **Workspace Access plugin type** (on page 2415) is the most versatile of them. This type of plugin allows you to contribute actions to the main menu and toolbars, create custom views, interact with the application workspace, make modifications to open documents, and add listeners for various events. A **Workspace Access** plugin can also **contribute frameworks** (on page 2422).

The Maven-based [Oxygen XML SDK](https://www.oxygenxml.com/doc/ug-editor/topics/plugins.html) comes with sample plugins and it provides the ability to compile Java extensions for your plugins and frameworks. Also, as a quick start for a **Workspace Access** plugin, you can use this project: [https://github.com/oxygenxml/sample-plugin-workspace-access](https://github.com/oxygenxml/sample-plugin-workspace-access).

The **Workspace Access** plugin API can also be used with a **JavaScript-based plugin** (on page 2418). Small plugin samples can be found here: [https://github.com/oxygenxml/wsaccess-javascript-sample-plugins](https://github.com/oxygenxml/wsaccess-javascript-sample-plugins).
You can create automated tests (on page 2450) for your plugins and debug them using the Eclipse IDE (on page 2450).

A plugin can either be installed manually or packed as an add-on and installed using Help->Install new add-ons (on page 2413).

Sample Plugins

The Oxygen GitHub site contains lots of open-source plugins (https://github.com/topics/oxygen-standalone-plugin). Most of these plugins are of the Workspace Access type.

There is also a sample project which contains various JavaScript-based plugins: https://github.com/oxygenxml/wsaccess-javascript-sample-plugins.

You can also find a variety of other publicly-hosted Oxygen plugins in the Public Hosted Oxygen Plugin and Framework Projects blog post.

Workspace Access Plugin

This type of plugin extension allows you to contribute actions to the main menu and toolbars, create custom views, interact with the application workspace, make modifications to open documents, and add listeners for various events. It is the most useful and most commonly used plugin extension.

A Workspace Access plugin (on page 2415) can also provide frameworks, allowing you to have a single add-on that provides both workspace-level extensions (independent of any given framework) and document type-specific frameworks. If the frameworks involve Java extensions (for example, custom dialog boxes or link text resolvers), they use the Java code for the Workspace Access plugin.

You can include frameworks with a Workspace Access plugin by declaring an "additional frameworks" extension in the plugin.xml file (on page 2422).

Java or JavaScript?

Oxygen XML Editor is a Java-based application and all of its APIs are Java-based. The entire user interface (buttons, views, dialog boxes) is built on top of the Java Swing architecture. The entire Javadoc API documentation is available here: https://www.oxygenxml.com/InstData/Editor/SDK/javadoc/.

A Workspace Access plugin can be implemented either in Java or in JavaScript. Sample Java-based Workspace Access plugins can be found on the Oxygen XML GitHub page.

Sample JavaScript-based implementations can be found in this sample project: https://github.com/oxygenxml/wsaccess-javascript-sample-plugins. The Rhino library is used to convert the JavaScript method calls to Java API calls: https://github.com/mozilla/rhino.

Related Information:
The **Workspace Access plugin extension** is called when the application starts and when it closes.

The **StandalonePluginWorkspace** API can be used in numerous ways:

- Customize the toolbars, contextual menu, and main menus. See: [Adding Toolbar and Menu Actions](on page 2466).


- Access the API for the **Project** view. See: https://www.oxygenxml.com/InstData/Editor/SDK/javadoc/ro/sync/exml/workspace/api/standalone/StandalonePluginWorkspace.html#getProjectManager--.

- Access utility methods to interact with the end-user (for example, show warning and error dialog boxes, update the results view, or change the status bar). See: https://www.oxygenxml.com/InstData/Editor/SDK/javadoc/ro/sync/exml/workspace/api/WorkspaceUtilities.html.

- Add a listener to be notified when a new XML document is opened, selected, or closed either in the main editing area or in the **DITA Maps Manager** view. See: https://www.oxygenxml.com/InstData/Editor/SDK/javadoc/ro/sync/exml/workspace/api/PluginWorkspace.html#addEditorChangeListener-ro.sync.exml.workspace.api.listeners.WSEditorChangeListener-int-.

- Provide access to opened XML documents via the **WSEditor** interface. Each opened XML document can be manipulated using the **WSEditor** interface. You can obtain its content, set new content to it, or save its content. You can also validate the editor contents or disable editing inside it. Depending on the current editing mode (**Text** or **Author**), you can gain access to the current editing page and send it either to the **Author editing page** or **Text editing page**. Both APIs allow you to make changes to the current document content.

### Adding Toolbar and Menu Actions

A **Workspace Access plugin extension** (on page 2415) can contribute custom actions to the contextual menu, main menus, or to the general toolbars.

- **Contributing a new toolbar action:**

  This sample **Workspace Access plugin** contributes a new toolbar called **SampleWorkspaceAccessToolbarID**. The java code of the sample plugin will use the **toolbar components customizer API**.

- **Contributing an action on the main menu:**
As exemplified in the sample plugin, the `addMenuBarCustomizer` API can be used either to add a new menu or to customize the existing main menu.

- **Contributing a contextual menu action:**

  The same sample plugin uses the `addMenusAndToolbarsContributorCustomizer` API to contribute a contextual menu customizer. Such a customizer can be contributed either for the Text or Author editing modes.

Once an action is added, you can define a new shortcut key for it using the `ActionProvider` API. The action can use the `WSEditor API` to make changes to an open XML document.

The same customizer API can be used to remove actions from the main menu, toolbars, framework-specific menus, and contextual menus.

**Adding a New Side-View**

A Workspace Access plugin (on page 2415) type can contribute a new side view to Oxygen XML Editor. For example, the following plugin.xml descriptor file defines a new view ID called `SampleWorkspaceAccessToolbarID`: https://github.com/oxygenxml/sample-plugin-workspace-access/blob/master/plugin.xml.

Once the new view ID is declared, the Java code of the plugin can add content to the view using the `pluginWorkspaceAccess.addViewComponentCustomizer` API.

**Customizing the Project View**

The API method `StandalonePluginWorkspace.getProjectManager()` allows access to various project-related functionalities:

- Add a new contextual menu action in the Project view.
- Access the set of resources currently selected in the Project view.
- Customize the icons that appear in the Project view.

A sample JavaScript-based plugin that uses this API to add a new contextual menu to the Project view can be found here: https://github.com/oxygenxml/wsaccess-javascript-sample-plugins/tree/master/OpenInTerminalProjectContextualAction.

**Customizing the DITA Maps Manager View**

You can add a listener to be notified when a new DITA map is opened, selected, or closed in the DITA Maps Manager view. Once the `editorOpened()` callback is received, you can obtain the opened WSEditor API, then send its current page to the `WSDitaMapEditorPage`.
The API method `WSDitaMapEditorPage` allows you to interact with the DITA map that is open in the DITA maps Manager view:

- Add a customizer for the icons and text presented in the tree.
- Enable or disable editing on the tree.
- Set a popup menu customizer.
- Get the selected nodes.
- Get access to the `AuthorDocumentController` API to make changes to the content.

Sample plugins:

- JavaScript-based plugin that customizes the icons and text presented for a DITA map that is open in the DITA Maps Manager view: https://github.com/oxygenxml/wsaccess-javascript-sample-plugins/tree/master/dmmCustomizeTopicTitlesAndIcons.
- JavaScript-based plugin that adds a new contextual menu action for a DITA map that is open in the DITA Maps Manager view: https://github.com/oxygenxml/wsaccess-javascript-sample-plugins/tree/master/contributePopupActionDMM.

### Persistent Storage

Your plugin may need to save plugin-specific information persistently between two sessions. The `PluginWorkspace.getOptionsStorage()` method allows you to save and retrieve (key, value) pairs persistently between sessions (between closing and restarting Oxygen XML Editor). You can also add listeners to be notified when the values for a certain key are changed.

### Contributing a Custom Preferences Page

There is a specific plugin extension type that can be used to contribute a custom preferences page (on page 2429) to the Preferences dialog box in Oxygen XML Editor. An example of how such a page is implemented can be found in this sample plugin: https://github.com/oxygenxml/oxygen-dita-prolog-updater-addon/blob/master/src/main/java/com/oxygenxml/prolog/updater/plugin/PrologOptionPageExtension.java.

### Imposing a Fixed Set of Global Preferences

You may want to impose a fixed set of global options to be used by all end-users who install the plugin. The `GlobalOptionsStorage` API provides the ability to set the following:

- Set a certain global option to a certain value: (https://www.oxygenxml.com/InstData/Editor/SDK/javadoc/ro/sync/exml/workspace/api/options/GlobalOptionsStorage.html#setGlobalObjectProperty-java.lang.String-java.lang.Object-) The `APIAccessibleOptionTags` interface contains a list with all keys that can be set to a custom value.

Other ways to share a common set of options with others are listed here: https://blog.oxygenxml.com/topics/sharingSettings.html.

Interaction with the End-User

If you need your plugin to frequently interact with the end user, some possibilities include:

• Your plugin can create Java Swing-based components (dialog boxes, frames) that are displayed when custom toolbar or menu actions (on page 2466) added by the plugin are called. You can also extend the Oxygen-specific API base class OKCancelDialog to create a dialog box that already includes OK and Cancel buttons. This specific base also automatically resizes its internal components depending on the currently used fonts or DPI settings and also properly positions the OK and Cancel buttons depending on the operating system (on macOS, the OK button is on the right part of the dialog box, while on Windows and Linux, it is placed on the left part of the dialog box). There is an entire API package that contains base implementations of Swing components and such implementations are recommended to be used for the plugin-contributed components to look like the ones contributed by Oxygen XML Editor.

• Your plugin can add a specific side view (on page 2467).

• The WorkspaceUtilities API allows you to:
  ◦ Show file and folder choosers.
  ◦ Show confirmation dialog boxes.
  ◦ Show information, warning, or error dialog boxes.
  ◦ Show a custom status message in the application.

• The ResultsManager API allows you to add results in the Results view. These results can point to a specific document at a specific line/column location.

• The title of the main application frame can be modified using this API: https://www.oxygenxml.com/InstData/Editor/SDK/javadoc/ro/sync/exml/workspace/api/Workspace.html#setParentFrameTitle-java.lang.String-.
Contributing Translations for New Labels and UI Text

You may want your plugin's interaction with the end-user (dialog boxes, pop-up messages, etc.) to be properly translated in all user interface languages (on page 338) supported by Oxygen XML Editor. The API method `StandalonePluginWorkspace.getResourceBundle()` will allow you to pass message keys that will be resolved by the application to specific language-dependent values by looking at a file called `translation.xml`, which needs to be placed in a folder called `i18n` in the plugin installation folder. The structure of the `translation.xml` file needs to look like this: https://www.oxygenxml.com/doc/ug-editor/topics/contribute-new-languages-extension.html.

Customizing the Application Layout

There are two main ways to customize the layout of the application:

- Remove some of the toolbars, actions, menus, or views that Oxygen XML Editor shows by default when the application starts. A sample plugin that filters the user interface based on an XML configuration file is available here: https://github.com/oxygenxml/oxygen-components-filter-plugin.

- Export the layout of the current views and toolbars in the application using the **Window->Export Layout** action, then use the **WorkspaceAccess plugin API** to impose a fixed value for a global option key:

```java
File layoutFile = new File(baseDir, "application.layout");
if (layoutFile.exists()) {
    PerspectivesLayoutInfo info = new PerspectivesLayoutInfo(true, false, ",",
    layoutFile.getAbsolutePath());
    pluginWorkspaceAccess.setGlobalObjectProperty("perspectives.layout.info", info);
    ...
}
```

Adding new User Interface Translations

There is a particular plugin extension to contribute a new language to Oxygen XML Editor: https://www.oxygenxml.com/doc/ug-editor/topics/contribute-new-languages-extension.html.

Frameworks

A **framework** configuration provides validation, content completion, and visual editing functionality for a certain XML vocabulary. Usually, a framework customization provides a schema used to validate and edit certain type of XML documents, a CSS used to edit the XML documents in the **Author** visual editing mode and various custom actions or behaviors used to enhance the editing experience. For more information about framework customization, see: https://blog.oxygenxml.com/topics/oxygenFrameworks.html.

Oxygen XML Editor comes with a lot of framework configuration folders ([OXYGEN_INSTALL_DIR]/frameworks) to support editing XML documents of various types (such as DocBook, DITA, XHTML, or TEI). All of these existing framework configurations can be further customized in the **Preferences->Document Type Associations** page. These framework configurations can be used as examples for building your own.
customization for a certain XML vocabulary or they can be extended if you want to share a modified version of a framework with others.

The Document Type Association configuration dialog box allows you to configure all the framework-specific settings.

You can also find various open-source frameworks for Oxygen XML Editor online: https://blog.oxygenxml.com/topics/Oxygen%20plugins%20and%20frameworks.html.

Once you have set up a framework configuration folder, it can be packaged as an add-on and shared with others or it can be packaged in workspace access plugins using the “additional framework” extension point in the plugin.xml file (on page 2422).

Customizing an Existing Framework

An existing framework that has full built-in support (for example the DITA framework) can be extended and customized. Afterward, this customization can be shared with others. You can use such a framework customization extension to:

- Provide custom new file templates.
- Provide a custom CSS layer to render the framework in the Author visual editing mode.
- Provide custom Schematron-based validation for the XML documents.
- Provide custom Author mode actions on the toolbar, in the contextual menu, and in the main framework-specific menu.
Customizing the Content Completion Proposals

When editing an XML document either in the Text or Author editing modes, you can invoke the Content Completion Assistant (Ctrl+Space in Text mode or ENTER in Author mode) to see the allowed XML elements or attributes that can be inserted at the current location. The Elements view also presents the elements that can be inserted in the document at a certain location, while the Attributes view presents a list of allowed attributes and their values.

The content completion proposals can be customized in various ways:

- Each framework can contain a special content completion configuration file. Such a file can:
  - Filter out element proposals for a parent element.
  - Configure a set of required attributes to be inserted along with a certain element.
  - Add new attribute value proposals and for each proposal, add an annotation that will appear in the Attributes view for each value.
  - Call an external XSLT script to compute value proposals for a certain attribute.
  - Customize how the element names are presented in the Outline view, Elements view, and Content Completion Assistant.

- You can alter the schema that is associated with the XML document. For example, in the case of the DITA vocabulary, you can create a DTD specialization plugin and integrate it into Oxygen XML Editor.

- You can use the SchemaManagerFilter API to filter the set of proposed elements and attribute values using Java code.

Adding Custom New File Templates

The New Document Wizard (on page 368) (File->New or the New button on the toolbar) presents custom file templates gathered from all frameworks installed in Oxygen XML Editor. A custom framework can have one or more special folders that contain custom new file templates.

Adding Custom Validation Stages

You can distribute a framework with a series of already configured validation scenarios. Also, this provides enhanced validation support that allows you to use multiple grammars to check the document. For example, you can use Schematron rules to impose guidelines that are otherwise impossible to enforce using conventional validation. See: Configuring Validation Scenarios for a Framework.

Adding Custom Transformation Scenarios

When distributing a framework to users, it is a good idea to have the transformation scenarios already configured. This helps the content authors publish their work in various formats. By being contained in the
framework configuration, the scenarios can be distributed along with the actions, menus, toolbars, and catalogs. See: Configuring Transformation Scenarios for a Framework.

Customizing the Author Visual Editing Mode

The **Author** visual editing mode is based on CSS. Besides supporting most of the CSS 3 specification, Oxygen XML Editor adds some custom CSS selectors, properties, and functions. Customization possibilities include:

- **Use CSS selectors** to match XML comments, processing instructions, entities, and CDATA sections.

- Change the tags display mode and tag color for certain elements, mark certain XML elements as not editable, and other customizations using additional CSS properties.

- Use custom CSS functions. For example, the `oxy_xpath` function allows you to run an XPath search over the document and use that value as static text.

- In custom pseudo-classes, you can match values that can be changed via a custom action.

- There are specific @media types that can be used to mark certain CSS sections for a certain distribution.

- Fonts can be dynamically loaded and used for rendering.

Adding Toolbar and Menu Actions

The framework customization (on page 2470) can define actions that appear on a framework-specific toolbar when editing content in the **Author** visual editing mode.

You can use the **Author Action** dialog box (on page 148) to configure the name, description, icons, menu shortcuts, and various XPath-enabled activation operations (on page 152).

You can use a variety of pre-defined operations in each activation mode to achieve various things:

- Insert an XML fragment in the document either at the current position or at a specified offset.

- Set an attribute with a certain value on a certain element.

- Invoke an XSLT script using the XSLTOperation to produce an XML fragment to be inserted in the document.

- Invoke a JavaScript function that can use the **Author** mode APIs to modify the document. Some samples of such operations can be found here: https://github.com/oxygenxml/javascript-sample-operations.

- Set a CSS pseudo-class on a certain element. The pseudo-class can be matched from the CSS to style various elements differently.

You can also create custom **Author** mode operations by extending the **AuthorOperation** Java API.
Once a custom action has been created, it can be added to the main menu, toolbar, or contextual menu.

**Embedding Form Controls**

By using custom CSS functions, you can embed form controls (checkboxes, combo boxes, text fields, pop-up boxes, buttons, etc.) in the **Author** visual editing mode to edit attribute values or text content for certain elements.

All the supported form controls can be found in the **Form Controls section**.

Sample XML and CSS documents that use form controls can be found in the `{OXYGEN_INSTALL_DIR}/samples/form-controls` folder.

**Adding Inline Actions**

Using the `oxy_button` and `oxy_buttonGroup` form controls, you can add inline actions in the **Author** visual editing mode. To see an example, you can open a Lightweight DITA topic from the folder `{OXYGEN_INSTALL_DIR}/samples/dita/lw-dita/`.

**Debugging CSS-related Problems**

The **CSS Inspector** view can be used to find out how various CSS styles are applied. For more information, see **Debugging CSS Stylesheets**.

**Customizing Links**

If you need to have working links between your XML document instances in the **Author** visual editing mode, consider the following possibilities:

- You can use the `-oxy-link` CSS property to specify a link target on a static icon placed before the element.

- You can use the `oxy_link-text()` CSS function to take control over the text presented inside a link using a specific Java extension.

- You can use a custom `ExtensionsBundle` implementation to be notified on a specific callback if the reference needs further processing.

- You can implement a custom link target element finder if the links are not referenced directly to elements that have an ID attribute. The link target element finder will be used to locate the target when the end-user clicks the link.

**Customizing the Smart Paste Mapping**

The **Smart Paste feature** in Oxygen XML Editor preserves certain style and structure information when copying content and pasting it into XML documents. It is also possible to **customize the mapping for the Smart Paste mechanism**.
If you want full control over this behavior, there are also Java extensions that can be customized.

**Difference Between a Framework (Document Type) and a Plugin Extension**

**Question**

What is the difference between a Framework (on page 3219) and a Plugin (on page 3222) Extension?

**Answer**

There are two possible ways to customize the application:

1. Implement a plugin.

   A plugin serves a general purpose and influences any type of XML file that you open in Oxygen XML Editor.

   For the Oxygen XML Editor Plugins API, Javadoc, samples, and documentation, go to https://www.oxygenxml.com/oxygen_sdk.html#Developer_Plugins

2. Create or modify the document type (on page 2133) that is associated to your specific XML vocabulary.

   This document type can be used, for instance, to provide custom actions for your type of XML files and to mount them on the toolbar, menus, and contextual menus.

   For example, if the end-users are editing DITA documents, all the toolbar actions that are specific for DITA are provided by the DITA framework. If you look in the Document Type Association preferences page (on page 138) there is a DITA document type. If you edit that document type you will see that it has an Author tab in the Document Type Configuration dialog box (on page 140). The subtabs in this tab can be used to define custom DITA actions and add them to the toolbars, main menus, or contextual menus.

   For information about developing your own document types (frameworks), see the Creating and Configuring Custom Frameworks (on page 2133) section.

   If you look on disk in the \[OXYGEN_INSTALL_DIR\]rameworks\dita folder, there is a file called dita.framework. That file gets updated when you edit a document type from the Document Type Association preferences page (on page 138). Then you can share that updated file with all users.

   The same folder contains some JAR (on page 3220) libraries. These libraries contain custom Java operations that are called when the user presses certain toolbar actions.

   The Oxygen SDK contains the Java sources from all the DITA Java customizations:

   https://www.oxygenxml.com/oxygen_sdk.html#XML_Editor_Authoring_SDK
Important:
It is possible for a plugin to share the same classes with a framework. For further details, go to How to Share the Classloader Between a Framework and a Plugin (on page 2444).

Related Information:
Adding a Custom Operation to an Existing Framework (on page 2177)

SDK Common Use Cases
This section contains details for specific use cases regarding customizations using the Oxygen SDK, Author Component (on page 2451), or plugins (on page 2411).

For additional questions, contact the Oxygen support team. The preferred approach is via email because these types of questions must be analyzed thoroughly. The Oxygen support team also provides code snippets, if applicable.

To stay up-to-date with the latest changes, discuss issues, and ask for solutions from other developers working with the Oxygen SDK, register on the Oxygen-SDK mailing list.

Add Custom Actions to the Contextual Menu

Use Case
You want to add your own custom actions to the contextual menu using an API.

Solution
The WSAuthorEditorPageBase.addPopUpMenuCustomizer and WSTextEditorPage.addPopUpMenuCustomizer API methods allow you to customize the contextual menu shown either in the Author or Text modes. The API is available both in the standalone application and in the Eclipse plugin.

To add actions to the Author page from your Eclipse plugin extension:

1. Create a pop-up menu customizer implementation:

```java
import org.eclipse.jface.action.ContributionManager;
import org.eclipse.ui.PlatformUI;
import org.eclipse.ui.menus.IMenuService;
import ro.sync.ecss.extensions.api.AuthorAccess;
import ro.sync.ecss.extensions.api.structure.AuthorPopupMenuCustomizer;
/**
 * This class is used to create the possibility to attach certain
 * menuContributions to the {link ContributionManager}, which is used for the
 * popup menu in the Author Page of the Oxygen Editor.<br />
 */
```
* You just need to use the org.eclipse.ui.menus extension and add a
* menuContribution with the locationURI: <b>menu:oxygen.authorpage</b/>
* */

```java
public class OxygenAuthorPagePopupMenuCustomizer implements AuthorPopupMenuCustomizer {

@Override
  public void customizePopUpMenu(Object menuManagerObj, AuthorAccess authoraccess) {
    if (menuManagerObj instanceof ContributionManager) {
      ContributionManager contributionManager = (ContributionManager) menuManagerObj;
      IMenuService menuService = (IMenuService) PlatformUI.getWorkbench()
        .getActiveWorkbenchWindow().getService(IMenuService.class);

      menuService.populateContributionManager(contributionManager,
        "menu:oxygen.authorpage";
      contributionManager.update(true);
    }
  }
}

2. Add a workbench listener and add the pop-up customizer when an editor is open in the Author page:

```java
Workbench.getInstance().getActiveWorkbenchWindow().getPartService()
  .addPartListener(
    new IPartListener() {
      @Override
        public void partOpened(IWorkbenchPart part) {
          if (part instanceof ro.sync.exml.workspace.api.editor.WSEditor) {
            WSEditorPage currentPage = ((WSEditor)part).getCurrentPage();

            if (currentPage instanceof WSAuthorEditorPage) {
              ((WSAuthorEditorPage)currentPage).addPopUpMenuCustomizer
                (new OxygenAuthorPagePopupMenuCustomizer());
            }
          }
        }
      }
    });
```

3. Implement the extension point in your plugin.xml file:

```xml
<extension
  point="org.eclipse.ui.menus">
  <menuContribution
    allPopups="false"
```
Add Custom Callouts

Use Case

You want to highlight validation errors, instead of underlining them (for example, changing the text background color to red or yellow) and display a message directly at the error position that describes the problem.

Solution

The Plugins API allows you to set a ValidationProblemsFilter that gets notified when automatic validation errors are available. Then you can map each of the problems to an offset range in the Author mode using the API WSTextBasedEditorPage.getStartEndOffsets(DocumentPositionedInfo). For each of those offsets, you can add either persistent or non-persistent highlights. If you add persistent highlights, you can also customize callouts to appear for each of them. The downside is that they need to be removed before the document gets saved. The result would look something like this:

Figure 607. Custom Callouts with Persistent Highlights

Here is a working example:

```java
/**
 * Plugin extension - workspace access extension.
 */
```
```java
public class CustomWorkspaceAccessPluginExtension implements WorkspaceAccessPluginExtension {

    /**
     * @see ro.sync.exml.plugin.workspace.WorkspaceAccessPluginExtension
     * applicationStarted(
     * ro.sync.exml.workspace.api.standalone.StandalonePluginWorkspace)
     */
    public void applicationStarted(
        final StandalonePluginWorkspace pluginWorkspaceAccess) {
        pluginWorkspaceAccess.addEditorChangeListener
        (new WSEditorChangeListener() {
            /**
             * @see WSEditorChangeListener#editorOpened(java.net.URL)
             */
            @Override
            public void editorOpened(URL editorLocation) {
                final WSEditor currentEditor = pluginWorkspaceAccess.getEditorAccess
                (editorLocation, StandalonePluginWorkspace.MAIN_EDITING_AREA);
                WSEditorPage currentPage = currentEditor.getCurrentPage();
                if(currentPage instanceof WSAuthorEditorPage) {
                    final WSAuthorEditorPage currentAuthorPage =
                    (currentAuthorPage.getPersistentHighlighter().setHighlightRenderer
                    (new PersistentHighlightRenderer() {
                        @Override
                        public String getTooltip(AuthorPersistentHighlight highlight) {
                            return highlight.getClonedProperties().get("message");
                        }
                        @Override
                        public HighlightPainter getHighlightPainter
                        (AuthorPersistentHighlight highlight) {
                            //Depending on severity could have different color.
                            ColorHighlightPainter painter = new ColorHighlightPainter
                            (Color.COLOR_RED, -1, -1);
                            painter.setBgColor(Color.COLOR_RED);
                            return painter;
                        }
                    });
                    currentAuthorPage.getReviewController().getAuthorCalloutsController().setCalloutsRenderingInformationProvider
                    (new CalloutsRenderingInformationProvider() {
```
@Override
public boolean shouldRenderAsCallout(AuthorPersistentHighlight highlight) {
    // All custom highlights are ours
    return true;
}

@Override
public AuthorCalloutRenderingInformation getCalloutRenderingInformation(
    final AuthorPersistentHighlight highlight) {
    return new AuthorCalloutRenderingInformation() {
        @Override
        public long getTimestamp() {
            // Not interesting
            return -1;
        }
        @Override
        public String getContentFromTarget(int limit) {
            return "";
        }
        @Override
        public String getComment(int limit) {
            return highlight.getClonedProperties().get("message");
        }
        @Override
        public Color getColor() {
            return Color.COLOR_RED;
        }
        @Override
        public String getCalloutType() {
            return "Problem";
        }
        @Override
        public String getAuthor() {
            return "";
        }
        @Override
        public Map<String, String> getAdditionalData() {
            return null;
        }
    };
}
currentEditor.addValidationProblemsFilter(new ValidationProblemsFilter() {
List<int[]> lastStartEndOffsets = new ArrayList<int[]>();
/**
 * @see ro.sync.exml.workspace.api.editor.validation.ValidationProblemsFilter
 #filterValidationProblems
 (ro.sync.exml.workspace.api.editor.validation.ValidationProblems)
 */

@override
public void filterValidationProblems(ValidationProblems validationProblems) {
    List<int[]> startEndOffsets = new ArrayList<int[]>();
    List<DocumentPositionedInfo> problemsList =
        validationProblems.getProblemsList();
    if(problemsList != null) {
        for (int i = 0; i < problemsList.size(); i++) {
            try {
                startEndOffsets.add(currentAuthorPage.getStartEndOffsets(problemsList.get(i)));
            } catch (BadLocationException e) {
                e.printStackTrace();
            }
        }
    }
    if(lastStartEndOffsets.size() != startEndOffsets.size()) {
        //Continue
    } else {
        boolean equal = true;
        for (int i = 0; i < startEndOffsets.size(); i++) {
            int[] o1 = startEndOffsets.get(i);
            int[] o2 = lastStartEndOffsets.get(i);
            if(o1 == null && o2 == null) {
                //Continue
            } else if(o1 != null && o2 != null
                && o1[0] == o2[0] && o1[1] == o2[1]){
                //Continue
            } else {
                equal = false;
                break;
            }
        }
        if(equal) {
            //Same list of problems already displayed.
            return;
        }
    }
}
//Keep last used offsets.
lastStartEndOffsets = startEndOffsets;

try {
if (! SwingUtilities.isEventDispatchThread()) {
    SwingUtilities.invokeLater(new Runnable() {
        @Override
        public void run() {
            //First remove all custom highlights.
            currentAuthorPage.getPersistentHighlighter().removeAllHighlights();
        }
    });
} catch (InterruptedException e) {
    e.printStackTrace();
}
}

if (problemsList != null) {
    for (int i = 0; i < problemsList.size(); i++) {
        //A reported problem (could be warning, could be error).
        DocumentPositionedInfo dpi = problemsList.get(i);
        try {
            final int[] currentOffsets = startEndOffsets.get(i);
            if (currentOffsets != null) {
                //These are offsets in the Author content.
                final LinkedHashMap<String, String> highlightProps =
                    new LinkedHashMap<String, String>();
                highlightProps.put("message", dpi.getMessage());
                highlightProps.put("severity", dpi.getSeverityAsString());
                if (! SwingUtilities.isEventDispatchThread()) {
                    SwingUtilities.invokeLater(new Runnable() {
                        @Override
                        public void run() {
                            currentAuthorPage.getPersistentHighlighter().addHighlight(
                                currentOffsets[0], currentOffsets[1] - 1, highlightProps);
                        }
                    });
                }
            }
        } catch (InterruptedException e) {
            e.printStackTrace();
        } catch (InvocationTargetException e) {
e.printStackTrace();
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}
}

}));
currentEditor.addEditorListener(new WSEditorListener() {
/**
 * @see WSEditorListener#editorAboutToBeSavedVeto(int)
 */
@Override
public boolean editorAboutToBeSavedVeto(int operationType) {
    try {
        if(! SwingUtilities.isEventDispatchThread()) {
            SwingUtilities.invokeLater(new Runnable() {
                @Override
                public void run() {
                    //Remove all persistent highlights before saving
                    currentAuthorPage.getPersistentHighlighter().removeAllHighlights();
                }
            });
        }
    } catch (InterruptedException e) {
        e.printStackTrace();
    } catch (InvocationTargetException e) {
        e.printStackTrace();
    }
    return true;
}
}
}

/**
 * @see WorkspaceAccessPluginExtension#applicationClosing()
 */
public boolean applicationClosing() {
    return true;
}
Add Custom Highlights to Content

Use Case

You want to add custom highlights to the document content in Author mode.

Solution

There are two types of highlights you can add:

1. **Non-PersistentHighlights** - Such highlights are removed when the document is closed and then re-opened.

   You can use the following API method:

   ```java
   ro.sync.exml.workspace.api.editor.page.author.WSAuthorEditorPageBase.getHighlighter()
   ```

   to obtain an `AuthorHighlighter` that allows you to add a highlight between certain offsets with a specified painter.

   For example, you can use this support to implement your own spell checker with a custom highlight for the unrecognized words.

2. **PersistentHighlights** - Such highlights are saved in the XML content as processing instructions.

   You can use the following API method:

   ```java
   ro.sync.exml.workspace.api.editor.page.author.WSAuthorEditorPageBase.getPersistentHighlighter()
   ```

   to obtain an `AuthorPersistentHighlighter` class that allows you to add a persistent highlight between certain offsets, set new properties for a specific highlight, and render it with a specified painter.

   For example, you can use this support to implement your own way of adding review comments.

Related Information:

[Adding Custom Persistent Highlights (on page 2237)](on_page_2237)

Auto-Generate an ID When a Document is Opened or Created

Use Case

You want to configure how the application generates IDs (you need IDs that have a certain format for each created topic).
Solution

This could be done implementing a plugin (on page 3222) for Oxygen XML Editor using the Plugins SDK:

There is a type of plugin called "Workspace Access" that can be used to add a listener to be notified when an editor is opened.

The implemented plugin would intercept the open editor and editor page change events (which occur when a new editor is created) and generate a new ID attribute value on the root element.

The Java code would look like this:

```java
pluginWorkspaceAccess.addEditorChangeListener(new WSEditorChangeListener() {
    /**
     * @see WSEditorChangeListener#editorOpened(java.net.URL)
     */
    @Override
    public void editorOpened(URL editorLocation) {
        WSEditor ed = pluginWorkspaceAccess.getEditorAccess/editorLocation, PluginWorkspace.MAIN_EDITING_AREA);
        generateID(ed);
    }
    /**
     * @see WSEditorChangeListener#editorPageChanged(java.net.URL)
     */
    @Override
    public void editorPageChanged(URL editorLocation) {
        WSEditor ed = pluginWorkspaceAccess.getEditorAccess/editorLocation, PluginWorkspace.MAIN_EDITING_AREA);
        generateID(ed);
    }
    private void generateID(WSEditor ed) {
        if(ed.getCurrentPage() instanceof WSAuthorEditorPage) {
            WSAuthorEditorPage authorEditPage = (WSAuthorEditorPage) ed.getCurrentPage();
            AuthorDocumentController ctrl = authorEditPage.getDocumentController();
            AuthorElement root = ctrl.getAuthorDocumentNode().getRootElement();
            if(root.getAttribute("id") == null || !root.getAttribute("id").getValue().startsWith("generated_")) {
                ctrl.setAttribute("id", new AttrValue("generated_" + Math.random()), root);
            }
        }
    }
}, PluginWorkspace.MAIN_EDITING_AREA);
```
Change the Default Track Changes (Review) Author Name

Use Case

You want to change the default author name used for Track Changes (on page 3224) in the Author Component.

Solution

The Track Changes (Review) author name is determined in the following order:

1. **API** - The review user name can be imposed through the following API:
   
   ```java
   ro.sync.ecss.extensions.api.AuthorReviewController.setReviewerAuthorName(String)
   ```

2. **Options** - If the author name was not imposed from the API, it is determined from the Author option set in the Review preferences page (on page 184).

3. **System properties** - If the author name was not imposed from the API or from the application options then the following system property is used:
   
   ```java
   System.getProperty("user.name")
   ```

So, to impose the Track Changes author, use one of the following approaches:

1. Use the API to impose the reviewer author name. Here is the online Javadoc of this method:
   
   https://www.oxygenxml.com/InstData/Editor/SDK/javadoc/ro/sync/ecss/extensions/api/AuthorReviewController.html#setReviewerAuthorName(java.lang.String)

2. Customize the default options and set a specific value for the Author name option set in the Review preferences page (on page 184).

3. Set the value of user.name system property when the Author Component is initializing and before any document is loaded.

Change the DOCTYPE of an Open XML Document

Use Case

You want to change the DOCTYPE of a document that is open in the Author mode.

Solution

The following API:

```java
ro.sync.ecss.extensions.api.AuthorDocumentController.getDoctype()
```

allows you to get the DOCTYPE of the current XML file open in the Author mode.

There is also an API method available that would allow you to set the DOCTYPE back to the XML:

```java
ro.sync.ecss.extensions.api.AuthorDocumentController.setDoctype(AuthorDocumentType)
```
Here is an example of how this solution would work:

```java
AuthorDocumentType
dt = new AuthorDocumentType("article", "testSystemID", "testPublicID",
    "<!DOCTYPE article PUBLIC "testPublicID" "testSystemID">");
docController.setDoctype(dt);
```

Basically, you could take the entire content from the existing DOCTYPE,

```java
ro.sync.ecss.extensions.api.AuthorDocumentType.getContent()
```

modify it to your needs, and create another `AuthorDocumentType` object with the new content and with the same public, system IDs.

For example, you could use this API if you want to add unparsed entities in the XML DOCTYPE.

### Control XML Serialization in the Oxygen XML Author Component

#### Use Case

You want to force the Oxygen XML Author Component to save the XML with zero indent size and not to break the line inside block elements (on page 3216).

#### Solution

Usually, in a standalone version of Oxygen XML Editor, the `Editor > Format` and `Editor > Format > XML` preferences pages allow you to control the way the XML is saved on the disk after you edit it in the Author mode.

Also, the `APIAccessibleOptionTags` interface contains a list of all accessible keys that can be read or set from the options.

In the Oxygen XML Editor application, you can either bundle a default set of options (on page 309) or use the `PluginWorkspace.setGlobalObjectProperty(String, Object)` API:

```java
//For not breaking the line
//Long line
pluginWorkspace.setObjectProperty
    (APIAccessibleOptionTags.EDITOR_LINE_WIDTH, new Integer(100000));
//Do not break before inline elements
pluginWorkspace.setObjectProperty
    (APIAccessibleOptionTags.EDITOR_FORMAT_INDENT_INLINE_ELEMENTS, false);
//For forcing zero indent
//Force indent settings to be controlled by us
pluginWorkspace.setObjectProperty
    (APIAccessibleOptionTags.EDITOR_DETECT_INDENT_ON_OPEN, false);
//Zero indent size
```
In the Oxygen XML Author Component, you can either bundle a fixed set of options (on page 2500), or use the Java API to set properties that overwrite the default options.

```java
// For not breaking the line
// Long line
AuthorComponentFactory.getInstance().setObjectProperty
(APIAccessibleOptionTags.EDITOR_LINE_WIDTH, new Integer(100000));

// Do not break before inline elements
AuthorComponentFactory.getInstance().setObjectProperty
(APIAccessibleOptionTags.EDITOR_FORMAT_INDENTINLINE_ELEMENTS, false);

// Force indent settings to be controlled by us
AuthorComponentFactory.getInstance().setObjectProperty
(APIAccessibleOptionTags.EDITOR_DETECT_INDENTONOPEN, false);

// Zero indent size
AuthorComponentFactory.getInstance().setObjectProperty
(APIAccessibleOptionTags.EDITOR_INDENT_SIZE, 0);
```

Related Information:
API Interface Documentation: `APIAccessibleOptionTags`

**Customize the Outline View in Text Mode**

**Use Case**

You want to customize the Outline view (on page 539) in Text mode.

**Solution**

Suppose that you have the following XML document:

```xml
<doc startnumber="15">
  <sec counter="no">
    <info/>
    <title>Introduction</title>
  </sec>
  <sec>
    <title>Section title</title>
    <para>Content</para>
  </sec>
  <sec>
    <title>Section title</title>
    <para>Content</para>
  </sec>
</doc>
```
and you want to display the XML content in a simplified Outline view like this:

doc "15"
sec Introduction
sec 15 Section title
sec 15.1 Section title
sec 16 Section title

Usually, an Outline view should have the following characteristics:

1. Double-clicking a node in the Outline view would select the corresponding XML content in the editor.
2. When the cursor moves in the open XML document, the Outline view would select the proper entry.
3. When modifications occur in the document, the Outline view would refresh.

A simple implementation using a Workspace Access plugin type could be something like this:

```java
/**
* Simple Outline for Text mode based on executing XPaths over the text content.
*/
public class CustomWorkspaceAccessPluginExtension implements WorkspaceAccessPluginExtension {

    /**
    * The custom outline list.
    */
    private JList customOutlineList;

    /**
    * Maps outline nodes to ranges in document
    */
    private WSXMLTextNodeRange[] currentOutlineRanges;

    /**
    * The current text page
    */
    private WSXMLTextEditorPage currentTextPage;

    /**
    */
```
private boolean enableCaretListener = true;

/**
 * @see WorkspaceAccessPluginExtension#applicationStarted
 (ro.sync.exml.workspace.api.standalone.StandalonePluginWorkspace)
 */
@Override
public void applicationStarted(
(final StandalonePluginWorkspace pluginWorkspaceAccess)
{
pluginWorkspaceAccess.addViewComponentCustomizer
(new ViewComponentCustomizer() {
/**
 * @see ViewComponentCustomizer#customizeView
 (ro.sync.exml.workspace.api.standalone.ViewInfo)
 */
@Override
public void customizeView(ViewInfo viewInfo) {
    if (viewInfo.getViewID().equals("SampleWorkspaceAccessID")) {
        customOutlineList = new JList();
        //Render the content in the Outline.
        customOutlineList.setCellRenderer(new DefaultListCellRenderer() {
            @Override
            public Component getListCellRendererComponent(JList<?> list, Object value, int index, boolean isSelected, boolean cellHasFocus) {
                JLabel label = (JLabel) super.getListCellRendererComponent(list, value, index, isSelected, cellHasFocus);
                String val = null;
                if (value instanceof Element) {
                    Element element = ((Element)value);
                    val = element.getNodeName();
                    if (!"".equals(element.getAttribute("startnumber"))) {
                        val += " " + "'", element.getAttribute("startnumber") + "'";
                    }
                }
            }
        });
    }
}
NodeList titles = element.getElementsByTagName("title");
    if(titles.getLength() > 0) {
        val += " \
" + titles.item(0).getTextContent() + "\n";
        label.setText(val);
    }
    return label;
});
//When we click a node, select it in the text page.
customOutlineList.addMouseListener(new MouseAdapter() {
    @Override
    public void mouseClicked(MouseEvent e) {
        if(SwingUtilities.isLeftMouseButton(e) && e.getClickCount() == 2) {
            int sel = customOutlineList.getSelectedIndex();
            enableCaretListener = false;
            try {
                currentTextPage.select(currentTextPage.getOffsetOfLineStart(currentOutlineRanges[sel].getStartLine()) +
                currentOutlineRanges[sel].getStartColumn() - 1,
                currentTextPage.getOffsetOfLineStart(currentOutlineRanges[sel].getEndLine()) +
                currentOutlineRanges[sel].getEndColumn());
            } catch (BadLocationException e1) {
                e1.printStackTrace();
            }
            enableCaretListener = true;
        }
    }
});
viewInfo.setComponent(new JScrollPane(customOutlineList));
viewInfo.setTitle("Custom Outline");
});
});
pluginWorkspaceAccess.addEditorChangeListener(new WSEditorChangeListener() {
    /**
     * @see WSEditorChangeListener#editorOpened(java.net.URL)
     */
    @Override
    public void editorOpened(URL editorLocation) {
    }
```java
// An editor was opened
WSEditor editorAccess = pluginWorkspaceAccess.getEditorAccess(editorLocation, StandalonePluginWorkspace.MAIN_EDITING_AREA);
if (editorAccess != null) {
    WSEditorPage currentPage = editorAccess.getCurrentPage();
    if (currentPage instanceof WSXMLTextEditorPage) {
        // User editing in Text mode an open XML document.
        final WSXMLTextEditorPage xmlTP = (WSXMLTextEditorPage) currentPage;
        // Reconfigure outline on each change.
        xmlTP.getDocument().addDocumentListener(new DocumentListener() {
            @Override
            public void removeUpdate(DocumentEvent e) {
                reconfigureOutline(xmlTP);
            }
            @Override
            public void insertUpdate(DocumentEvent e) {
                reconfigureOutline(xmlTP);
            }
            @Override
            public void changedUpdate(DocumentEvent e) {
                reconfigureOutline(xmlTP);
            }
        });
        JTextArea textComponent = (JTextArea) xmlTP.getTextComponent();
        textComponent.addCaretListener(new CaretListener() {
            @Override
            public void caretUpdate(CaretEvent e) {
                if (currentOutlineRanges != null && currentTextPage != null &&
                    enableCaretListener) {
                    enableCaretListener = false;
                    // Find the node to select in the outline.
                    try {
                        int line = xmlTP.getLineOfOffset(e.getDot());
                        for (int i = currentOutlineRanges.length - 1; i >= 0; i--) {
                            if (line > currentOutlineRanges[i].getStartLine() &&
                                line < currentOutlineRanges[i].getEndLine()) {
                                customOutlineList.setSelectedIndex(i);
                                break;
                            }
                        }
                    }
                } catch (BadLocationException e1) {
                    e1.printStackTrace();
                }
            }
        });
    }
}
```
```java
enableCaretListener = true;
}
}
});
}
}
/**
* @see WSEditorChangeListener#editorActivated(java.net.URL)
*/
@Override
public void editorActivated(URL editorLocation) {
    // An editor was selected, reconfigure the common outline
    WSEditor editorAccess = pluginWorkspaceAccess.getEditorAccess(editorLocation, StandalonePluginWorkspace.MAIN_EDITING_AREA);
    if (editorAccess != null) {
        WSEditorPage currentPage = editorAccess.getCurrentPage();
        if (currentPage instanceof WSXMLTextEditorPage) {
            // User editing in Text mode an open XML document.
            WSXMLTextEditorPage xmlTP = (WSXMLTextEditorPage) currentPage;
            reconfigureOutline(xmlTP);
        }
    }
}, StandalonePluginWorkspace.MAIN_EDITING_AREA);
}
/**
* Reconfigure the outline
* *
* @param xmlTP The XML Text page.
*/
protected void reconfigureOutline(final WSXMLTextEditorPage xmlTP) {
    try {
        // These are DOM nodes.
        Object[] evaluateXPath = xmlTP.evaluateXPath("//doc | //sec");
        // These are the ranges each node takes in the document.
        currentOutlineRanges = xmlTP.findElementByXPath("//doc | //sec");
        currentTextPage = xmlTP;
        DefaultListModel listModel = new DefaultListModel();
        if (evaluateXPath != null) {
```
for (int i = 0; i < evaluateXPath.length; i++) {
    listModel.addElement(evaluateXPath[i]);
}
customOutlineList.setModel(listModel);
} catch (XPathException ex) {
    ex.printStackTrace();
}

/**
 * @see WorkspaceAccessPluginExtension#applicationClosing()
 */
@Override
public boolean applicationClosing() {
    return true;
}

---

Disable Context-Sensitive Menu Items for Custom Author Actions

**Use Case**

You want to disable menu items for custom Author mode actions depending on the cursor context.

**Solution**

By default, Oxygen XML Editor does not toggle the enabled/disabled states for actions based on whether or not the activation XPath expressions for that certain Author mode action are fulfilled. This is done because the actions can be many and evaluating XPath expression on each cursor move can lead to performance problems. However, if you have your own ro.sync.ecss.extensions.api.ExtensionsBundle implementation you can overwrite the method:

```java
to.sync.ecss.extensions.api.ExtensionsBundle.createAuthorExtensionStateListener()
```

and when the extension state listener gets activated, you can use the API like this:

```java
/**
 * @see ro.sync.ecss.extensions.api.AuthorExtensionStateListener#activated
 (ro.sync.ecss.extensions.api.AuthorAccess)
 */
public void activated(final AuthorAccess authorAccess) {
    // Add a caret listener to enable/disable extension actions:
```
authorAccess.getEditorAccess().addAuthorCaretListener(new AuthorCaretListener()
{
  @Override
  public void caretMoved(AuthorCaretEvent caretEvent) {
    try {
      Map<String, Object> authorExtensionActions =
      authorAccess.getEditorAccess().getActionsProvider().getAuthorExtensionActions();
      //Get the action used to insert a paragraph. It's ID is "paragraph"
      AbstractAction insertParagraph = (AbstractAction) authorExtensionActions.get("paragraph");
      //Evaluate an XPath expression in context of the current node
      Object[] evaluateXPath = authorAccess.getDocumentController().evaluateXPath
      ("*[ancestor-or-self::p]", false, false, false, false);
      if (evaluateXPath != null && evaluateXPath.length > 0 &&
      evaluateXPath[0] != null) {
        //We are inside a paragraph, disable the action.
        insertParagraph.setEnabled(false);
      } else {
        //Enable the action
        insertParagraph.setEnabled(true);
      }
    } catch (AuthorOperationException e) {
      e.printStackTrace();
    }
  } //End of method
});

When the extension is deactivated, you should remove the CaretListener to avoid adding multiple listeners that perform the same functionality.

**Dynamically Add Form Controls Using a Styles Filter**

**Use Case**

You want to add form controls using an API.

**Solution**

Usually, a form control is added from the CSS using one of the built-in form controls (on page 2373). However, in some cases you do not have all the information you need to properly initialize the form control at CSS level. In these cases you can add the form controls by using the API, more specifically ro.sync.ecss.extensions.api.StylesFilter.

For instance, if you want a combo box form control and the values to populate the combo are specified inside a file (or they come from a database). Here is how to add the form control from the API:
public class SDFStylesFilter implements StylesFilter {

    public Styles filter(Styles styles, AuthorNode authorNode) {
        if (authorNode.getType() == AuthorNode.NODE_TYPE_PSEUDO_ELEMENT
                && "before".equals(authorNode.getName())) {
            authorNode = authorNode.getParent();
            if ("country".equals(authorNode.getName())) {
                // This is the BEFORE pseudo element of the "country" element.
                // Read the supported countries from the configuration file.
                Map<String, Object> formControlArgs = new HashMap<String, Object>();
                formControlArgs.put(InplaceEditorArgumentKeys.PROPERTY_EDIT, "#text");
                formControlArgs.put(InplaceEditorArgumentKeys.PROPERTY_TYPE,
                        InplaceEditorArgumentKeys.TYPE_COMBOBOX);
                String countries = readCountriesFromFile();
                formControlArgs.put(InplaceEditorArgumentKeys.PROPERTY_VALUES, countries);
                formControlArgs.put(InplaceEditorArgumentKeys.PROPERTY_EDITABLE, "false");

                // We also add a label in form of the form control.
                Map<String, Object> labelProps = new HashMap<String, Object>();
                labelProps.put("text", "Country: ");
                labelProps.put("styles", "* {width: 100px; color: gray;}");
                StaticContent[] mixedContent = new StaticContent[]{
                    new LabelContent(labelProps),
                    new EditorContent(formControlArgs)};
                styles.setProperty(Styles.KEY_MIXED_CONTENT, mixedContent);
            }
        }

        // The added form control is the only way the element can be edited.
        if ("country".equals(authorNode.getName())) {
            styles.setProperty(Styles.KEY_VISIBILITY, "-oxy-collapse-text");
        }

        return styles;
    }
}

If the execution of the `formControlArgs.put(InplaceEditorArgumentKeys.PROPERTY_VALUES, countries);` line consumes too much execution time (for example, if it connects to a database or if it needs to extract data from a very large file), you can choose to delay it until the values are actually needed by the form control. This approach is called *lazy evaluation* and can be implemented as follows:
formControlArgs.put(InplaceEditorArgumentKeys.PROPERTY_VALUES,
    new LazyValue<List<CIValue>>() {
        public java.util.List<CIValue> get() {
            // We avoid reading the possible values until they are actually requested.
            // This will be a List with CIValues created over countries:
            France, Spain, Great Britain
            return readCountriesFromFile();
        }
    });

The lazy evaluation approach can be used for the following form controls properties:

- InplaceEditorArgumentKeys.PROPERTY_VALUES
- InplaceEditorArgumentKeys.PROPERTY_LABELS
- InplaceEditorArgumentKeys.PROPERTY_TOOLTIPS

The full source code for this example is available inside the Oxygen SDK.

Dynamically Modify the Content Inserted by the Author

Use Case

You want to insert typographic quotation marks instead of double quotes.

Solution

By using the API you can set a document filter to change the text that is inserted in the document in Author mode. You can use this method to change the insertion of double quotes with the typographic quotes.

Here is some sample code:

authorAccess.getDocumentController().setDocumentFilter
(new AuthorDocumentFilter()) {
    /**
     * @see ro.sync.ecss.extensions.api.AuthorDocumentFilter#insertText
     * (ro.sync.ecss.extensions.api.AuthorDocumentFilterBypass, int, java.lang.String)
     */
    @Override
    public void insertText(AuthorDocumentFilterBypass filterBypass,
        int offset, String toInsert) {
        if(toInsert.length() == 1 && \"".equals(toInsert)) {
            //User typed a quote but he actually needs a smart quote.
            //So we either have to add \u201E (start smart quote)
            //Or we add \u201C (end smart quote)
            //Depending on whether there's already a start smart quote inserted
            in the current paragraph.
    }
try {
    AuthorNode currentNode =
authorAccess.getDocumentController().getNodeAtOffset(offset);
    int startOfTextInCurrentNode = currentNode.getStartOffset();
    if(offset > startOfTextInCurrentNode) {
        Segment seg = new Segment();
        authorAccess.getDocumentController().getChars(startOfTextInCurrentNode,
        offset - startOfTextInCurrentNode, seg);
        String previosTextInNode = seg.toString();
        boolean insertStartQuote = true;
        for (int i = previosTextInNode.length() - 1; i >= 0; i--) {
            char ch = previosTextInNode.charAt(i);
            if ('\u201C' == ch) {
                //Found end of smart quote, so yes, we should insert a start one
                break;
            } else if ('\u201E' == ch) {
                //Found start quote, so we should insert an end one.
                insertStartQuote = false;
            }
        }
        if(insertStartQuote) {
            toInsert = "\u201E";
        } else {
            toInsert = "\u201C";
        }
    }
} catch (BadLocationException e) {
    e.printStackTrace();
}
System.err.println("INSERT TEXT |" + toInsert + "|");
super.insertText(filterBypass, offset, toInsert);
});
An alternative to using a document filtering is the use of a
`ro.sync.ecss.extensions.api.AuthorSchemaAwareEditingHandlerAdapter`, which has clear callbacks indicating
the source from where the API is called (Paste, Drag and Drop, Typing).

**Extend the Java Functionality of an Existing Framework (Document Type)**

**Use Case**

You want to change the way a DocBook 4 `<xref>` displays in Author mode based on what element is at the
@linkend.

**Solution**

Follow these steps:

1. Create a Maven Java project and add a dependency on the Oxygen XML Editor classes:

   ```xml
   <dependency>
   <groupId>com.oxygenxml</groupId>
   <artifactId>oxygen-sdk</artifactId>
   <version>${oxygen.version}</version>
   </dependency>
   ```

   where `${oxygen.version}` is the version of Oxygen XML Editor.

   Alternatively, if the project does not use Maven, all the transitive dependencies of the above Maven
   artifact need to be added to the classpath of the project.

2. Also add the `\[OXYGEN_INSTALL_DIR\]/frameworks/docbook/docbook.jar` to the class path of
   the project.

3. Create a class that extends `ro.sync.ecss.extensions.docbook.DocBook4ExtensionsBundle` and
   overwrites the method: `ro.sync.ecss.extensions.api.ExtensionsBundle#createLinkTextResolver()`.

4. For your custom resolver implementation you can start from the Java sources of the
   `ro.sync.ecss.extensions.docbook.link.DocbookLinkTextResolver` (the Java code for the entire DocBook
   customization is present in a subfolder in the Oxygen SDK).

5. Pack your extension classes in a JAR (on page 3220) file. Copy the JAR to:
   `\[OXYGEN_INSTALL_DIR\]/frameworks/docbook/custom.jar`.


7. **Open the Preferences dialog box (Options > Preferences) (on page 124)** and go to Document Type
   **Association**. Edit the DocBook 4 document type. In the Classpath list add the path to the new JAR. In
   the extensions list select your custom extension instead of the regular DocBook one.

8. You can rename the document type and the `docbook framework` folder to something else (such as
   `custom_docbook`) and share it with others. A document type can also be installed using the add-on
   support (on page 2289).
Impose Custom Options for Authors

Use Case

You want to force Track Changes (on page 3224) to be enabled at startup.

Solution

There are two ways to enable Track Changes for every document that you open:

1. You could customize the default options (on page 309) that are used by your authors and set the Track Changes - Initial State option (on page 183) to Always On.
2. Use an API to toggle the Track Changes state after a document is opened in Author mode:

```java
// Check the current state of Track Changes
boolean trackChangesOn = authorAccess.getReviewController().isTrackingChanges();
if (!trackChangesOn) {
    // Set Track Changes state to On
    authorAccess.getReviewController().toggleTrackChanges();
}
```

Insert an Element with all the Required Content

Use Case

You want to insert a DITA image element that points to a certain resource and has required content and you want the required content be automatically inserted.

Solution

The API ro.sync.ecss.extensions.api.AuthorSchemaManager can propose valid elements that can be inserted at the specific offset. Using the method AuthorSchemaManager.createAuthorDocumentFragment(CIElement), you can convert the proposed elements to document fragments (on page 3218) (which have all the required content filled in) that can then be inserted in the document.

```java
AuthorSchemaManager schemaManager =
    this.authorAccess.getDocumentController().getAuthorSchemaManager();

WhatElementsCanGoHereContext context =
    schemaManager.createWhatElementsCanGoHereContext(
        this.authorAccess.getEditorAccess().getCaretOffset());

List<CIElement> possibleElementsAtCaretPosition =
    schemaManager.whatElementsCanGoHere(context);

loop: for (int i = 0; i < possibleElementsAtCaretPosition.size(); i++) {
    CIElement possibleElement = possibleElementsAtCaretPosition.get(i);
    List<CIAAttribute> attrs = possibleElement.getAttributes();
```
if (attrs != null) {
    for (int j = 0; j < attrs.size(); j++) {
        CIAttribute ciAttribute = attrs.get(j);
        if (ciAttribute.getName().equals("class")) {
            if (ciAttribute.getDefaultValue() != null
                && ciAttribute.getDefaultValue().contains(" topic/image ")) {
                //Found a CIElement for image
                //Create a fragment that contains all required child elements already built.
                AuthorDocumentFragment frag =
                    schemaManager.createAuthorDocumentFragment(possibleElement);
                //Now set the @href to it.
                //Ask the user and obtain a value for the @href
                //Then:

                String href = "test.png";
                List<AuthorNode> nodes = frag.getContentNodes();
                if (!nodes.isEmpty()) {
                    AuthorElement imageEl = (AuthorElement) nodes.get(0);
                    imageEl.setAttribute("href", new AttrValue(href));
                }
                //And insert the fragment.
                this.authorAccess.getDocumentController().insertFragment
                    (this.authorAccess.getEditorAccess().getCaretOffset(), frag);
                break loop;
            }
        }
    }
}

Related Information:
AuthorDocumentFragment Class

Modify the XML Content on Open

Use Case
You want to convert fixed paths in an attribute value to relative paths.

Solution
The Plugins SDK contains a sample plugin type called WorkspaceAccess. Such a plugin is notified when the application starts and it can do what you want in a couple of ways:
1. Add a listener that notifies you when the user opens an XML document. Then if the XML document is opened in the Author visual editing mode you can use the Author API to change attributes:

```java
pluginWorkspaceAccess.addEditorChangeListener(new WSEditorChangeListener() {
  /**
   * @see WSEditorChangeListener#editorOpened(java.net.URL)
   */
  @Override
  public void editorOpened(URL editorLocation) {
    WSEditor openedEditor = pluginWorkspaceAccess.getCurrentEditorAccess
    (StandalonePluginWorkspace.MAIN_EDITING_AREA);
    if (openedEditor.getCurrentPage() instanceof WSAuthorEditorPage) {
      WSAuthorEditorPage authPage = (WSAuthorEditorPage)
      openedEditor.getCurrentPage();
      AuthorDocumentController docController =
      authPage.getDocumentController();
      try {
        //All changes will be undone by pressing Undo once.
        docController.beginCompoundEdit();
        fixupImageRefs(docController,
        docController.getAuthorDocumentNode());
      } finally {
        docController.endCompoundEdit();
      }
    }
  }
}

private void fixupImageRefs
(AuthorDocumentController docController, AuthorNode authorNode) {
  if (authorNode instanceof AuthorParentNode) {
    //Recursively
    List<AuthorNode> contentNodes =
    ((AuthorParentNode)authorNode).getContentNodes();
    if (contentNodes != null) {
      for (int i = 0; i < contentNodes.size(); i++) {
        fixupImageRefs(docController, contentNodes.get(i));
      }
    }
  }
  if (authorNode.getType() == AuthorNode.NODE_TYPE_ELEMENT) {
    AuthorElement elem = (AuthorElement) authorNode;
    if ("image".equals(elem.getLocalName())) {
      if (elem.getAttribute("href") != null) {
```
2. An API to open XML documents in the application:

```java
ro.sync.exml.workspace.api.Workspace.open(URL)
```

So you can create a plugin that automatically opens XML documents one at a time from a certain folder in the application, makes modifications to them, and saves the content by calling:

```java
ro.sync.exml.workspace.api.editor.WSEditorBase.save()
```

then closes the editor by calling:

```java
ro.sync.exml.workspace.api.Workspace.close(URL)
```

### Modify the XML Content on Save

#### Use Case

You the revised date on a DITA document to be updated when it is saved.

#### Solution

The Plugins SDK contains a sample plugin type called **WorkspaceAccess**. Such a plugin is notified when the application starts.

You can add a listener that notifies you before the user saves an XML document. Then if the XML document is opened in the **Author** visual editing mode you can use the **Author API** to change attributes before the save takes place:

```java
@override
public void applicationStarted(final StandalonePluginWorkspace pluginWorkspaceAccess) {
    pluginWorkspaceAccess.addEditorChangeListener(new WSEditorChangeListener() {
        @Override
        public void editorOpened(URL editorLocation) {
            final WSEditor editorAccess = pluginWorkspaceAccess.getEditorAccess
```
Multiple Rendering Modes for the Same Document in Author Mode

Use Case

You want to add multiple buttons, each showing a different visualization mode of the same document, in Author mode.
Solution

In the toolbar in Author mode, there is a Styles drop-down menu that contains alternate CSS styles (on page 3216) for the same document. To add an alternate CSS stylesheet, open the Preferences dialog box (Options > Preferences) (on page 124), go to Document Type Association, select the document type associated with your documents and click Edit. In the Document Type configuration dialog box (on page 140) that appears, go to the Author tab, and in the CSS subtab add references to alternate CSS stylesheets.

For example, one of the alternate CSS stylesheets offered for the DITA document type is located here (by default):

\[OXYGEN\_INSTALL\_DIR\]/frameworks/dita/css_classed/hideColspec.css

If you open it, you will see that it imports the main CSS (on page 3220) and then adds selectors of its own.

Obtain the Currently Selected Element Using the Author API

Use Case

In Author mode, if an element is fully selected, you want to perform an action on it. If not, you want to perform an action on the node that is located at the cursor position.

Solution

When an element is fully selected by the user the selection start and end offsets are actually outside of the node’s offset bounds. So using AuthorDocumentController.getNodeAtOffset will actually return the parent of the selected node. A special API is available that makes it easier for you to determine this situation: WSAuthorEditorPageBase.getFullySelectedNode().

```java
AuthorDocumentController controller = authorPageAccess.getDocumentController();
AuthorAccess authorAccess = authorPageAccess.getAuthorAccess();
int caretOffset = authorAccess.getEditorAccess().getCaretOffset();

AuthorElement nodeAtCaret =
    (AuthorElement) authorAccess.getEditorAccess().getFullySelectedNode();
if (nodeAtCaret == null) {
    // We have no fully selected node. We can look at the cursor offset.
    nodeAtCaret = (AuthorElement) authorAccess.getDocumentController().getNodeAtOffset(caretOffset);
}
```

// Or we could look at the selection start and end, see which node is the parent of each offset and get the closest common ancestor.
Open a Document from Another Application

**Restriction:**
This feature is currently only available for macOS users.

**Use Case**
You want to open a document from another application in Oxygen XML Editor.

**Solution**
The Oxygen XML Editor installation kit for macOS comes with a special protocol handler that can be used if you want to open remote resources in the application (for example, opening a file from a CMS). The protocol is `edit-in-oxygen` and you can use it from a command line like this:

```bash
open edit-in-oxygen:protocol://host/path/file.xml
```

For example, if you start the following from the command line:

```bash
```

Oxygen XML Editor will start and open the HTML content from the URL `http://www.oxygenxml.com/index.html`.

**Tip:**
You can also use anchors on the URL to point to specific lines or elements inside the open document:

[Opening a Document at a Specific Location Using a Command-Line Interface](on page 383)

Run XSLT or XQuery Transformations

**Use Case**
You want to run XSL 2.0 / 3.0 transformations with Saxon EE using the Oxygen SDK.

**Solution**
The API class `ro.sync.exml.workspace.api.util.XMLUtilAccess` allows you to create an XSLT Transformer that implements the JAXP interface `javax.xml.transform.Transformer`. Then this type of transformer can be used to transform XML. Here's just an example of transforming when you have an `AuthorAccess API` available:

```java
InputSource is = new org.xml.sax.InputSource
(URLUtil.correct(new File("test/personal.xsl"))).toString);
   xs1Src = new SAXSource(is);
   javax.xml.transform.Transformer transformer =
   authorAccess/XMLUtilAccess().createXSLTTransformer
   (xs1Src, null, AuthorXMLUtilAccess.TRANSFORMER_SAXON_ENTERPRISE_EDITION);
   transformer.transform(new StreamSource(new File("test/personal.xml")),
   new StreamResult(new File("test/personal.html")));
```
If you want to create the transformer from the plugin side, you can use this method instead:


**Save a New Document with a Predefined File Name Pattern**

**Use Case**

You want Oxygen XML Editor to automatically generate a file name comprising a UUID plus file extension using the SDK.

**Solution**

This could be done implementing a plugin (on page 3222) for Oxygen XML Editor using the Plugins SDK.

There is a type of plugin called *Workspace Access* that can be used to add a listener to be notified before an opened editor is saved. The implemented plugin would intercept the save events when a newly created document is untitled and display an alternative chooser dialog box, then save the topic with the proper name.

The Java code would look like this:

```java
private static class CustomEdListener extends WSEditorListener{
    private final WSEditor editor;
    private final StandalonePluginWorkspace pluginWorkspaceAccess;
    private boolean saving = false;
    public CustomEdListener(StandalonePluginWorkspace pluginWorkspaceAccess, WSEditor editor) {
        this.pluginWorkspaceAccess = pluginWorkspaceAccess;
        this.editor = editor;
    }

    @Override
    public boolean editorAboutToBeSavedVeto(int operationType) {
        if(! saving &&
            editor.getEditorLocation().toString().contains("Untitled")) {
            File chosenDir = pluginWorkspaceAccess.chooseDirectory();
            if(chosenDir != null) {
                File chosenFile =
                    new File(chosenDir, UUID.randomUUID().toString() + ".dita");
                SwingUtilities.invokeLater(new Runnable() {
                    @Override
                    public void run() {
                        saving = true;
                        editor.saveAs(new URL(chosenFile.toURI().toASCIIString()));
                    }
                });
            }
        }
    }
}
```
```
e.printStackTrace();
}
finally {
    saving = false;
}
}
});

//Reject the original save request.
return false;
}
return true;
}
}

@Override
public void applicationStarted(final StandalonePluginWorkspace pluginWorkspaceAccess) {
    pluginWorkspaceAccess.addEditorChangeListener(new WSEditorChangeListener() {
        @Override
        public void editorOpened(URL editorLocation) {

            final WSEditor editor = pluginWorkspaceAccess.getEditorAccess(editorLocation, PluginWorkspace.MAIN_EDITING_AREA);
            if (editor != null && editor.getEditorLocation().toString().contains("Untitled")) {

                //Untitled editor
                editor.addEditorListener(new CustomEdListener(pluginWorkspaceAccess, editor));
            }
        }
    }, PluginWorkspace.MAIN_EDITING_AREA);
}

Split Paragraph on Enter (Instead of Showing Content Completion List)

Use Case

You want to split the paragraph on Enter instead of showing the content completion list.

Solution

Yes, it is possible by creating your own custom operation.
To obtain this behavior, follow this procedure:

1. Create a custom Author mode operation (on page 2177) that handles the split. You can use the AuthorDocumentController.split API to achieve this.
2. Create a JAR library that contains its compiled version.
3. Open the Preferences dialog box (Options > Preferences) (on page 124), go to Document Types Association, and select your framework.
4. Click Edit and in the Document Type configuration dialog box (on page 140), go to the Classpath tab (on page 144) and add a reference to the JAR library for your custom operation.
5. Go to the Author tab, then go to the Actions subtab.
6. Click the + New button and use the Action dialog box (on page 148) to create your own paragraph split action.
7. Make sure you assign Enter as the Shortcut Key and specify your custom operation in the Operations section.

Result: Now, when you press Enter, your Java operation will be invoked to split the paragraph instead of opening the Content Completion Assistant.

Tip: The Content Completion Assistant can still be invoked by using the Ctrl+Space keyboard shortcut.

Use Custom Rendering Styles for Entity References, Comments, or PIs

Use Case

You want to display entity references in the Author mode without the distinct gray background and tag markers.

Solution

There is a built-in CSS stylesheet in the Oxygen XML Editor libraries that is used when styling content in the Author mode, no matter what CSS you use. This CSS has the following content:

```css
@namespace oxy url('http://www.oxygenxml.com/extensions/author');
@namespace xi "http://www.w3.org/2001/XInclude";
@namespace xlink "http://www.w3.org/1999/xlink";
@namespace svg "http://www.w3.org/2000/svg";
@namespace mml "http://www.w3.org/1998/Math/MathML";

oxy|document {
    display: block !important;
}

oxy|cdata {
```


display: morph !important;
white-space: pre-wrap !important;
border-width: 0px !important;
margin: 0px !important;
padding: 0px !important;

}

oxy|processing-instruction {

display: block !important;
color: rgb(139, 38, 201) !important;
white-space: pre-wrap !important;
border-width: 0px !important;
margin: 0px !important;
padding: 0px !important;

}

oxy|comment {

display: morph !important;
color: rgb(0, 100, 0) !important;
background-color: rgb(255, 255, 210) !important;
white-space: pre-wrap !important;
border-width: 0px !important;
margin: 0px !important;
padding: 0px !important;

}

oxy|reference:before,
 oxy|entity[href]:before{
   link: attr(href) !important;
text-decoration: underline !important;
color: navy !important;

margin: 2px !important;
padding: 0px !important;

}

oxy|reference:before {

display: morph !important;
content: url(../images/editContent.gif) !important;

}

oxy|entity[href]:before{

```
display: morph !important;
content: url(../images/editContent.gif) !important;
}

oxy|reference,
oxy|entity {
  editable: false !important;
  background-color: rgb(240, 240, 240) !important;
  margin: 0px !important;
  padding: 0px !important;
}

oxy|reference {
  display: morph !important;
}

oxy|entity {
  display: morph !important;
}

oxy|entity[href] {
  border: 1px solid rgb(175, 175, 175) !important;
  padding: 0.2em !important;
}

xi|include {
  display: block !important;
  margin-bottom: 0.5em !important;
  padding: 2px !important;
}

xi|include:before,
xi|include:after {
  display: inline !important;
  background-color: inherit !important;
  color: #444444 !important;
  font-weight: bold !important;
}

xi|include:before {
  content: url(../images/link.gif) attr(href) !important;
  link: attr(href) !important;
}
```
/* No direct display of the MathML and SVG images. */

svg

In the CSS used for rendering the XML in Author mode, do the following:

1. Import the special Author mode namespace.
2. Use a special selector to customize the entity node.

Example:

```css
@namespace oxy url("http://www.oxygenxml.com/extensions/author");

oxy|entity {
  background-color: inherit !important;
  margin: 0px !important;
  padding: 0px !important;
  -oxy-display-tags:none;
}
```

You can overwrite styles in the predefined CSS to customize style comments, processing instructions, and CDATA sections. You can also customize the way `<xi:include>` elements are rendered.
19. Add-ons

Oxygen XML Editor offers various default add-ons (on page 3222) that can be installed to provide additional functionality to Oxygen XML Editor. Some additional community submissions are also available, although community add-ons are not officially supported or endorsed. For a full list of add-ons that are officially supported for Oxygen XML Editor, see Oxygen XML Add-on Repositories.

This chapter contains information about the default add-ons that are available to install directly from Oxygen XML Editor.

To install one of the default add-ons, follow this procedure:

1. Go to Help > Install new add-ons to open an add-on selection dialog box.
2. Enter or paste https://www.oxygenxml.com/InstData/Addons/default/updateSite.xml in the Show add-ons from field or select it from the drop-down menu.
3. Select the add-on you want to install and click Next.
4. Read the end-user license agreement. Then select the I accept all terms of the end-user license agreement option and click Finish.
5. Restart the application.

Collaboration
Git Client
Content Fusion Connector
Feedback Connector
Migration/Conversions
Batch Documents Converter
DITA Editing
DITA Prolog Updater
DITA References View
Terminology
Terminology Checker
Vale Linter for Markdown and HTML Validation
Translation
Translator Helper
DITA Translation Package Builder
Productivity
Smart Autocomplete (Experimental)
Emmet Abbreviations
Writer Helper (Experimental)
Development
Resources

For more information about extending Oxygen XML Editor through add-ons, see the following webinars/presentations:

- Webinar: Extending the Functionality of Oxygen Using Add-ons
- Webinar: Add-ons You Can Use for Technical Writing
- XML Prague Conference Presentation: All About Oxygen Add-ons

Collaboration

Git Client Add-on

An add-on is available that contributes a built-in Git client directly in Oxygen XML Editor. Once the add-on is installed, a Git Staging view is available that includes various actions that perform common Git commands, such as push, pull, change branch, commit, and more. It also includes a built-in tool for comparing and merging changes.

Quick Installation

You can drag the following Install button and drop it into the main editor in Oxygen (version 24.1 or newer) to quickly initiate the installation process:

Install

Manual Installation

To manually install the add-on, follow this procedure:

1. Go to Help > Install new add-ons to open an add-on selection dialog box. Enter or paste https://www.oxygenxml.com/InstData/Addons/default/updateSite.xml in the Show add-ons from field or select it from the drop-down menu.
2. Select the Git Client add-on and click Next.
3. Read the end-user license agreement. Then select the **I accept all terms of the end-user license agreement** option and click **Finish**.

4. Restart the application.

**Result:** The **Git Staging** view is now available. To open the view, select **Git Staging** from the **Window > Show View** menu (or select **Git Client** from the **Tools** menu). This view acts as a basic Git client integrated directly in Oxygen XML Editor, and it provides support for committing changes to a Git repository, comparing and merging changes, resolving conflicts, and other Git commands. A **Git** menu is also contributed in the main menu bar that includes various Git actions.

**Git Staging View Interface**

Once the **Git Client** add-on is installed, the **Git Staging** view is available by selecting it from the **Window > Show View** menu (or select **Git Client** from the **Tools** menu). The **Git Staging** view is the main interface where most of the actions that trigger Git commands and other features can be accessed.

**Figure 608. Git Staging View**

The **Git Staging** view includes the following actions/features (most are also available in the **Git** menu):

**↑ Push**

Pushes your local repository changes to the remote repository.
Pull
Pulls the changes from the remote repository into your local repository.

Stash drop-down menu
Includes the following submenu items:

Stash changes
Creates a new stash from the working copy changed file.

List stashes
Opens a dialog box that lists the existing stashes and you can choose to apply or remove a stash.

Show current repository history
Opens the Git History view (on page 2521) at the bottom of the application.

Show Git Branch Manager view
Opens the Git Branch Manager view (on page 2525) on the right side of the application.

Ellipsis (three vertical dots) menu
Includes the following submenu items:

Clone new repository
Clones a repository into a new directory.

Push
Pushes your local repository changes to the remote repository.

Pull (merge)
Pulls the latest changes from the remote repository and combines them with your local unpublished changes.

Pull (rebase)
Pulls the latest changes from the remote repository and combines them with your local unpublished changes.

Show branches
Opens the Git Branch Manager view (on page 2525) on the right side of the application.

Show tags
Opens a dialog box that lists all the tags.

Show history
Opens the Git History view (on page 2521) at the bottom of the application.
**Submodule**

Opens a combo box where you can select the desired submodule to open and work with.

**Stash changes**

Creates a new stash from the working copy changed file.

**List changes**

Opens a dialog box that lists the existing stashes and you can choose to apply or remove a stash.

**Manage remote repositories**

Opens a dialog box where you can add, edit, or delete existing remote repositories.

**Track remote branch**

Opens a dialog box where you can select a remote branch that the local branch will track for executing fetch, push, or pull commands.

**Edit repository "config" file**

Opens the configuration file for the repository in the main editor so that it can be edited.

**Preferences**

Opens the Git Client preferences page where you can configure some options that relate to the add-on.

**Reset all credentials**

Resets all credentials to the default values.

**Unstaged files area**

Any changed, unstaged files are listed in this box. The following actions are available at the top-right corner of the box:

- **Stage selected**
  
  Moves the selected unstaged, changed files to the Staged files area.

- **Stage all**
  
  Moves all unstaged, changed files to the Staged files area.

- **Switch to tree view**
  
  Switches to a tree-like view.

**Staged files area**

All staged files are listed in this box. The following actions are available at the top-right corner of the box:
Unstage selected
Moves the selected staged files to the Unstaged files area.

Unstage all
Moves all staged files to the Unstaged files area.

Switch to tree view
Switches to a tree-like view.

Commit message area
This box in this area is used to write a commit message. The following actions are available at the top-right corner of the box:

Choose a previously used comment
Allows you to re-use a commit message that you used in the past.

Automatically push changes to remote branch
When a commit is performed, the committed changes are also pushed to the remote repository.

Amend last commit
Allows you to edit the selected commit. You can combine staged changes with the previous commit instead of creating an entirely new commit. It can also be used to simply edit the previous commit message without changing its snapshot. This action should not be performed on public commits (commits that were pushed to the remote repository).

Cloning a Repository
Click the Clone new repository button (it has a ‘+’ sign as the icon) and provide the following:

- Repository URL - The URL of the remote repository to be cloned.
- Checkout branch - A specific branch of the repository that is being cloned. The default branch will be cloned if another one is not specified.
- Destination path - The local path where the repository will be cloned.

After cloning a repository, it will automatically be set as the current working copy.

Making an Oxygen Project a Git Repository
When showing the Git Staging side-view after opening an Oxygen XML Editor project that is not already a Git repository, Oxygen XML Editor offers the possibility to make that project also a local Git repository. This is very useful, for example, for newly created projects. After creating the local repository, bind it to a remote repository. You will be asked to specify the URL of the corresponding remote repository on your first attempt to push or pull changes.
Authentication

The Git Client supports both HTTPS and SSH connections to GitHub, GitLab, Bitbucket, and more.

To access the remote repository, you need to provide your authentication details (if not using unprotected SSH keys). If no such information is found in the add-on's settings, you will be prompted for them.

The Git Client allows you to authenticate over HTTPS by using either a basic authentication method (username + password) or a personal access token.

Notes:
- The authentication using personal access tokens has been tested with GitHub and GitLab.
- Bitbucket uses a concept similar to personal access tokens, named app passwords. An app password must be provided as the password for Basic authentication, along with the correct username.
- As of August 13, 2021, GitHub will no longer accept password-based authentication.

If you have the two-factor authentication (2FA) enabled for GitHub or GitLab, to authenticate over HTTPS, you must generate a personal access token for your profile, and back in the Git Staging view in Oxygen XML Editor, use the generated token value as the authentication password when asked for your credentials.

If, for example, you have been using a GitHub account but you decide to switch to another GitHub account, you would need to reset your credentials so that you will be prompted for new ones. This is because only one set of credentials for each Git platform/server is stored. To reset your credentials, go to the toolbar at the top of the Git Staging side-view, click the settings icon (a cogwheel), and select Reset all credentials.

Selecting a Working Copy

Click the Browse button to the right of the Working copy combo box to select a working copy from your file system. The selected folder must be a Git repository.

Switching Between Local Branches

To switch between local branches from the Git Staging view, use the Branch combo box. Local branches can also be changed using the Git Branch Manager (on page 2525).

New branches can be created from the Git Branch Manager (on page 2525) or from the History table using the Create branch action in the contextual menu.

Stashing Files

A Git stash is a way of temporarily saving the changes you have made to your working copy so that you can continue to use other Git operations in the meantime, and you can then re-apply the stashed changes later.
On the toolbar, there is a **Stash** drop down menu with a **Stash changes** action for creating a new stash out of the working copy changed file and a **List stashes** action that presents a dialog box with the existing stashes. From, the **List stashes** dialog box, you can also choose to apply or remove a stash.

### Creating Tags

Tagging is used to capture a specific point in history, for example to mark a release. A tag is like a branch that doesn't change. You can create tags from the **History view** contextual menu. In the **Ellipsis (three vertical dots)** menu on the right side of the **Git Staging** view's toolbar, there is a **Show tags** action that lists all the tags. For tags present only in the local branch, you can choose to delete them or to push them to the remote repository. The **Show tags** action is also available in the **Git** menu from the main menu bar.

### Working with Submodules

When cloning a repository that contains submodules, all submodules are initialized and cloned as well. When pulling changes from the remote repository, the submodules are also updated. The update of the submodules when performing a pull operation depends on the **Update all submodules after pulling changes from the remote repository** option from **Options > Preferences > Plugins > Git Client** (the option is enabled by default).

To open and work with a Git submodule, use the **Submodules** action from the **Ellipsis (three vertical dots)** menu (on the right side of the **Git Staging** view's toolbar) and select the desired submodule from the presented combo box. As an alternative, if the submodule is modified and is presented in the **Unstaged files area** (on page 2523), the **Open** contextual menu action can be used to open it.

The tooltip of a modified submodule shown in the **Unstaged files area** presents information about the currently and previously tracked commits.

### Showing the Repository History

To show the history, invoke the **Show current repository history** action from the toolbar of the **Git Staging** panel (look for the clock icon) or go to **Window > Show view > Git History**. This opens the **Git History** view at the bottom of the application.

In the toolbar, you can choose which branches will have the history shown. You have the following options:

- **Current branch** - Shows the commits for the current branch, including unpushed local and unpulled remote commits.
- **Current local branch** - Shows the commits only for the current local branch, including unpushed local commits.
- **All branches** - Shows the commits for all branches (both local and remote), including unpushed local and unpulled remote commits.
- **All local branches** - Shows the commits only for local branches, including unpushed local commits.

For each commit in the history table, the following actions are available:
• **Create branch** - Used to create a new branch starting from the selected commit. The new branch is automatically checked out by default. To disable this behavior, deselect the **Checkout branch** option in the **Create branch** dialog box.

**Tip:**
To publish a new branch to the remote repository and start tracking that branch, you need to simply push the local branch using the dedicated action from the **Git Staging** side-view.

• **Create tag** - Used to create a new tag on the selected commit. You have the option to push the tag to the remote repository.

• **Checkout** - Used to checkout a branch at a specific commit (either in detached head form or by creating a new branch at that commit).

• **Revert commit** - Used to create a new commit that reverts all the changes from the selected commit.

• **Reset ”[branch_name]” to this commit** - Used to undo changes by moving the HEAD of the current branch to the selected commit.

The **Git History** view presents all the affected resources for each commit in a list, in the bottom-right area. It includes a text filter field at the top that you can use to conduct searches (e.g. by Date, Author, or Commit ID). An included revision graph helps you to understand how commits connect with one another. For each resource, the following actions are available in the contextual menu:

• **Open** (available for added and modified resources) - This action opens the selected resource.

• **Open previous version** (available for deleted resources) - This action opens the version of the selected resource from before its deletion.

• **Open working copy version** (available for added and modified resources) - This action opens the working copy version of the selected resource. It also works if the resource has been renamed in between versions.

• **Reset file to this commit** (available for added and modified resources) - This action checks out the selected version of the resource, overwriting its current version. It does not work if the resource has been renamed or deleted in between versions.

• **Compare with previous version** (available for modified resources) - This action compares the selected version of the selected resource with the previous one using the **Compare Files tool** (on page 474).

• **Compare with working copy version** (available for modified resources) - This action compares the selected version of the selected resource with the current one using the **Compare Files tool** (on page 474).

• **Compare with each other** (available when selecting 2 versions of a single file) - This action compares the selected versions with each other using the **Compare Files tool** (on page 474).

**Blame**

The contextual menu of each unstaged resource contains a **Show blame** action that opens the selected resource in the main editing area and colors the editor lines with different colors based on the revision
information. Selecting a line in the opened editor will highlight the corresponding entry from the history table in the Git History side-view.

This action is also available in the contextual menu of the current editor and of the Git resources from the Project side-view (on page 402).

Unstaged Files Area

In the Unstaged files area, you will see all the modifications that have occurred in your working copy (files that have been modified, new files, and deleted files) and are not part of the next commit.

- Various actions are available in the contextual menu (Open, Open in compare editor, Stage, Discard, Show history, Show blame, and more).
- You can stage files (i.e. move them to the Staged files area (on page 2523)) using the actions from the toolbar found above the top-right corner of this area. You can choose between staging all the files, by clicking the Stage all button (double arrow icon), and staging specific files, by selecting them and clicking the Stage selected button (single arrow icon).
- You can switch between the list view and the tree view by clicking on the Switch to tree/list view button positioned to the right of the staging buttons.

Staged Files Area

In the Staged files area, you will see all the resources that are ready to be committed. The files from this area can be unstaged and sent back to the Unstaged files area (on page 2523). This area has actions similar to those from the Unstaged files area (on page 2523), with the exception of the Show history and Show blame actions that are not available here.

Comparing Changes and Conflict Resolution

At any time, if you want to see the differences between the last commit and your current modifications, you can double-click a file from either the Unstaged files area (on page 2523) or Staged files area (on page 2523), and the Compare Files (on page 474) window will appear and highlight the changes.

If the file has a conflict (has been modified both by you and another), Oxygen's 3-Way file comparison feature (on page 477) will show a comparison between the local change, the remote change, and the original base revision.

Committing

After staging the files, on the bottom of the view you can provide a commit message and commit them to your local repository. For convenience, you can also select a previously provided message.

In the toolbar above the Commit message text area, there are a few toggle buttons that affect your commit if they are enabled:
• **Amend last commit** - Enabling this option is a convenient way to modify the most recent commit. It lets you combine staged changes with the previous commit instead of creating an entirely new commit. It can also be used to simply edit the previous commit message without changing its snapshot. This action should not be performed on public commits (commits that were pushed to the remote repository).

• **Automatically push changes to remote when committing** - If this option is enabled, when a commit is performed, the committed changes are also pushed to the remote repository.

### Push/Pull (with Merge or Rebase)

To push your local repository changes to the remote repository, use the Push button from the view's toolbar (up arrow).

To bring the changes from the remote repository into your local repository, use one of the Pull actions from the same toolbar (down arrow). You can choose between **Pull (merge)** and **Pull (rebase)**. The invoked action is promoted as the current action of the toolbar button.

---

**Note:**

When pushing a local branch that does not have a corresponding remote branch, a remote branch will automatically be created with the same name as the local branch.

### File Conflict Solving Workflow

After editing a file, committing it to the local repository, and trying to push it to the remote repository, if a warning appears about not being up to date with the repository, follow these steps:

1. Pull the data from the repository using one of the **Pull** actions.
2. In the Unstaged files area (on page 2523), select each conflicted file and resolve the conflicts. You can do this, for example, by opening the conflicted files in the compare editor, either by double-clicking on them or by using the contextual menu action, and then choose what changes you want to keep and discard, and save the document. You can also use the Resolve using Mine, Resolve using Theirs, or Mark as resolved actions from the contextual menu of a resource.
3. If you choose to use the compare editor, after you close it, the file will be staged automatically and moved to the Staged files area (on page 2523).

At this point, the next actions depend on which **Pull** action was chosen:

• **Pull (merge):**
  1. When all the conflicts are resolved and no more files are left in the Unstaged files area (on page 2523), the changes can be committed.
  2. Enter a message and commit. You will now have new changes to push.
  3. Push the changes to the remote repository.
Note:
You can abort the merge by clicking the **Abort merge** button. This will revert the repository to its previous state prior to the pull attempt.

**Pull (rebase):**
1. When all the conflicts are resolved, click the **Continue rebase** button.
2. Push any outgoing changes.

Note:
You can abort the rebase by clicking the **Abort rebase** button. This will revert the repository to its previous state prior to the pull request.

The Project View and the Current Editor

For resources from Git repositories, this add-on also contributes a variety of actions in the contextual menus of the **Project side-view (on page 402)** and the current editor (**Text** and **Author** modes). These actions include: **Show history**, **Show blame**, **Git Diff** (only in the **Project** view), and **Commit** (only in the **Project** view).

Git Branch Manager

To show all the local and remote branches, click the **Show Git Branch Manager** button on the toolbar of the **Git Staging** panel (look for the branches icon) or select **Git Branch Manager** from **Window > Show view**. By default, the **Git Branch Manager** is presented to the right of the editing area.

The **Git Branch Manager** side-view displays all the branches as a tree. The tree can be filtered using the text field at the top of the panel and you can reload the information by using the **Refresh** action. When hovering the cursor over a branch name, a tooltip is displayed that provides information about the last commit performed on that branch (such as the author and the date of the commit).

The following actions are available in the contextual menu for each local branch:

- **Checkout branch** - Checks out the selected branch and switches the local repository to the selected branch.

- **Create branch** - Creates a new branch using the selected branch as the starting point. The new branch is automatically checked out by default. To disable this behavior, deselect the **Checkout branch** option in the **Create branch** dialog box.

Tip:
To publish a new branch to the remote repository and start tracking that branch, you need to simply push the local branch using the dedicated action from the **Git Staging** side-view.
• **Merge "SELECTED_BRANCH" into "CURRENT_BRANCH"** - Brings all the changes from 
  SELECTED_BRANCH into CURRENT_BRANCH.

• **Delete branch** - Deletes the selected branch.

For the remote branches, the **Checkout branch** action checks out the selected branch and creates a local branch from the selected remote branch.

**Preferences**

The **Git Client** add-on contributes a preferences page to Oxygen XML Editor. To access it, go to **Options > Preferences > Plugins > Git Client** or click the **Settings** button from the toolbar of the **Git Staging** view and select **Preferences**. This preferences page includes the following options:

• **When detecting a Git repository inside a newly opened project** - This determines what happens to the current working copy when a project that contains a Git repository is opened in the **Project side-view** *(on page 402)*. You can choose between:
  ◦ **Always switch to the new working copy**
  ◦ **Never switch to the new working copy**
  ◦ **Always ask** (default value)

• **Notify me about new commits in the remote repository** - When this option is selected, Oxygen XML Editor will show notification messages when it detects that new commits have been pushed to the remote repository. By default, this option is not selected.

• **Update all submodules after pulling changes from the remote repository** - If this option is selected, when a repository is updated using the **Pull** operation, all sub-modules are updated as well. This option is selected by default.

• **Global Options/Project Options** - If you select **Project Options**, the settings are stored in the project file (.xpr) that can easily be **shared with other users** *(on page 313)*.

**Editor Variables**

The **Git Client** contributes the following editor variables:

• **${git(working_copy_name)}** - The name of the working copy directory.
• **${git(working_copy_path)}** - The absolute file path of the working copy directory.
• **${git(working_copy_url)}** - The location of the working copy directory as a URL.
• **${git(short_branch_name)}** - The short name of the current branch (e.g. dev).
• **${git(full_branch_name)}** - The full name of the current branch (e.g. refs/heads/dev).

**Resources**

For more information about the Git Client add-on, as well as details regarding other popular add-ons that extend the functionality of Oxygen XML Editor, watch the following webinars/presentations:
Content Fusion Connector Add-on

An **Oxygen Content Fusion Connector** add-on is available that can be installed to integrate **Oxygen Content Fusion** with Oxygen XML Editor. **Content Fusion** is a flexible, intuitive collaboration platform designed to adapt to virtually any type of workflow that a collaborative team may use for their documentation review process.

Quick Installation

You can drag the following **Install** button and drop it into the main editor in **Oxygen** (version 24.1 or newer) to quickly initiate the installation process:

Install

Manual Installation

To manually install the **Oxygen Content Fusion Connector** add-on in **Oxygen XML Editor** or **Oxygen XML Author**, follow these steps:

1. Go to **Window > Show View** and select **Content Fusion Task Manager**.
2. If it has not already been installed, a message will appear asking you if you want to install the add-on.
3. To continue, click **Install** and follow the on-screen instructions.
4. Restart Oxygen XML Editor.

**Result:** **Content Fusion Task Manager** is now available in the **Window > Show View** menu. Selecting this button/action opens the **Content Fusion Task Manager** view.

To fully take advantage of all the benefits and features, your organization will need an **Oxygen Content Fusion Enterprise Server**. This solution allows you to host, setup, and configure your own server and control your data. With this model, you are also able to upload custom frameworks and plugins, and to configure various settings. It is possible to evaluate **Content Fusion** free of charge, for a limited time. For more information, see the **Oxygen Content Fusion website**.

For more information about using **Content Fusion** in Oxygen XML Editor, see the **Oxygen Content Fusion User Guide**.

Integrating Oxygen Feedback with Oxygen XML Editor/Author

It is possible to integrate **Oxygen Feedback** with **Oxygen XML Editor/Author** so that your documentation team can see all the comments added in your WebHelp output. This means they can react to user feedback by making corrections and updating the source content without leaving the application. This is made possible by
an add-on that when installed in Oxygen XML Editor/Author, a Feedback Comments Manager view becomes available.

⚠️ **Attention:**
The Feedback Comments Manager view only functions in Oxygen XML Editor or Oxygen XML Author. It does not work in the Oxygen XML Developer edition.

To see a demonstration of Oxygen Feedback being integrated with Oxygen XML Editor/Author, watch our Webinar: **DITA Publishing and Feedback with Oxygen Tools**.

**Quick Installation**

You can drag the following Install button and drop it into the main editor in Oxygen (version 24.1 or newer) to quickly initiate the installation process:

![Install]

**Manual Installation**

To manually install this add-on in Oxygen XML Editor/Author, follow this procedure:

1. Go to Help > Install new add-ons to open an add-on selection dialog box.
2. Enter or paste https://www.oxygenxml.com/InstData/Addons/default/updateSite.xml in the Show add-ons from field or select it from the drop-down menu.
3. Select Oxygen Feedback Connector add-on and click Next.
4. Select the I accept all terms of the end user license agreement option and click Finish.
5. Restart the application.

**Result:** The Feedback Comments Manager view is now available. By default, it is displayed to the right of the main editing panel. If the view is not displayed, it can be opened by selecting it from the Window > Show View menu. You can also use the Show Feedback Comments Manager from the contextual menu in the DITA Maps Manager.

**Connecting the Feedback Comments Manager View**

If this is your first time using it, you need to configure a mapping between Oxygen XML Editor/Author and your Feedback site configuration(s). If you have already configured a mapping, you just need to connect the Feedback Comments Manager to your Feedback account.

To connect, follow this procedure:

1. Select Connect from the user drop-down menu at the top-right corner of the Feedback Comments Manager.

**Step Result:** You are directed to the administration login page in your default browser.
2. Log in with your credentials, click **Authorize** in the resulting page, and go back to *Oxygen XML Editor/Author*.

3. If this is your first time using the **Feedback Comments Manager**, you need to configure a mapping:
   a. Select **Preferences** from the user drop-down menu at the top-right corner of the **Feedback Comments Manager**.

   **Step Result:** You are directed to the **Oxygen Feedback Connector** preferences page (you can also reach this page by going to **Options > Preferences > Plugins > Oxygen Feedback Connector**).

   b. Click the **New** button at the bottom of the mappings table.

   **Step Result:** This opens the **New Oxygen Feedback Mapping** dialog box where you can configure the mapping.

   c. In the **Context DITA map location** field, click the 📀 browsing button and specify the URL of your context DITA map. This is the path to the root DITA map used to publish your output.

   d. For the **Published site base URL** field, *Oxygen XML Editor/Author* automatically detects all the base URLs for the site configurations that you have the role of **Moderator**, **Admin**, or **Owner**. You can type the base URL or select it from the drop-down menu.

   e. Click **OK** in both dialog boxes.

**Result:** The comments that exist in the published WebHelp output for the site configurations attached to the current user's organization (based on the configured mapping) will be loaded in the **Feedback Comments Manager** view.

**Using the Feedback Comments Manager View**

If the view is not already open in *Oxygen XML Editor/Author*, go to **Window > Show View > Feedback Comments Manager**. You can also use the 📀 **Show Feedback Comments Manager** from the contextual menu in the **DITA Maps Manager**.
The Feedback Comments Manager view contains the following actions and components:

**Home button**

Use this button to open the Oxygen Feedback administration interface in your default browser.

**User Name drop-down menu**

Once connected and authorized, your user name will appear in the upper right corner of the view.

The drop-down menu offers the following actions:

**Refresh**

Forces a refresh to synchronize the information in the view and reloads the first 20 items.

**Preferences**

Opens the Oxygen Feedback Connector preferences page where you can configure a mapping between Oxygen XML Editor/Author and your Feedback site configuration(s).

**Help**

Opens the Oxygen Feedback user guide in your default browser.

**Connect/Disconnect**
Signs you in or out of your **Oxygen Feedback** account.

### Feedback for

You can choose which display mode to use for retrieving comments. The options are:

- **Context DITA map** - If there are base URLs mapped to the current context map, comments from all versions defined by the given base URLs will be loaded.
- **Current editor** - Only comments added in the output for the current topic will be loaded. This mode will retrieve comments from web pages generated from any DITA source file (.dita file extension) as well as any source file with a file extension of `xml`, `html`, or `md`.

### Filters

You can apply filters for **Site**, **Version**, and **Status**:

#### Site

You can choose to show **All** site configurations that you are assigned as **Moderator**, **Admin**, or **Owner** or you can choose a specific site configuration.

#### Version

You can choose show **All** versions for the site configurations that you are assigned as **Moderator**, **Admin**, or **Owner** or you can choose a specific version for the specified site configuration. If the **Site** filter is set to **All**, this drop-down includes the name for all the versions available for the currently detected sites. If there are multiple versions with the same name, the name will be shown only once, but the filtering will detect all the versions with the selected name.

#### Status

You can choose between:

- **Not resolved (Pending, Open, Reopened)** - Displays comments that are in a state of **Pending**, **Open**, or **Reopened**.
- **Pending** - Only displays comments that are in a **Pending** state.
- **Open** - Only displays comments that are in an **Open** state.
- **Reopened** - Only displays comments that are in an **Reopened** state.
- **Resolved** - Only displays comments that are in a **Resolved** state.
- **All** - Displays all comments.

**Note:**

The filters are persistent between sessions. If the comments are retrieved for the **Context DITA map**, they are saved in options associated with the context map. Otherwise, they are associated to a ditamap that is determined from the one that is set in the mapping preferences.
Note:
Comments are only retrieved for site configurations that you have the role of Moderator, Admin, or Owner.

In the comments area, each comment is displayed as its own box and they are grouped by the topic title or path of the web page where the comments were added and then sorted by date (most recent first). If a DITA topic from context map (or the current topic) can be computed from the web page where a comment is added, you can double-click anywhere in the comment box to open the topic in the main editor. You can also click the group name (the topic title or path) to open the particular topic in the WebHelp output, in your default browser.

At the bottom-right corner of the comment box for each first-level comment (a comment that is not a reply), there is a link to Resolve or Reopen the comment. The Resolve action is only available for first-level comments (not replies) and it marks the comment as Resolved. The Reopen action is only available for resolved comments and it marks the comment as Reopened.

Note:
If a new reply is added to a thread that is marked as Resolved, the thread is automatically marked as Reopened.

Each comment box also includes the commenter’s name and avatar, the time the comment was added, the state of the comment (e.g. Open, Rejected, Deleted, Resolved, Reopened), the initials of the site configuration, the version number, a link to open the WebHelp topic in your browser (the most specific URL for a version is used), and the content of the comment (up to 4 lines). If you hover over a status badge for comments whose status is Rejected, Deleted, or Resolved, you can see the user name who changed the status and the time it was changed.

If there are more than 20 top-level comments, the comments are retrieved in chunks of 20 (replies are not limited) and there is a Load more comments button at the bottom that will retrieve the next chunk of top-level comments.

Contextual Menu

If you right-click anywhere in a comment box, the contextual menu offers the following actions:

Open topic

If the Feedback for option is set to Context DITA Map and a parent file (from the current DITA map hierarchy) could be determined, selecting this option opens the parent document in the main editor. If the Open topic action is executed on a block-level comment, the parent document is opened and the element that the block-level comment is attached to is highlighted in Author mode (or the cursor is placed at the beginning of the element in Text mode).

Copy content (Ctrl + C)
Copies the content of the comment to the system clipboard.

**Show comment**

Opens a dialog box with the entire comment displayed. You can copy selected content from the comment to the system clipboard by clicking the **Copy** button at the bottom of the dialog box. If not content is selected, it copies the entire content of the comment.

# Migration/Conversions

## Batch Documents Converter Add-on

Oxygen XML Editor offers an add-on that contributes actions in the following submenus:

- **Batch Documents Converter** submenu located in the **Tools** menu and the contextual menu of resources in the **Project** view.
- **Additional conversions** submenu located in **File > Import/Convert**.
- **Import** submenu located in the **Append child**, **Insert Before**, and **Insert After** submenus from the contextual menu of the **DITA Maps Manager** view.

The first time you invoke any of these actions, Oxygen XML Editor will ask you if you want to install it and offer a wizard to help with the installation process. Once installed, you need to restart Oxygen XML Editor and those same actions will then contain the list of available conversions. Selecting an action from the submenu will open a dialog box where you can configure the options for the corresponding conversion. You can batch convert between the following formats:

- HTML to XHTML
- HTML to DITA
- HTML to DocBook4
- HTML to DocBook5
- Markdown to XHTML
- Markdown to DITA
- Markdown to DocBook4
- Markdown to DocBook5
- Word (`.doc` or `.docx`) to XHTML
- Word (`.doc` or `.docx`) to DITA
- Word (`.doc` or `.docx`) to DocBook4
- Word (`.doc` or `.docx`) to DocBook5
- Excel to DITA
- Confluence to DITA
- DocBook to DITA
- JSON to XML
- XML to JSON
• JSON to YAML
• YAML to JSON

When actions are invoked from the contextual menu of the DITA Maps Manager view, the resulting documents from the conversion are automatically inserted in the map as follows:

• Actions from Append child inserts map nodes as children of the currently selected node.
• Actions from Insert Before inserts map nodes as siblings of the currently selected node, above the current node in the map.
• Actions from Insert After inserts map nodes as siblings of the currently selected node, below the current node in the map.

Quick Installation

You can drag the following Install button and drop it into the main editor in Oxygen (version 24.1 or newer) to quickly initiate the installation process:

Install

Manual Installation

To manually install the Batch Documents Converter add-on:

1. Go to Help > Install new add-ons to open an add-on selection dialog box.
2. Enter or paste https://www.oxygenxml.com/InstData/Addons/default/updateSite.xml in the Show add-ons from field or select it from the drop-down menu.
3. Select the Batch Documents Converter add-on and click Next.
4. Read the end-user license agreement. Then select the I accept all terms of the end-user license agreement option and click Finish.
5. Restart the application.

Result: A Batch Documents Converter submenu will now be available in the Tools menu and in the contextual menu. This submenu will contain a list of the various types of available conversions. Selecting one of the types of conversions will open a dialog box where you can configure options for the conversion.

The add-on can also be installed using the following alternative installation procedure:

2. Unzip it inside {oXygenInstallDir}/plugins. Make sure you don't create any intermediate folders. After unzipping the archive, the file system should look like this: {oXygenInstallDir}/plugins/oxygen-batch-converter-addon-x.y.z, and inside this folder, there should be a plugin.xml file.
Configuration

Options for configuring the conversions can be found in the preferences page of the add-on (Options > Preferences > Plugins > Batch Documents Converter) or in the conversion dialog box.

Conversions from Word (Word Styles Mapping)

The conversions from MS Word work best if you only use the MS Word styles to semantically mark up your document. It is important that sections from the Word document are well defined using the heading styles.

Use the Word styles mapping option from the Batch Documents Converter preferences page to configure any of the types of Word conversions (Word to HTML, Word to DITA, Word to DocBook4, and Word to DocBook5) by setting a mapping between Word elements and styles to the corresponding HTML element.

If the Word document contains paragraphs formatted with custom styles, they have to be set in the Word styles mapping configuration. Those that are not set will be converted into simple paragraphs. To add custom styles, you can use the default configuration from the table. For example, if you use a custom Word style named Document Title, you can map this to the HTML "h1" element:

<table>
<thead>
<tr>
<th>Word element</th>
<th>Word style</th>
<th>HTML elements</th>
</tr>
</thead>
<tbody>
<tr>
<td>p</td>
<td>Document Title</td>
<td>h1:fresh</td>
</tr>
</tbody>
</table>

The resulting 'h1' element will be transformed into the corresponding element when converting to DITA, DocBook 4, and DocBook 5.

The Word styles mapping table contains the following columns:

Word element

This column allows one of the following Word elements:

- p - Word paragraph
- r - Word run
- b - bold text
- i - italicized text
- u - underlined text
- strike - strikethrough text
- table - table
- p:unordered-list(x) - unordered list (where 'x' is the nesting level of the list)
- p:ordered-list(x) - ordered list (where 'x' is the nesting level of the list)
This column can be used to map a paragraph, run, or table with a specific style (referenced by name).

Styles can also be referenced by style ID. This is the ID used internally in the .docx file. To map a paragraph or run with a specific style ID, append a dot followed by the style ID in the Word element column (for example: p.Heading1).

**HTML elements**

This column can be used to map the resulting HTML elements. It allows a single element or multiple nested elements.

The nested elements can be declared by using the ‘>’ character (for example: ul > li).

The class attribute can be specified on the resulting HTML elements by appending a dot followed by the class value, after the element (for example: p.myClass).

When converting Word to DITA, these class attributes are automatically converted to outputclass attributes. This may be useful if you want to apply extra processing on the resulting DITA document using a custom XLS stylesheet.

The :fresh syntax can be used to create new elements. If it is not used, the converter will try to reuse the element and close it only when it is necessary.

For example, if the following configuration is set:

<table>
<thead>
<tr>
<th>p</th>
<th>Heading 1</th>
<th>h1</th>
</tr>
</thead>
</table>

When the converter finds consecutive Word paragraphs with the style name Heading 1, these will be converted into a single h1 element that contains the text appended from all of the Word paragraphs.

If h1:fresh is set in the last column, the converter will create separate h1 elements.

To ignore elements, the ‘!’ character can be added in the HTML elements column.

The Export button can be used to export the word styles configuration to an XML file. This exported file can be used to configure the MS Word Dynamic Conversion (on page 3125) from Oxygen XML Editor by copying the file in the DITA-OT plugin directory: [OXYGEN_INSTALL_DIR]/frameworks/dita/DITA-OT3.x/plugins/com.oxygenxml.dynamic.resources.converter.

The Import button allows you to import the word styles configuration from an exported XML file.

**Note:**

The Word styles mapping configuration is applied only for the newer version of MS Word files formatted in the Microsoft Office Open XML (DOCX) format.
Maximum Heading Level for Creating Topics

The Maximum heading level for creating topics option from the Batch Documents Converter preferences page allows you to set a maximum heading level that the converter will process as DITA topics. The headings with a higher level will be converted to section elements.

When the output is a DITA topic, this option sets the maximum heading level that will be converted as a nested topic in the document.

When the output is a DITA map, this option sets the maximum heading level that will be extracted as a DITA topic file and referenced in the DITA map hierarchy.

Note:
This option only applies to the HTML to DITA and Word to DITA conversions.

Word to DITA

The Create DITA maps from Word documents containing multiple headings option from the conversion dialog box allows you to decide whether the output will be a DITA map or a DITA topic. When this option is selected, the sections from your Word document marked by titles or headings will be separated into individual DITA topics and referenced in a DITA map. If the word document does not contain multiple sections, the output will be a single topic. When this option is not selected, the output will be a topic with nested topics and sections according to the number of titles and headings from the Word document.

Markdown to DITA

The Create DITA maps from Markdown documents containing multiple headings option from the conversion dialog box allows you to decide whether the output will be a DITA map or a DITA topic. When this option is selected, all headings from your Markdown document will be separated into individual DITA topics and referenced in a DITA map. If the Markdown document does not contain multiple headings, the output will be a single topic. When this option is not selected, the output will be a topic with nested topics or sections according to the number of headings from the document.

The Create short description elements option from the conversion dialog box allows you to decide whether or not the shortdesc elements are created in the output DITA document. When this option is selected, the first paragraph before the headings from the Markdown document will be converted into DITA short description elements. When this option is not selected, the output will not contain the short description element.

HTML to DITA

The Create DITA maps from HTML documents containing multiple headings option from the conversion dialog box allows you to decide whether the output will be a DITA map or a DITA topic. When this option is selected, the headings from your HTML document will be separated into individual DITA topics and referenced in a DITA map. If the HTML document does not
contain multiple sections, the output will be a single topic. When this option is not selected, the output will be a topic with nested topics or sections according to the number of headings from the document.

**Confluence to DITA**

The Confluence to DITA conversion processes the HTML content generated by the Atlassian® Confluence (see https://www.atlassian.com/software/confluence) export process. To export Confluence content to HTML, log in to your Atlassian® Confluence account and navigate to the specific space that you want to export. Then go to Space Settings > Export space and choose to export it as HTML. The resulting index.html file must be provided in the Input files list from the conversion dialog box.

**DocBook to DITA**

The Create DITA maps from DocBook documents containing multiple sections option from the conversion dialog box allows you to decide whether the output will be a DITA map or a DITA topic. When this option is selected, the sections from your DocBook document will be converted into individual DITA topics and referenced in a DITA map. When this option is not selected, the output will be a single topic with nested topics.

**Resources**

For more information about the Batch Converter add-on, as well as details regarding other popular add-ons that extend the functionality of Oxygen XML Editor, watch the following webinars/presentations:

- Webinar: Working with DITA in Oxygen - Migrating to DITA and Refactoring
- Webinar: Extending the Functionality of Oxygen Using Add-ons

**DITA Editing**

**DITA Prolog Updater Add-on**

Oxygen XML Editor offers an add-on that contributes a preferences page (Options > Preferences > Plugins > DITA Prolog Updater) that includes various options for updating the prolog section of a DITA topic or map.

**Quick Installation**

You can drag the following Install button and drop it into the main editor in Oxygen (version 24.1 or newer) to quickly initiate the installation process:

![Install]

**Manual Installation**

To manually install the add-on, follow these instructions:
1. Go to Help > Install new add-ons to open an add-on selection dialog box.
2. Enter or paste https://www.oxygenxml.com/InstData/Addons/default/updateSite.xml in the Show add-ons from field or select it from the drop-down menu.
3. Select the DITA Prolog Updater add-on and click Next.
4. Read the end-user license agreement. Then select the I accept all terms of the end-user license agreement option and click Finish.
5. Restart the application.

Result: The DITA Prolog Updater preferences page will now be available in Options > Preferences > Plugins.

DITA Prolog Updater Preferences Page

The contributed preferences page (Options > Preferences > Plugins) includes the following general options:

**Author**

Specifies the name of the author. By default, it is the system user name.

**Date format**

Specifies the format of the date that will be added in the prolog section. If the date format entered is invalid, the yyyy/MM/dd format is used by default.

**Limit the number of revised dates to**

Specifies the number of revisions that will be kept. Anytime a <revised> element is added in the prolog section and the specified limit has been reached, the oldest <revised> element is deleted.

**Custom type attribute value for the original author**

Specifies a custom value for the type attribute of the <author> element that is inserted to emphasize the primary or original author of the content. When is not set, the creator value is used.

**Custom type attribute value for an additional author**

Specifies a custom value for the type attribute of the <author> element that is inserted to emphasize an additional author who is not the primary/original author. When is not set, the contributor value is used.

These options are followed by the following options that can be set for DITA topics or maps (or both):

**Enable automatic prolog update on save**

When this option is selected, the prolog is updated anytime the document is saved.

**Set original author**

When this option is selected, the primary or original author is set in the prolog when the document is saved. This option is only applicable for new documents.

**Set created date**

When this option is selected, a created date is added to the prolog when the document is saved. This option is only applicable for new documents.
Update additional authors

When this option is selected, an additional author is set in the prolog when the document is saved. This option is only applicable for already existing documents.

Update revised dates

When this option is selected, a revised date is added to the prolog when the document is saved. This option is only applicable for already existing documents.

For more information, see the details for this DITA Prolog Updater add-on in GitHub.

DITA References View Add-on

Oxygen XML Editor offers an add-on that contributes a DITA References side view that shows all outgoing and incoming references for the current DITA topic. Once the add-on is installed, the view is available in both Text and Author modes.

Quick Installation

You can drag the following Install button and drop it into the main editor in Oxygen (version 24.1 or newer) to quickly initiate the installation process:

Install

Manual Installation

To manually install the add-on, follow these instructions:

1. Go to Help > Install new add-ons to open an add-on selection dialog box.
2. Enter or paste https://www.oxygenxml.com/InstData/Addons/default/updateSite.xml in the Show add-ons from field or select it from the drop-down menu.
3. Select the DITA References View add-on and click Next.
4. Read the end-user license agreement. Then select the I accept all terms of the end-user license agreement option and click Finish.
5. Restart the application.

Result: Once installed, you need to restart Oxygen XML Editor and the DITA References view becomes available. To open the view, select DITA References in the Window > Show View menu.

Outgoing References

The types of outgoing references that are presented in this view include:

- Image references (referenced with an @href or @keyref attribute)
- References to other media resources (DITA objects pointing to video, audio, or embeddable frames)
- Cross references (referenced in an @xref element with an @href or @keyref attribute)
- Key references (referenced with a @keyref attribute)
- Content references (referenced with a @conref or @conkeyref attribute)
• Related links (referenced with an @href or @keyref attribute)
• Related links defined in relationship tables

This side-view is synchronized with the main editor to make it easy to locate a reference in the current document. It also includes contextual menu actions for opening the target of an outgoing reference or showing its definition location. You can also double-click a reference to open its target.

Incoming References

The Incoming references tab shows all references to the current opened DITA topic.

• The references are presented by taking both the Main Files support and the current context map configured in the DITA Maps Manager view into account.
• The references are grouped into three categories: Map, Cross, and Content.
• To open the target of a reference, you can double-click the references, or press Enter with the cursor located on the reference, or right-click the reference and choose Open reference.
• To copy the location of the target of the reference, select the reference and press Ctrl + C (Cmd + C on Mac).
• If a referenced topic is expanded, references to that topic are also displayed.
• While editing a document, if changes appear that might influence how the references were computed, the Refresh button can be used to re-compute the references graph and show an updated list of incoming references.

For more information, see the details for the DITA Outgoing References View add-on in GitHub.

Resources

For more information about the DITA References add-on, as well as details regarding other popular add-ons that extend the functionality of Oxygen XML Editor, watch the following webinars/presentations:

• Webinar: Add-ons You Can Use for Technical Writing
• Webinar: Extending the Functionality of Oxygen Using Add-ons

Terminology

Terminology Checker Add-on

Oxygen XML Editor offers an add-on that provides support for checking terminology. Once the add-on is installed, you can create a terminology file with a set of rules for each term (or sequence of characters) you want flagged. After referencing the custom file, Oxygen XML Editor will automatically highlight matched terms in the Author visual editing mode and offer some contextual menu actions.

Tip:
The terminology checker works for any document opened in the Author visual editing mode, including XML file types, as well as JSON and HTML5 document types.
Quick Installation

You can drag the following Install button and drop it into the main editor in Oxygen (version 24.1 or newer) to quickly initiate the installation process:

Install

Manual Installation

To manually install the add-on, follow this procedure:

1. Go to Help > Install new add-ons to open an add-on selection dialog box.
2. Enter or paste https://www.oxygenxml.com/InstData/Addons/default/updateSite.xml in the Show add-ons from field or select it from the drop-down menu.
3. Select the Terminology Checker add-on and click Next.
4. Read the end-user license agreement. Then select the I accept all terms of the end-user license agreement option and click Finish.
5. Restart the application.

Creating Custom Rules for the Terminology Checker

To create your own custom rules for the terminology checker, follow this procedure:

1. Create a terminology file. There is a template available to help you get started in the New Document wizard (on page 368). Click the New button on the toolbar or select File > New and search for the Terminology File template. Here is an example of the structure for this type of file:

   ```xml
   <incorrect-terms lang="en">
     <incorrect-term ignorecase="true">
       <match>Oxygen</match>
       <suggestion></suggestion>
       <message>Product name should be inside a tag.</message>
     </incorrect-term>
   </incorrect-terms>
   
   Notes:
   ◦ On the incorrect-terms root element, you can specify the @lang attribute. When set, the terms defined in this terminology file will be applied when the closest @xml:lang attribute of the checked node matches the value specified. Not setting this attribute means that the incorrect terms will be applied for all nodes.
   ◦ On the incorrect-terms root element, you can specify the @phase attribute. The value of this attribute is inherited to incorrect-term children nodes. Not setting this attribute means the default phase is used.

   The allowed values are:
• **always** - Incorrect terms are always presented (default value).
• **editing** - Incorrect terms are shown when the document is open in the **Author** mode.
• **validation** - Incorrect terms are shown when the document is checked from a validation scenario.

For example, set this attribute if you want to apply the most important rules when validating with the *Validate and Check for Completeness* action, while still keeping them applied in the editing window.

- The `<incorrect-term>` element must include a `<match>` element.
- On the `<match>` element, you can specify the `@type` attribute with the values `character`, `whole-words`, or `regular-expression`. The default value is `whole-word`, except when the matched term contains Japanese, Chinese, or Korean characters because Asian languages do not usually use spaces to separate words. Example:

```xml
<incorrect-term>
  <match type="character">ing</match>
  <message>Progressive tense should not be allowed</message>
</incorrect-term>
```

- Each `<incorrect-term>` element has a `@severity` attribute that can be set to one of the values `info`, `warning`, or `error`. Example:

```xml
<incorrect-term severity="error">
  <match>he</match>
  <message>Pronouns should be avoided.</message>
</incorrect-term>
```

- The `<suggestions>` element can be left blank or there can be one or more of them inside the `<incorrect-term>` element. It supports regular expressions grouping.
- If you want to replace the match with an XML fragment, you can set the `@format` attribute on the `<suggestion>` element with the value `xml`. For example:

```xml
<incorrect-term ignorecase="true">
  <match type="whole-word">Oxygen XML Editor</match>
  <suggestion format="xml">&lt;ph keyref="oxygen"/&gt;</suggestion>
  <message>Replace all occurrences of product with key reference.</message>
</incorrect-term>
```

- The `<message>` element is optional. If present, its content is displayed in a tooltip when you hover over a highlight. It supports regular expressions grouping.
- The `ignorecase` attribute specifies whether or not the match is case-sensitive.
- The `<link>` element is optional. If present, it provides the source for this rule. Example:

```xml
<incorrect-term ignorecase="true">
  <match type="whole-word">Oxygen XML Editor</match>
  <suggestion format="xml">&lt;ph keyref="oxygen"/&gt;</suggestion>
</incorrect-term>
```
2. Go to Options > Preferences > Plugins > Oxygen Terminology Checker and in the Project Terminology Folder field, specify the folder where the default terminology file is located. Note that the default location is displayed in the Project Terminology Folder field regardless of whether or not the file exists on disk. You can also specify an Additional Terminology folder.

3. Click OK several times to apply the changes and close the preferences dialog box.

**Result:** If any of the terms (or sequence of characters) that are defined in the terminology file are detected in any open file, Oxygen XML Editor highlights the matches in the Author visual editing mode.

**Note:** If you have a folder named oxygen-term-checker in the current project that is open in the Project view, all the files in that folder will also be loaded by the terminology checker.

**Using Vale Rules with the Terminology Checker**

The Terminology Checker has partial support for applying custom Vale rules.

Supported Vale scopes: heading, table.header, table.cell, list, paragraph, code, strong, emphasis.

Supported Vale extension points: Existence, Substitution, Occurrence, Repetition, Conditional.

**Result:** If any of the terms (or sequence of characters) that are defined in the terminology file are detected in any open file, Oxygen XML Editor highlights the matches in the Author visual editing mode.

**Note:** If you have a folder named oxygen-term-checker in the current project that is open in the Project view, all the files in that folder will also be loaded by the terminology checker. As an example, the Oxygen XML Editor user guide has a folder with some of the Microsoft style guide rules: https://github.com/oxygenxml/userguide/tree/master/DITA/oxygen-term-checker. Once the user guide project is open in the Oxygen XML Editor Project view, the add-on will start using those rules to check the content.

**Resources:** You can find already created Vale rules that implement various checks on the following websites:

- Vale rules that aim to automate the Microsoft style guide: https://github.com/errata-ai/Microsoft/tree/master/Microsoft.
Working with the Terminology Checker

The Terminology Helper side view shows all problems found in the document. You can right-click each problem to apply possible fixes or to find out more details about it. The tooltip for each problem displays a custom message and more information (e.g. for Vale rules, it also displays the name of the Vale rule file that defines the rule). You can filter problems based on their severity, match, and message and the toolbar has actions to navigate between problems or to open the Terminology Checker preferences page.

You can also right-click problems highlighted in the Author visual editing mode to access the following contextual menu actions:

- **Replace with "..."** - Replaces the currently highlighted match with the content inside the `<suggestion>` element.
- **Replace all with "..."** - Replaces all instances of the highlighted match found in the current document with the content inside the `<suggestion>` element.
- **Correct all matching highlights** - Replaces all highlighted matches (all matched terms) within the document with the content inside the first `<suggestion>` element from the terminology file.

The terminology checking can be disabled by clicking the Show/Hide Terminology Highlights toolbar button.

### Other Notes:

- The checker automatically skips deleted content with tracked changes and space-preserved elements (e.g. codeblocks).
- When replacements are performed, the capitalization is preserved.
- In the Oxygen XML Editor Options > Preferences > Plugins > Oxygen Terminology Checker page, you can define the highlight colors to be used for each issue depending on its severity. You can also reference a folder that contains the terminology rules. This folder can contain other folders with terminology files or just the terminology files. The option that controls automatic capitalization can also be found in this preferences page.
- If you select Project Options (in the Oxygen Terminology Checker preferences page), the settings are stored in the project file (.xpr) that can easily be shared with other users (on page 313).

### Resources

For more information about the Terminology Checker add-on, as well as details regarding other popular add-ons that extend the functionality of Oxygen XML Editor, watch the following webinar:

- **Webinar: Add-ons You Can Use for Technical Writing**
Vale Linter for Markdown and HTML Validation Add-on

The Vale Validation add-on runs the Vale linter over the currently edited file and presents the validation errors in the results area at the bottom of the application. A Linter is a tool that automatically verifies specific rules against your code or documentation. This is useful for enforcing a style guide or for catching commonly mistaken branding issues.

Quick Installation

You can drag the following Install button and drop it into the main editor in Oxygen (version 24.1 or newer) to quickly initiate the installation process:

Install

Manual Installation

To manually install this add-on, follow this procedure:

1. Go to Help > Install new add-ons to open an add-on selection dialog box.
2. Enter or paste https://www.oxygenxml.com/InstData/Addons/default/updateSite.xml in the Show add-ons from field or select it from the drop-down menu.
3. Select the Vale Linter for Markdown and HTML Validation add-on and click Next.
4. Read the end-user license agreement. Then select the I accept all terms of the end-user license agreement option and click Finish.
5. Restart the application.

Setup the Oxygen Vale Validation

To set up the Vale validation, follow this procedure:

1. Download and unzip the proper Vale executable for your OS. On Linux and macOS, you must give executable permission to the Vale executable. You can do this by opening a console in the Vale directory and running:
   
   ```bash
   chmod u+x vale
   ```

2. Go to Options > Preferences > Plugins > Oxygen Vale Validation and specify the path to the previously downloaded Vale executable.
3. In the same preferences page, you can also specify a path to a Vale configuration file (.vale.ini). Vale automatically detects this file by looking 6 levels up in the current file's ancestor directories, but you can also impose one.

Vale Styles

Vale uses collections of individual YAML files (or "rules") to enforce particular writing constructs. These collections are referred to as styles and are organized in a nested folder structure at a user-specified location. The .vale.ini file is where you will control most of Vale's behavior, including which files to lint and how to
lint them. Vale automatically detects `.vale.ini`, but you can also specify the path to `.vale.ini` from the plugin's preferences page (Options > Preferences > Plugins > Oxygen Vale Validation).

**Third-party Styles**

Vale has a growing selection of pre-made styles available for download from its style library.

**Validation**

After setting up the Vale executable, creating or downloading Vale styles, and specifying the path to `.vale.ini`, the add-on will intercept the Automatic Validation (on page 775) and Manual Validation (on page 775) and contribute errors and warning obtained by running Vale validation over the current file. The errors and warnings are highlighted in the editor.

**Note:**

Although Vale supports multiple file formats, the Vale Validation add-on currently only supports Markdown (*.md files) and HTML files.

**Translation**

**Translator Helper Add-on**

Oxygen XML Editor offers an add-on that provides support for helping translators. Once the add-on is installed, you can use Google Translate to create a preliminary translation for your XML content that can later be manually improved. The add-on is also useful for showing potential customers how a DITA project will look like after it is translated into their native language. After the add-on is installed, a Translator Helper view is available from the Window > Show View menu.

**Quick Installation**

You can drag the following Install button and drop it into the main editor in Oxygen (version 24.1 or newer) to quickly initiate the installation process:

[Install]

**Manual Installation**

To manually install the add-on, follow these instructions:

1. Go to Help > Install new add-ons to open an add-on selection dialog box.
2. Enter or paste `http://www.oxygenxml.com/InstData/Addons/default/updateSite.xml` in the Show add-ons from field.
3. Select the Translator Helper add-on and click Next.
4. Read the end-user license agreement. Then select the I accept all terms of the end-user license agreement option and click Finish.
5. Restart Oxygen XML Editor.
**Result:** The Translator Helper view now be available and can be selected from the Window > Show View menu.

**Features:**

- You can translate content to various languages using Google Translate or DeepL Translator.
  1. In Oxygen XML Editor, select the content you want to translate in Author mode. You can also select the entire content of the document.
  2. Right-click, select Translate, and choose the target translator and language.
  3. Copy the translated content from the resulting web page (the easiest way to do this is to use the Copy translation button for Google Translate or the Copy to clipboard button for DeepL Translator).
  4. Go back to Oxygen XML Editor. If your original selection included multiple inline elements (or entire block elements or the entire document), a dialog box will be displayed. Click the Replace button to replace the selection with the copied translated content. If you leave the Show original content side-by-side option selected, the original untranslated content will be displayed in the Translator Helper side view.
- You can choose to display your original content in a Translator Helper side view so that you can see the original untranslated content side-by-side with the content that you are editing/ translating. To open this side view, right-click anywhere in the Author mode editor pane and select the Original Content Side By Side action (or select Window > Show View > Translator Helper).

**Resources**

For more information about the Translator Helper add-on, as well as details regarding other popular add-ons that extend the functionality of Oxygen XML Editor, watch the following webinar:

- Webinar: Add-ons You Can Use for Technical Writing

**DITA Translation Package Builder Add-on**

Oxygen XML Editor offers an add-on that contributes contextual menu actions that help you build a translation package for DITA files that can be sent to translators. You can also extract the changed files back into your project once you receive the package back from the translators.

**Quick Installation**

You can drag the following Install button and drop it into the main editor in Oxygen (version 24.1 or newer) to quickly initiate the installation process:

**Manual Installation**

To manually install it the add-on, follow these instructions:
1. Go to Help > Install new add-ons to open an add-on selection dialog box.
2. Enter or paste https://www.oxygenxml.com/InstData/Addons/default/updateSite.xml in the Show add-ons from field or select it from the drop-down menu.
3. Select the DITA Translation Package Builder add-on and click Next.
4. Read the end-user license agreement. Then select the I accept all terms of the end-user license agreement option and click Finish.
5. Restart the application.

Result: A Translation Package Builder submenu will now be available in the contextual menu of the DITA Maps Manager. This submenu includes actions to generate a ZIP package of modified files that can be sent to translators, as well as an action to extract translated files back into your DITA project.

For more information, see the details for this Translation Package Builder add-on in GitHub.

Using the Translation Package Builder

Once the add-on is installed, a sub-menu named Translation Package Builder is available in contextual menu of the DITA Maps Manager with the following actions:

- **Generate Milestone** - Use this action first. It generates a unique hash for each documentation resource. This information is used by the second action (Create Modified Files Package) to detect which files have been modified. A milestone file should be generated after you install this add-on and then again after each package is sent to translators.

- **Create Modified Files Package** - This action detects which files have been changed since the last generated milestone. These files are packed inside a ZIP file that can be sent to translators. After doing this, you can also generate a new milestone so that the next package will only contain new changes.

- **Apply Package** - When the translated files arrive from the translator, you should open the DITA map that corresponds to the received language (e.g. open dita-map-french.ditamap if the package contains the french translation). Invoking this action will extract the changed files inside the map's directory.

Resources

For more information about the Translation Package Builder add-on, as well as details regarding other popular add-ons that extend the functionality of Oxygen XML Editor, watch the following webinars/presentation:

- Webinar: Add-ons You Can Use for Technical Writing
- Webinar: Automate XML processing with Oxygen XML Scripting
Productivity

Smart Autocomplete Add-on (Experimental)

Oxygen XML Editor offers a Smart Autocomplete add-on that provides support for helping writers create content by providing text completions. This can be done by in two ways: by using a built-in algorithm, or by using a language model (provided by Open AI®).

Quick Installation

You can drag the following Install button and drop it into the main editor in Oxygen (version 24.1 or newer) to quickly initiate the installation process:

Install

Manual Installation

To manually install this add-on, follow this procedure:

1. Go to Help > Install new add-ons to open an add-on selection dialog box. Enter or paste https://www.oxygenxml.com/InstData/Addons/default/updateSite.xml in the Show add-ons from field or select it from the drop-down menu.
2. Select the Smart Autocomplete add-on and click Next.
3. Read the end-user license agreement. Then select the I accept all terms of the end-user license agreement option and click Finish.
4. Restart the application.

Result: The Smart Autocomplete side view should be now available.

Use Cases for the Built-in Algorithm

1. Completions with existing project text.
2. You can organize text in some of your project files as a kind of completion proposals library.

Use Cases for the Language Models

Technical content:

1. Completions (note that it is very creative, but not always accurate, so you need to check the proposed content)
2. Automatic markup
3. Keywords extraction
4. Summarizing
5. Translation

Non-technical content:

1. Open AI represents a trademark of the Open AI LLC https://openai.com/api/policies/terms/
1. Brainstorming ideas
2. Paraphrasing
3. Copywriting
4. Creating commercial ads.

Resources

For more information about the Smart Autocomplete add-on, watch our video demonstration:

https://www.youtube.com/embed/-CGjBvh7-Pc

UI Overview

The core functionality of the Smart Autocomplete add-on is to offer text continuation after you press a space after a word. There are three radio buttons that allow you to select the completion engine:

- **Built-in** - Uses the built-in algorithm *(on page 2551).*
- **Open AI** - Uses the Open AI models *(on page 2552).*
- **None** - Disables the functionality.

If you select the **Auto-insert best proposals** checkbox, the insertion of the text is done automatically. Deselecting this option allows you to choose from a list of proposals displayed in the view.

Using the Built-in Algorithm

The built-in algorithm analyzes the set of all XML, HTML, or Markdown files from your project and creates a local index. Once this is selected, you can initiate the indexing of project files by clicking the **Reindex Project** button. The same effect is obtained if you reindex the project files from the **Open/Find Resources** view. When you save one of the indexed files, this will be re-indexed incrementally.

Advantages:

- It is fast and works on your local computer.
- It generates content that you already have in the project.
- You can use it as a kind of text template library. The quality of the completions increase with the size of your text content in the project.
Disadvantages:

- The proposals are not creative.
- There is no semantic search.

Using the Open AI Models

*Open AI GPT3* is the largest commercial AI model available to integrators. It contains about 175 billions weights, and it was trained on a huge corpus of text. Similar to other encoder/decoder language models, the training was done by 'hiding' some words from the input and asking the model to guess the missing word.

**GPT 3**

![Image](image.png)

Starting with huge amount of text...  
...by masking and guessing missing text  

Huge models like:

- OpenAI

...get extraordinary skills:

- Understands concepts
- Capable of advanced tasks

The resulting language model was able to find semantic connections between words, in some way 'understanding' the meaning of a window of text, while being capable of doing much more than it was trained to do initially (e.g. responding to questions).

Advantages of the Open AI GPT-3 Language Model:

- Semantic awareness
- Extensible
- 'Creative'

Disadvantages:

- Slow due to network latency since content is sent to Open AI® servers.

**Important:**

Check the license terms of the Open AI platform to make sure your company agrees with sending parts of the edited document to the Open AI.
• Not always factually correct since it uses probabilistic responses.

⚠️ Important:
Always check the generated content. It behaves more like a brain than a database.

To use the GTP-3 language models, choose Open AI for the Engine. The interface then displays the following areas:

• The list of proposals.
• A panel with actions that can transform the selected text from the editor.
• A control panel with three tabs:
  ◦ **Model** - This drop-down allows you to select the Open AI model you intend to use for providing completions. It can be one of the base models, or one you fine-tuned (or trained) from your data. The default selection is *text-davinci-001*.

⚠️ Note:
The base models are named according to the following conventions (they contain one of the following keywords): *davinci* (signals the most capable model), *curie*, *babbage*, and *ada*.

  ◦ **Fine-tune Jobs** - From this tab, you can start fine-tuned jobs. Once a fine-tuned job has finished, the resulting model is listed in the **Model** combo box.
  ◦ **API Key** - This tab allows you to configure the Open AI API Key.

### Installing the Open AI API Key

To connect to the Open AI platform, follow these instructions to obtain an API Key:

2. Click on your organization avatar, then choose **View API Keys**.
3. Click + **Create a new secret key**.
4. Copy the key to the clipboard.
5. Paste the key into the **API Key** text field in the **Smart Autocomplete** view.

You can check that the add-on is connected to Open AI by switching to the **Model** tab. If the combo box contains a list of model names, the connection was successful. The default selected model is *text-davinci-001*.

### Getting Completions

To get completions while typing content, enter a space and wait for a few seconds. Typing more text cancels the completion request. You can try different models to see different results. You can also try to fine tune a model to get better completions.
Creating a Fine-Tuned Model

In the Fine-tune Jobs tab, you can create new models by using a process called fine tuning.

1. Click the New Fine-tune Job button at the bottom of the view. A dialog box is displayed.
2. In the resulting dialog box, you can choose to create a model starting from a base one (see the Base model combo box) that is trained over the text of XML, TXT, Markdown, or PDF files from the current project, from a specified folder, or from a file that contains a lot of "prompt"/"completion" examples in JSONL format. This file may be created by a tool (or manually) from the training results in better models.
3. Click Start Fine tune job to start the job and it will be listed in the Fine-tune Jobs panel.
4. To check the job progress, you can click the Refresh button (the Status column will be updated).
   Another way is to use the Job Events action from the contextual menu. Example of a list of events:

   Fri Feb 25 08:45:22 EET 2022 info Created fine-tune: ft-jAeo6XJ3dyXF0mchWAD0wWFb
   Fri Feb 25 08:45:30 EET 2022 info Fine-tune costs $11.85
   Fri Feb 25 08:45:30 EET 2022 info Fine-tune enqueued. Queue number: 0
   Fri Feb 25 08:45:34 EET 2022 info Fine-tune started
   Fri Feb 25 08:53:34 EET 2022 info Completed epoch 1/4
   Fri Feb 25 09:00:53 EET 2022 info Completed epoch 2/4
   Fri Feb 25 09:08:10 EET 2022 info Completed epoch 3/4
   Fri Feb 25 09:15:30 EET 2022 info Completed epoch 4/4
   Fri Feb 25 09:15:56 EET 2022 info Uploaded model:
   Fri Feb 25 09:16:00 EET 2022 info Uploaded result file: file-Avrc2wpFB1LKFiR7PFKver4o
   Fri Feb 25 09:16:00 EET 2022 info Fine-tune succeeded

To cancel a job, use Cancel action from the contextual menu.
5. Once a job is finished, you can click Refresh in the Model tab and you will find the name of the new model in the combo box.

Using XSLT to Create an Open AI Training Set

Suppose you need to fine tune a model to learn to apply markup to your text.

To train or fine tune a base model, you need to create a data set in JSONL format. An Open AI JSONL file contains prompt - completion pairs, each pair on a single line.

```json
{"prompt": "<prompt text>", "completion": "<sample of generated text>"}
```

For more details about this format, see: https://beta.openai.com/docs/guides/fine-tuning/prepare-training-data.

In the following example, an XSLT stylesheet is used to generate the JSONL file. Prompt-completion pairs are created by putting the plain text together with the marked text from each of these elements.

prompt
First, some constants are defined and then the root matching template that scans a folder that contains DITA topics is created:

```xml
<!-- Method "text" because we generate JSON -->
<xsl:output method="text" omit-xml-declaration="yes"/>

<!-- Paragraphs that are smaller than this limit are processed. -->
<xsl:variable name="limit" select="500"/>

<!-- Open AI recommends adding terminators to the prompt and completion -->
<xsl:variable name="prompt-terminator" select="' ### '"/>
<xsl:variable name="completion-terminator" select="'$$$'"/>

<xsl:template match="/">
    <xsl:for-each select="collection('file:/C:/path-to-topics-folder')">
        <xsl:call-template name="process-file"/>
    </xsl:for-each>
</xsl:template>

Next, the template that processes each of the topic files is defined. Only the `<p>` and `<li>` elements are selected in this example, and for each of them, a prompt-completion pair is generated.

```xml
<xsl:template name="process-file">
    <xsl:for-each select=".//*[contains(@class, ' topic/p ') or contains(@class, ' topic/li ')]">
        <xsl:variable name="text" select="oxy:clean(.)"/>
        <xsl:variable name="text-with-markup"
            select="oxy:clean($text-with-markup)"
        />
        <xsl:if test="string-length($text)&lt;$limit and
            string-length($text-with-markup)&lt;$limit and
            string-length($text)&gt;0 and
            string-length($text-with-markup)&gt;0 and
            string-length($text) != string-length($text-with-markup)">
            {"prompt":"<xsl:value-of select="$text"
            "completion":<xsl:value-of select="$text-with-markup"/>}
        </xsl:if>
    </xsl:for-each>
</xsl:template>
```
Since the format is JSON, the prompt and completion strings need to be cleaned up by escaping the quotes and apostrophes. This function is used from the above template.

```xml
<xsl:function name="oxy:clean">
  <xsl:param name="txt"/>
  <xsl:variable name="apos">'
  </xsl:variable>
  <xsl:variable name="quot">"</xsl:variable>
  <xsl:variable name="txt">
    <xsl:value-of select="replace(normalize-space($txt), $quot, $apos)"/>
  </xsl:variable>
  <xsl:variable name="txt">
    <xsl:value-of select="replace($txt,'\\', '\\\\')"/>
  </xsl:variable>
  <xsl:sequence select="$txt"/>
</xsl:function>
```

Next, to serialize the markup from a paragraph, all the inlines of interest are matched and the element names and attributes are emitted for each of them:

```xml
<!-- Match the inline elements and serializes them. -->
<xsl:template match="*">
  <!-- Only non empty elements. -->
  <xsl:text>&lt;</xsl:text>
  <xsl:if test="not(*)">
    <xsl:value-of select="local-name()"/>
  </xsl:if>
  <xsl:for-each select="*[contains(@class, ' ui-d/uicontrol ') or contains(@class, ' ui-d/menucascade ') or contains(@class, ' ui-d/shortcut ') or contains(@class, ' topic/i ') or contains(@class, ' topic/b ') or contains(@class, ' sw-d/filepath ') or contains(@class, ' pr-d/codeph ') or contains(@class, ' pr-d/codeblock ')]" mode="serialize-markup">
    <xsl:value-of select="local-name()"/>
  </xsl:for-each>
</xsl:template>
```

To apply this stylesheet, create an XSLT transformation scenario in Oxygen XML Editor. You can choose any well-formed XML file as the input (the stylesheet scans a folder using the collection function, so this input will be ignored) and specify the output file.

**Actions for Text Refactoring**

Under the list of proposals, there is a toolbar that shows a set of buttons you can use to alter or refactor the selected text from the editor. For example, to transform a paragraph into a list with the summary of ideas, you can select its text, then click the **Ordered Lists** button. The selection will be replaced with a new structured text.

**Oxygen Emmet Plugin**

An **Oxygen Emmet Plugin** is available as an add-on and it provides the means for high-speed coding and editing in **Text** mode or **Author** mode via a content assistance mechanism. It can be used for HTML, XSL, CSS, LESS, and other formats. For example, with the Emmet add-on installed, you can type abbreviations (similar to CSS selectors) and expand them into full-fledged HTML code. The add-on contributes a submenu named **Emmet** in the contextual menu and it contains actions for expanding abbreviations or wrapping content with an expanded abbreviation. The two actions can also be invoked using the **Alt + Shift + E (Ctrl + Shift + E on macOS)** or **Alt + Shift + W (Ctrl + Shift + W on macOS)** keyboard shortcuts.

**Quick Installation**

You can drag the following **Install** button and drop it into the main editor in **Oxygen** (version 24.1 or newer) to quickly initiate the installation process:
Manual Installation

To manually install this add-on, follow this procedure:

1. Go to Help > Install new add-ons to open an add-on selection dialog box. Enter or paste https://www.oxygenxml.com/InstData/Addons/default/updateSite.xml in the Show add-ons from field or select it from the drop-down menu.

   **Tip:**
   For HTML, CSS, LESS, or XSL files, you can simply right-click anywhere in the editor pane and select Emmet from the pop-up menu.

2. Select the Oxygen Emmet Plugin add-on and click Next.
3. Read the end-user license agreement. Then select the I accept all terms of the end-user license agreement option and click Finish.
4. Restart the application.

**Result:** The Emmet actions will now be available using the keyboard shortcuts or in the Emmet submenu (located in the contextual menu of Text mode or Author mode).

**Emmet Actions**

The two contributed actions are:

**Expand abbreviation** [Alt + Shift + E (Ctrl + Shift + E on macOS)]

   In Text mode, after entering an abbreviation, invoking this action will expand a valid abbreviation into a code snippet, depending on the document type.

   In Author mode, invoking the action opens a dialog box where you can enter an abbreviation. After you click OK, a valid abbreviation is expanded into a code snippet, depending on the document type.

   **Tip:**
   For HTML, CSS, LESS, or XML-based document types, you can also use Ctrl + Space to expand Emmet abbreviations.

**Wrap with abbreviation** [Alt + Shift + W (Ctrl + Shift + W on macOS)]

   It opens a dialog box where you can enter an abbreviation and after clicking OK, the abbreviation is expanded with the selected content added in the last element of the generated snippet.

**Abbreviation Expansion Examples**

Here are some examples for HTML:
• **Expand abbreviation** example:

```
#page>div.logo+ul#navigation>li*5>a{Item $}
```

expands into:

```html
<div id="page">
  <div class="logo"></div>
  <ul id="navigation">
    <li><a href="" title="Item 1">Item 1</a></li>
    <li><a href="" title="Item 2">Item 2</a></li>
    <li><a href="" title="Item 3">Item 3</a></li>
    <li><a href="" title="Item 4">Item 4</a></li>
    <li><a href="" title="Item 5">Item 5</a></li>
  </ul>
</div>
```

• **Wrap with abbreviation** example:

If the following content is selected to be wrapped:

```
About
News
Products
Contacts
```

then

```
ul>li[title=$#](#$)+img[src=https://www.ex1.com/$#[alt=item$]
```

expands into:

```html
<ul>
  <li title="About">About<img src="https://www.ex1.com/About" alt="item1"></li>
  <li title="News">News<img src="https://www.ex1.com/News" alt="item2"></li>
  <li title="Products">Products<img src="https://www.ex1.com/Products" alt="item3"></li>
  <li title="Contacts">Contacts<img src="https://www.ex1.com/Contacts" alt="item4"></li>
</ul>
```

You can also use Emmet abbreviations for other XML documents. Here are some examples of expanded abbreviations for DITA:

• `prolog>author {AuthorName}`

expands into:
• simpletable {strow>stentry*4}*4

expands into a 4x4 simple table.

• ul>li*3

expands into an unordered list with 3 list items.

• ol>li[id="item$"]*3

expands into:

```html
<ol id="ol_gff_bjd_mkb">
  <li id="item1"/>
  <li id="item2"/>
  <li id="item3"/>
</ol>
```

Here are a few CSS examples:

• @f+

expands into:

```css
@font-face {
  font-family: 'FontName';
  src: url('FileName.eot');
  src: url('FileName.eot?#iefix') format('embedded-opentype'),
       url('FileName.woff') format('woff'),
       url('FileName.ttf') format('truetype'),
       url('FileName.svg#FontName') format('svg');
  font-style: normal;
  font-weight: normal;
}
```

• -br

expands into:

```css
-webkit-border-right: ;
-moz-border-right: ;
-ms-border-right: ;
```
Writer Helper Add-on (Experimental)

Oxygen XML Editor offers an add-on that provides support for helping technical writers. Once the add-on is installed, a Writer Helper view is available from the Window > Show View menu. Once opened, you can use one of its available tabs to access the functionality.

Quick Installation

You can drag the following Install button and drop it into the main editor in Oxygen (version 24.1 or newer) to quickly initiate the installation process:

Install

Manual Installation

To manually install the add-on, follow these instructions:

1. Go to Help > Install new add-ons to open an add-on selection dialog box.
2. Enter or paste http://www.oxygenxml.com/InstData/Addons/default/updateSite.xml in the Show add-ons from field.
3. Select the Writer Helper add-on and click Next.
4. Read the end-user license agreement. Then select the I accept all terms of the end-user license agreement option and click Finish.
5. Restart Oxygen XML Editor.

Result: The Writer Helper view now be available and can be selected from the Window > Show View menu.

Review

You can select folders or files in the Project view and start a review session for them. By default, the review is done in random order and all reviewed documents are opened with change tracking enabled. During the review process, you can see a list of already reviewed documents and check the status of your progress.

This feature is useful for reviewing a set of documents.
Similar Content

You can select folders or files in the Project view, then choose a list of elements to check for similarity. Once you start the Find similar content processing, information from all XML, HTML, and Markdown files is gathered to determine elements that have similar text content. Once the similar matches are presented, you can click on each of them to show a detailed list of places where the matches are found.

If the Show similar content for the current file checkbox is enabled and the similar elements have been found, once you open a document, a results tab will show elements the current document has in common with all other documents that were analyzed.

This feature is useful for finding possible elements that can be reused instead of being copy-pasted between documents.

Read Aloud

Select content in the current document and use the Play button to listen to how it is read by an operating system-specific narrator.

This feature is useful to review content by listening to it.

Tips

You can view a list of editing tips, tuned for technical writers. Each tip might also have a Mode details link to open the Oxygen XML Editor user guide.

This feature is useful for discovering productivity tips to help make your editing experience more efficient and productive.

Resources

For more information about the Writer Helper add-on, as well as details regarding other popular add-ons that extend the functionality of Oxygen XML Editor, watch the following webinar:

• Webinar: Add-ons You Can Use for Technical Writing

Development

XSpec Helper View Add-on

Oxygen XML Editor offers an add-on that contributes an XSpec Test Results view, that provides the ability to run XSpec test scenarios and view the results directly in the application.

Quick Installation

You can drag the following Install button and drop it into the main editor in Oxygen (version 24.1 or newer) to quickly initiate the installation process:
Manual Installation

To manually install the add-on, follow these instructions:

1. Go to Help > Install new add-ons to open the add-on selection dialog box.
3. Select the XSpec Helper View and XSpec Framework add-ons (both are required), and click Next.
4. Read the end-user license agreement. Then select the I accept all terms of the end-user license agreement option and click Finish.
5. Restart the application.

The add-on contributes a Run XSpec test scenarios button on the main toolbar. Click that button to execute an XSpec file and open its results in the XSpec Test Results view. At this point you can do the following:

- Switch to the XSLT and use the Run buttons in this view to execute a particular scenario.
- For each test, there is also a Show button that selects the corresponding test in the main editing area.
- For failed tests, you can click a particular test to open a diff comparison between the expected and actual results.

For more information, see the details for this Oxygen XSpec Helper View add-on in GitHub.

Saxon XSLT and XQuery Transformer Add-on

Oxygen XML Editor offers an add-on that installs an external Saxon XSLT and XQuery processor and allows you to validate and transform XSLT and XQuery documents with that external processor. The default add-on update site includes versions for Saxon 9.6, 9.7, 9.8, 9.9, 10.8, and 11.2.

Manual Installation

To manually install the add-on, follow these instructions:

1. Go to Help > Install new add-ons to open an add-on selection dialog box.
2. Enter or paste https://www.oxygenxml.com.InstData/Addons/default/updateSite.xml in the Show add-ons from field or select it from the drop-down menu.
3. Select one of the versions of the Saxon XSLT and XQuery Transformer add-on (9.6, 9.7, 9.8, 9.9, 10.8, or 11.2) and click Next.
4. Read the end-user license agreement. Then select the I accept all terms of the end-user license agreement option and click Finish.
5. Restart the application.
Result: When you configure the validation or transformation scenario, you will now have the option to choose the particular Saxon transformer.

Restriction: Debugging XSLT/XQuery transformations based on this engine is NOT supported.

ICU4J Library Add-on

Oxygen XML Editor offers an add-on that provides the entire ICU4J JAR library used for sorting and collations with XSLT processors. The Saxon 9 XSLT processor has an `<xsl:sort>` element that can be used to sort elements based on their text content. If the XML elements contain text in exotic languages, Saxon needs a more powerful sort algorithm and this JAR library adds this capability.

Manual Installation

To manually install it the add-on, follow these instructions:

1. Go to Help > Install new add-ons to open an add-on selection dialog box.
2. Enter or paste https://www.oxygenxml.com/InstData/Addons/default/updateSite.xml in the Show add-ons from field or select it from the drop-down menu.
3. Select the Contribute ICU4J Library add-on and click Next.
4. Read the end-user license agreement. Then select the I accept all terms of the end-user license agreement option and click Finish.
5. Restart the application.

Result: The ICU4J library is now contributed to the Oxygen installation directory.

For more information, see the details for this Contribute ICU4J Library add-on in GitHub.

XSD to JSON Schema Converter

Oxygen XML Editor includes a tool for converting an XML Schema file (XSD) to a JSON Schema file. The XSD to JSON Schema action for invoking the tool can be found in the Tools > JSON Tools menu. It requires an additional add-on to be installed, so the first time you invoke the action, Oxygen XML Editor will present a dialog box asking if you want to install it. Once installed, you need to restart Oxygen XML Editor and the XSD to JSON Schema action will invoke the tool.

Quick Installation

You can drag the following Install button and drop it into the main editor in Oxygen (version 24.1 or newer) to quickly initiate the installation process:
Manual Installation

To manually install the add-on, follow these instructions:

1. Go to Help > Install new add-ons to open an add-on selection dialog box.
2. Enter or paste https://www.oxygenxml.com/InstData/Addons/default/updateSite.xml in the Show add-ons from field or select it from the drop-down menu.
3. Select the XSD to JSON Schema add-on and click Next.
4. Read the end-user license agreement. Then select the I accept all terms of the end-user license agreement option and click Finish.
5. Restart the application.

Result: The XSD to JSON Schema dialog box is now available and can be selected from the Tools > JSON Tools menu.

Converting XSD to JSON Schema

To convert an XML Schema (XSD) to a JSON Schema, follow these steps:

1. Select the XSD to JSON Schema action from the Tools > JSON Tools menu.

   **Step Result:** The XSD to JSON Schema dialog box is displayed:

   ![Figure 610. XSD to JSON Schema Dialog Box](image)

2. In the XSD URL field, choose or enter the URL of the XML Schema document. The conversion supports XSD versions 1.0 and 1.1.
3. In the Output file field, choose the path for the resulting output file.
4. [Optional] You can select the Open in Editor option to open the resulting JSON Schema document in the main editing pane.
5. For the JSON Schema version option, choose the version of the resulting JSON schema. The possible choices are: Draft 4, Draft 6, Draft 7, and Draft 2019-09.
6. [Optional] If you select the **Restrict additional content** option, then `additionalProperties` (for objects) and `additionalItems` (for arrays) will be set to `false` in the resulting schema. By default, these keys are not in the schema, meaning that providing additional content (according to the schema) is allowed.

7. [Optional] You can select the **Preserve case of names from the XSD** option if you want the names from the XSD to remain unchanged in the resulting JSON Schema. Otherwise, the default JAXB naming algorithm will be applied (for example, "some.name" is changed to "SomeName", or "Some_oth3r_name" is changed to "SomeOth3RName").

8. Click the **Convert** button.

**Result:** The original XSD document is now converted to a JSON Schema document. The resulting JSON Schema will be the specified draft and will contain:

- The `$id` of the schema, generated from XSD `targetNamespace`.
- The `$definitions` section, which declares complex and enum types.
- The `anyOf` section, which lists possible top-level elements as an array of objects.

**Other Possible Results:**

- If an XSD type extends another type, then its schema is combined with the schema of the base type using the `allOf` keyword.
- If an extension in XSD defines an element with the same name as an attribute in the base, a property named `rest` is generated to avoid name conflicts in JSON.
- If a property of a complex type is a collection property, the schema of the collection items will be wrapped in the JSON array schema.

**Conversion Mappings**

The following table lists the specific conversion mapping details.

<table>
<thead>
<tr>
<th>XML Schema Type</th>
<th>JSON Schema Representation</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>anySimpleType</code></td>
<td>string, number, integer, boolean, null</td>
</tr>
<tr>
<td><code>anyType</code></td>
<td>string, number, integer, boolean, null, object, array</td>
</tr>
<tr>
<td><code>string</code></td>
<td>string</td>
</tr>
<tr>
<td><code>normalizedString</code></td>
<td>string</td>
</tr>
<tr>
<td><code>token</code></td>
<td>string</td>
</tr>
<tr>
<td><code>language</code></td>
<td>string</td>
</tr>
<tr>
<td><code>Name</code></td>
<td>string</td>
</tr>
<tr>
<td><code>NCName</code></td>
<td>string</td>
</tr>
<tr>
<td><code>ID</code></td>
<td>string</td>
</tr>
<tr>
<td><code>IDREF</code></td>
<td>string</td>
</tr>
<tr>
<td>XML Schema Type</td>
<td>JSON Schema Representation</td>
</tr>
<tr>
<td>------------------</td>
<td>----------------------------------------------------------------</td>
</tr>
<tr>
<td>IDREFS</td>
<td>array of strings</td>
</tr>
<tr>
<td>ENTITY</td>
<td>string</td>
</tr>
<tr>
<td>ENTITIES</td>
<td>array of strings</td>
</tr>
<tr>
<td>NMTOKEN</td>
<td>string</td>
</tr>
<tr>
<td>NMTOKENS</td>
<td>array of strings</td>
</tr>
<tr>
<td>boolean</td>
<td>boolean</td>
</tr>
<tr>
<td>base64Binary</td>
<td>array of integers</td>
</tr>
<tr>
<td>hexBinary</td>
<td>array of integers</td>
</tr>
<tr>
<td>float</td>
<td>number</td>
</tr>
<tr>
<td>decimal</td>
<td>number</td>
</tr>
<tr>
<td>integer</td>
<td>integer</td>
</tr>
<tr>
<td>nonPositiveInteger</td>
<td>integer</td>
</tr>
<tr>
<td>negativeInteger</td>
<td>integer</td>
</tr>
<tr>
<td>long</td>
<td>integer</td>
</tr>
<tr>
<td>int</td>
<td>integer</td>
</tr>
<tr>
<td>short</td>
<td>integer</td>
</tr>
<tr>
<td>byte</td>
<td>integer</td>
</tr>
<tr>
<td>nonNegativeInteger</td>
<td>integer</td>
</tr>
<tr>
<td>unsignedLong</td>
<td>integer</td>
</tr>
<tr>
<td>unsignedInt</td>
<td>integer</td>
</tr>
<tr>
<td>unsignedShort</td>
<td>integer</td>
</tr>
<tr>
<td>unsignedByte</td>
<td>integer</td>
</tr>
<tr>
<td>positiveInteger</td>
<td>integer</td>
</tr>
<tr>
<td>double</td>
<td>number</td>
</tr>
<tr>
<td>anyURI</td>
<td>string with &quot;format&quot;:&quot;uri&quot;</td>
</tr>
<tr>
<td>QName</td>
<td>object with &quot;namespaceURI&quot;, &quot;localPart&quot;, &quot;prefix&quot;</td>
</tr>
<tr>
<td>duration</td>
<td>string</td>
</tr>
<tr>
<td>dateTime</td>
<td>string with &quot;format&quot;:&quot;date-time&quot;</td>
</tr>
<tr>
<td>date</td>
<td>string with &quot;format&quot;:&quot;date&quot;</td>
</tr>
</tbody>
</table>
### Conversion Limitations

In most cases, the conversion creates an equivalent schema, but there are some limitations:

- Restrictions/facets are not taken into consideration when converting `fractionDigits`, `pattern`, `totalDigits`, `WhiteSpace`, `minInclusive`, `maxInclusive`, and the restrictions for length, except `enumeration`. However, extensions and indicators are properly converted (`minOccurs`, `maxOccurs`, `group`, `sequence`, `choice`).
- The `<documentation>` element is not converted into `<description>`.
- The `@substitutionGroup` attribute for an element that has no declared type becomes a reference to the element that can substitute it.
- The `@block` attribute is not taken into consideration during the conversion.

### Generating Java Classes from XML Schema

Oxygen XML Editor includes a tool for generating Java classes from an XML Schema (XSD) file. The Generate Java classes from XML Schema (XSD) action for invoking the tool can be found in the Tools menu. It requires an additional add-on to be installed, so the first time you invoke the action, Oxygen XML Editor will present a dialog box asking if you want to install it. Once installed, you need to restart Oxygen XML Editor and the action will invoke the Java class generator tool.

#### Quick Installation

You can drag the following Install button and drop it into the main editor in Oxygen (version 24.1 or newer) to quickly initiate the installation process:

![Install Button](image)

#### Manual Installation

To manually install it the add-on, follow these instructions:

1. Go to Help > Install new add-ons to open an add-on selection dialog box.
2. Enter or paste `https://www.oxygenxml.com/InstData/Addons/default/updateSite.xml` in the Show add-ons from field or select it from the drop-down menu.
3. Select the Generate Java classes from XML Schema (XSD) add-on and click Next.
4. Read the end-user license agreement. Then select the I accept all terms of the end-user license agreement option and click Finish.
5. Restart the application.

**Result:** The Generate Java classes from XML Schema (XSD) dialog box is now available and can be selected from the Tools menu.
Generating Java Classes

To generate Java classes, follow these steps:

1. Select the **Generate Java classes from XML Schema (XSD)** action from the **Tools** menu.

   **Step Result:** The **Generate Java classes from XML Schema (XSD)** dialog box is displayed:

   ![Generate Java Classes from XML Schema (XSD) Dialog Box]

   2. Choose or enter the **XSD URL** of the XML Schema document.
   3. Choose the path for the **Output folder** where the generated files will be stored.
   4. [Optional] You can choose the **Package name** for the Java package that will contain the generated source files. If not specified, the name will be generated based on the value of the `@targetNamespace` attribute.
   5. [Optional] You can select the **Open in Editor** option to open the **ObjectFactory.java** file in the editor. This java class contains factory methods for all other classes in the package.
   6. Click the **Generate** button.

   **Result:** The Java class files will be generated inside the new package, located in the specified output folder.

Generating JSON Schema Documentation

Oxygen XML Editor includes a tool for generating documentation for a JSON Schema file in HTML format. To generate JSON Schema documentation, select **JSON Schema Documentation** from the **Tools > Generate Documentation** menu. You can also open the tool by using the **Generate Documentation** toolbar button. This opens a dialog box where you can specify the location of the JSON Schema file and HTML output file.

This tool requires an additional add-on to be installed, so the first time you invoke the action, Oxygen XML Editor presents a dialog box asking if you want to install it. Once installed, you need to restart Oxygen XML Editor and the **JSON Schema Documentation** action will invoke the tool.

**Quick Installation**

You can drag the following **Install** button and drop it into the main editor in **Oxygen** (version 24.1 or newer) to quickly initiate the installation process:
Manual Installation

To manually install the add-on, follow these instructions:

1. Go to Help > Install new add-ons to open an add-on selection dialog box.
2. Enter or paste https://www.oxygenxml.com/InstData/Addons/default/updateSite.xml in the Show add-ons from field or select it from the drop-down menu.
3. Select the JSON Schema Documentation add-on and click Next.
4. Read the end-user license agreement. Then select the I accept all terms of the end-user license agreement option and click Finish.
5. Restart the application.

Result: The JSON Schema Documentation dialog box is now available and can be selected from the Tools > Generate Documentation menu.

JSON Schema Documentation Dialog Box

![JSON Schema Documentation Dialog Box](image)

The JSON Schema Documentation dialog box includes the following fields and options:

**JSON Schema URL**

The URL of the JSON Schema file. You can specify the path by using the text field, the history drop-down menu, or the browsing actions in the Browse drop-down list.

**Output file**
The path to the folder where the generated HTML file will be saved.

**Split output into multiple files**

If selected, the application splits the output into multiple files. You can choose between splitting them by **component** name or **location**.

**Open in Browser/System Application**

If selected, the generated result is opened in the system application associated with the output file type (HTML).

**Included component details**

This section can be used to specify whether or not the following components are shown in the generated documentation:

- **Annotations** - Displays the annotations (title, description) for each component (property or definition).
- **Constraints** - Displays the schema constraints for each component, according to its type.
- **Properties** - Displays the `properties` of an Object Schema.
- **Pattern Properties** - Displays the `patternProperties` of an Object Schema.
- **Enumerations** - Displays the enumerated values, if specified in the schema.
- **Source** - Displays the text schema source for each component.
- **Used By** - Displays the list of all the components that reference the current definition.
- **Composition** - Displays the `oneOf`, `anyOf`, and `allOf` Compositors that are used for combining schemas.
- **Diagram** - Displays the diagram image for each component. The diagrams are generated according to the options specified in the [Schema Design preferences page](on page 198). Diagrams are also generated for components within schemas from referenced files.
- **Image Map** - Diagrams will include hyperlinks that navigate to each particular component.
- **Location** - Displays the schema location for each component.

You can click **Generate** at any point to generate the JSON Schema documentation.

**Generated JSON Schema Documentation in HTML Format**

After generating the JSON Schema documentation, it is presented in a visual diagram style with various sections, hyperlinks, and options.
The generated documentation includes a Table of Contents on the left pane with links to particular sections in the right pane. You can collapse or expand details by using the Showing options or the Collapse or Expand buttons.

**OpenAPI Tester**

Oxygen XML Editor includes a testing tool for OpenAPI files. The tool provides the ability to inspect OpenAPI request responses and to ensure that they work as expected. It can be used for OpenAPI 3.x in JSON or YAML format.

To use the tool, select OpenAPI Tester from the Tools > JSON Tools menu. This opens a dialog box where you can specify the location of the OpenAPI file that you want to test.

This tool requires an additional add-on to be installed, so the first time you invoke the action, Oxygen XML Editor presents a dialog box asking if you want to install it. Once installed, you need to restart Oxygen XML Editor and the OpenAPI Tester action will invoke the tool.
Quick Installation

You can drag the following Install button and drop it into the main editor in Oxygen (version 24.1 or newer) to quickly initiate the installation process:

Install

Manual Installation

To manually install the add-on, follow these instructions:

1. Go to Help > Install new add-ons to open an add-on selection dialog box.
2. Enter or paste https://www.oxygenxml.com/InstData/Addons/default/updateSite.xml in the Show add-ons from field or select it from the drop-down menu.
3. Select the OpenAPI Tester add-on and click Next.
4. Read the end-user license agreement. Then select the I accept all terms of the end-user license agreement option and click Finish.
5. Restart the application.

Result: The OpenAPI Tester dialog box is now available and can be selected from the Tools > JSON Tools menu.

OpenAPI Tester Dialog Box

![OpenAPI Tester Dialog Box](image)

After selecting the URL for the OpenAPI document (either local or remote) and clicking OK, the OpenAPI Tester view becomes visible on the right side of the editor. The view can also be opened by selecting OpenAPI Tester from the Window > Show View menu.
The tester loads the selected OpenAPI document and fills in the corresponding fields. The tester fields are as follows:

**URL**

The URL of the OpenAPI file. You can specify the path by using the text field or the **Browse** button. If you specify the path directly in the text field, you need to click the **Reload** button. You can use the **Clear** button if you want to remove all the content from the view.

**Server**

The list of servers defined by the OpenAPI document. The global "servers" array can be overridden on the path level or operation level. If it is not provided or is empty, the server URL defaults to "/".

**Path**

The list of individual endpoints (paths) in the API. The full request URL is constructed as `<server-url>/path`.

**Operation**

The list of HTTP operations supported by the selected path. OpenAPI 3.0 supports get, post, put, patch, delete, head, options, and trace.

**Parameters tab**
The definition of the parameters for the selected operation. OpenAPI 3.0 supports parameters passed via path, query string, headers, and cookies. The required parameters will be marked with a red asterisk. For array parameters, items must be separated by a comma.

**Authorization tab**

The list of available authentication types. The authentication data is persistent and can be removed from the contextual menu.

**Request Body tab**

The media type and the body of the request (if the selected operation allows it). For example, GET will not allow specifying a body since that HTTP method disallows it. Oxygen XML Editor will create a sample request body based on the JSON Schemas defined in the OpenAPI document. Usually, you just need to change a few values for the request to be valid.

**Variables tab**

The list of variables used for server templating (if applicable). Variables are indicated by {curly brackets} in the server URL.

**Send button**

Executes the request by creating a HTTP client with all the information extracted from the view.

**Open response in editor**

If selected, the response will be opened in the main editing pane.

**Response area**

Initially this area is empty. After using the Send button, it presents the message received from the server in response to the request. The expected content type of the message is JSON. The response status and possible errors that may occur are presented right below this area. The status has a green foreground for successful requests and a red foreground otherwise.

**Generating OpenAPI Documentation**

Oxygen XML Editor includes a tool for generating documentation for OpenAPI components in HTML format, including annotations and cross references. The documentation displays information about the servers, paths, components and tags defined in the OpenAPI 3.0 documents and it is presented in a visual diagram style with various sections, hyperlinks, and filtering options.

**Quick Installation**

You can drag the following Install button and drop it into the main editor in Oxygen (version 24.1 or newer) to quickly initiate the installation process:

![Install Button]

**Manual Installation**

To manually install it the add-on, follow these instructions:
1. Go to Help > Install new add-ons to open an add-on selection dialog box.
2. Enter or paste https://www.oxygenxml.com/InstData/Addons/default/updateSite.xml in the Show add-ons from field or select it from the drop-down menu.
3. Select the OpenAPI Documentation Generator add-on and click Next.
4. Read the end-user license agreement. Then select the I accept all terms of the end-user license agreement option and click Finish.
5. Restart the application.

Result: The OpenAPI Documentation dialog box is now available and can be selected from the Tools > Generate Documentation menu.

Generating OpenAPI Documentation

To generate OpenAPI documentation, select OpenAPI Documentation from the Tools > Generate Documentation menu. This opens a dialog box where you can specify the location of the OpenAPI file and the HTML output file.

This tool requires an additional add-on to be installed, so the first time you invoke the action, Oxygen XML Editor presents a dialog box asking if you want to install it. Once installed, you need to restart Oxygen XML Editor and the OpenAPI Documentation action will invoke the tool.

Figure 616. OpenAPI Documentation Dialog Box

The OpenAPI Documentation dialog box includes the following fields and options:

OpenAPI URL

The URL of the OpenAPI file. You can specify the path by using the text field or the browsing button ( ).
**Output file**

The path to the folder where the generated HTML file will be saved.

**Split output into multiple files**

If selected, the application splits the output into multiple files. You can choose between splitting them by **component** name or **location**.

**Included component details**

This section can be used to specify whether or not details about the following components that belong to internal or imported schemas are shown in the generated documentation:

- **Annotations** - Displays the annotations (title, description) for each component (property or definition).
- **Constraints** - Displays the schema constraints for each component, according to its type.
- **Source** - Displays the text schema source for each component.
- **Location** - Displays the schema location for each component.
- **Used By** - Displays the list of all the components that reference the current definition.
- **Properties** - Displays the properties of an Object Schema.
- **Pattern Properties** - Displays the patternProperties of an Object Schema.
- **Enumerations** - Displays the enumerated values, if specified in the schema.
- **Diagram** - Displays the diagram image for each component. The diagrams are generated according to the options specified in the **Schema Design preferences page (on page 198)**. Diagrams are also generated for components within schemas from referenced files.
- **Composition** - Displays the oneOf, anyOf, and allOf compositors that are used for combining schemas.
- **Image Map** - Diagrams will include hyperlinks that navigate to each particular component.

You can click **Generate** at any point to generate the OpenAPI documentation.

**Generated OpenAPI Documentation in HTML Format**

After generating the OpenAPI documentation, it is presented in a visual diagram style with various sections, hyperlinks, and options.
The generated documentation includes a Table of Contents on the left pane with links to particular sections in the right pane. You can collapse or expand details by using the Showing options or the Collapse or Expand buttons.

Others

DocBook Checker Add-on

Oxygen XML Editor offers an add-on that allows you to validate DocBook files. It reports issues such as broken internal and external links, problems with images, or profiling conditions that conflict with profiling preferences.

Quick Installation

You can drag the following Install button and drop it into the main editor in Oxygen (version 24.1 or newer) to quickly initiate the installation process:

Manual Installation

To manually install the add-on, follow these instructions:
1. Go to Help > Install new add-ons to open an add-on selection dialog box.
2. Enter or paste https://www.oxygenxml.com/InstData/Addons/default/updateSite.xml in the Show add-ons from field or select it from the drop-down menu.
3. Select the DocBook Checker add-on and click Next.
4. Read the end-user license agreement. Then select the I accept all terms of the end-user license agreement option and click Finish.
5. Restart the application.

**Result:** A Check DocBook for Completeness action will now be available on the toolbar and in the contextual menu. This action opens a dialog box that offers various validation options for running a completeness check on the current DocBook document.

For more information, see the details for this DocBook Checker add-on in GitHub.

**CGM Image Support Add-on**

Oxygen XML Editor offers an add-on that provide experimental support for CGM 1.0 images. To allow the rendering of CGM images in Author mode, the Oxygen CGM support add-ons need to be installed in Oxygen XML Editor.

**Quick Installation**

You can drag the following Install button and drop it into the main editor in Oxygen (version 24.1 or newer) to quickly initiate the installation process:

[Install]

**Manual Installation**

To manually install the Oxygen CGM support add-on, follow these instructions:

1. Go to Help > Install new add-ons to open an add-on selection dialog box.
2. Enter or paste https://www.oxygenxml.com/InstData/Addons/default/updateSite.xml in the Show add-ons from field or select it from the drop-down menu.
3. Select the Oxygen CGM support add-on and click Next.
4. Read the end-user license agreement. Then select the I accept all terms of the end-user license agreement option and click Finish.
5. Restart the application.

**Result:** You should be able to see CGM images in Author mode.

For more information, see the details about this CGM Support add-on in GitHub.
Tools

Oxygen XML Editor includes a variety of helpful tools to help you accomplish XML-related tasks. This section presents many of those tools. These tools are available in the **Tools** menu and some of them can be launched through keyboard shortcuts or command-line scripts.

Refactoring XML Documents

In the life cycle of XML documents there are instances when the XML structure needs to be changed to accommodate various needs. For example, when an associated schema is updated, an attribute may have been removed, or a new element added to the structure.

These types of situations cannot be resolved with a traditional *Find/Replace* tool, even if the tool accepts regular expressions. The problem becomes even more complicated if an XML document is computed or referenced from multiple modules, since multiple resources need to be changed.

To assist you with these types of refactoring tasks, Oxygen XML Editor includes a specialized **XML Refactoring** tool that helps you manage the structure of your XML documents.

XML Refactoring Tool

The **XML Refactoring** tool is presented in the form of an easy to use wizard that is designed to reduce the time and effort required to perform various structure management tasks. For example, you can insert, delete, or rename an attribute in all instances of a particular element that is found in all documents within your project.

To access the tool, select the **XML Refactoring** action from one of the following locations:

- The **Tools** menu.
- The **Refactoring** submenu from the contextual menu in the **Project** view (on page 402).
- The **Refactoring** submenu from the contextual menu in the **DITA Maps Manager** view (on page 2897).

**Note:**
The built-in refactoring operations are also available from the **Refactoring** submenu in the contextual menu of **Author** or **Text** mode. This is useful because by selecting the operations from the contextual menu, Oxygen XML Editor considers the editing context to skip directly to the wizard page of the appropriate operation and to help you by preconfiguring some of the parameter values. For your convenience, the last 5 operations that were finished (on page 2583) or previewed (on page 2583) also appear in the **Refactoring** submenu of the contextual menu in the **Project** view and the **DITA Maps Manager**.
XML Refactoring Wizard

The XML Refactoring tool includes the following wizard pages:

**Refactoring operations**

The first wizard page presents the available operations, grouped by category. To search for an operation, you can use the filter text box at the top of the page.

![XML Refactoring Wizard](image)

**Configure Operation Parameters**

The next wizard page allows you to specify the parameters for the refactoring operation. The parameters are specific to the type of refactoring operation that is being performed. For example, to delete an attribute you need to specify the parent element and the qualified name of the attribute to be removed.
Scope and Filters

The last wizard page allows you to select the set of files that represent the input of the operation.

Scope section

In the Scope section, you can select from predefined resource sets (such as the current file, your whole project, the current DITA map (on page 3218) hierarchy for DITA projects, etc.) or you can define your own set of resources by creating a working set (on page 3224).

Filters
The Filters section includes the following options:

- **Include files** - Allows you to filter the selected resources by using a file pattern. For example, to restrict the operation to only analyze build files you could use `build*.xml` for the file pattern.
- **Restrict only to known XML file types** - When selected, only resources with a known XML file type will be affected by the operation.
- **Look inside archives** - When selected, the resources inside archives will also be affected.

### Preview

You can use the **Preview** button to open a comparison panel where you can review all the changes that will be made by the refactoring operation before applying the changes.

### Finish

After clicking the **Finish** button, the operation will be processed and Oxygen XML Editor provides no automatic means for reverting the operations. Any **Undo** action will only revert changes on the current document.

---

**Troubleshooting:**

If an operation fails, a notification will be displayed in the **Results** panel with some information about the error. For example, if the operation was invoked on a read-only resource, the error will indicate that a read-only file cannot be converted.

**Tip:**

If an operation takes longer than expected you can use the ▶️ **Stop** button in the progress bar to cancel the operation.

**Restriction:**

XML refactoring operations cannot preserve CDATA sections. If your document contains XML CDATA sections, the refactoring operations will convert them to plain text nodes.

---

**Built-in Refactoring Operations**

The XML Refactoring tool includes a variety of built-in operations that can be used for common refactoring tasks. They are grouped by category in the **Refactoring operations** wizard page. You can also access the operations from the **Refactoring** submenu in the contextual menu of **Author** or **Text** mode. The operations are also grouped by category in this submenu. When selecting the operations from the contextual menu, Oxygen XML Editor considers the editing context to get the names and namespaces of the current element or attribute, and uses this information to preconfigure some of the parameter values for the selected refactoring operation.
The following built-in operations are available:

**Refactoring Operations for Attributes**

**Add/Change attribute**

Use this operation to change the value of an attribute or insert a new one. This operation allows you to specify the following parameters:

**Parent element section**

**Element**

The parent element of the attribute to be changed, in the form of a local name from any namespace, a local name with a namespace prefix, or an XPath expression.

**Attribute section**

**Local name**

The local name of the affected attribute.

**Namespace**

The namespace of the affected attribute.

**Value**

The value for the affected attribute.

**Options section**

You can choose between one of the following options for the **Operation mode**:

**Add the attribute in the parent elements where it is missing**

Adds the attribute to all instances of the specified parent element.

**Change the value in the parent elements where the attribute already exists**

Replaces the value of the already existing attribute in all instance of the specified parent element.

**Both**

Adds the attributes to the instances where it is missing and replaces the value in instances where the attribute already exists.

**Convert attribute to element**
Use this operation to convert a specified attribute to an element. This operation allows you to specify the following parameters:

**Parent element section**

**Element**

The parent element of the attribute to be converted, in the form of a local name from any namespace, a local name with a namespace prefix, or an XPath expression.

**Attribute section**

**Local name**

The local name of the affected attribute.

**Namespace**

The namespace of the affected attribute.

**New element section**

**Local name**

The local name of the new element.

**Namespace**

The namespace of the new element.

**Delete attribute**

Use this operation to remove one or more attributes. This operation requires you to specify the following parameters:

**Element**

The parent element of the attribute to be deleted, in the form of a local name from any namespace, a local name with a namespace prefix, or an XPath expression.

**Attribute**

The name of the attribute to be deleted.

**Rename attribute**

Use this operation to rename an attribute. This operation requires you to specify the following parameters:

**Element**

The parent element of the attribute to be renamed, in the form of a local name from any namespace, a local name with a namespace prefix, or an XPath expression.

**Attribute**

The name of the attribute to be renamed.
New local name

The new local name of the attribute.

Replace in attribute value

Use this operation to search for a text fragment inside an attribute value and change the fragment to a new value. This operation allows you to specify the following parameters:

Target attribute section

Element

The parent element of the attribute to be modified, in the form of a local name from any namespace, a local name with a namespace prefix, or an XPath expression.

Attribute

The name of the attribute to be modified.

Find / Replace section

Find

The text fragments to find. You can use Perl-like regular expressions.

Replace with

The text fragment to replace the target with. This parameter can bind regular expression capturing groups ($1, $2, etc.) from the find pattern.

Refactoring Operations for Comments

Delete comments

Use this operation to delete comments from one or more elements. This operation requires you specify the following parameter:

Element

The target element (or elements) that will have comments deleted, in the form of a local name from any namespace, a local name with a namespace prefix, or an XPath expression.

Note:
Comments that are outside the root element will not be deleted because the serializer preserves the content before and after the root.

Refactoring Operations for DITA Topics

Change topic ID to file name
Use this operation to change the ID of a topic to be the same as its file name.

**Convert CALS tables to simple tables**

Use this operation to convert DITA CALS tables to simple tables.

**Convert DITA 1.3 Maps and Topics to DITA 2.0**

Use this operation to convert topics and maps that adhere to the DITA 1.3 standard to the DITA 2.0 standard.

- Changes DOCTYPE declarations and XML Schema/Relax NG schema references.
- DITA Map changes:
  - Removes the `@lockmeta` attribute.
  - Removes the `<topicset>` and `<topicsetref>` elements.
  - Removes the `<anchor>` and `<anchorref>` elements and the `@anchorref` attribute.
  - Migrates the `@navtitle` attribute as a `<navtitle>` element.
  - Migrates the `@title` attribute as a `<title>` element.
  - Converts the `@copy-to` attribute to a `<resourceid>` element.
  - Replaces the `@print` attribute with an `@deliveryTarget` attribute.
  - Convert topicmeta `<linktext>` to `<linktitle>`.
  - Removed `<hasInstance>, `<hasKind>, `<hasNarrower>, `<hasPart>, `<hasRelated>, and `<relatedSubjects>` from subject scheme relationship tables in subject scheme, including `<subjectRelTable>, `<subjectRelHeader>, `<subjectRel`, and `<subjectRole>`.
- DITA task changes:
  - Converts the `<substep>` element to a `<step>` element.
  - Converts the `<substeps>` element to a `<steps>` element.
- DITA topic changes:
  - Removes the `@type` attribute with the value `fastpath`.
  - Converts the `@alt` attribute to an `<alt>` element.
  - Replaces the `<index-sort-as>` element with a `<sort-as>` element.
  - Removes the `<itemgroup>` element.
  - Moves the contents of the `<titlealts>` element inside the `<prolog>`.
  - Removes the `@domains` attribute.
  - Renames `<sectiondiv>` to `<div>`.
  - Remove `@query` attribute from `<link>` element.
  - Remove `@specentry` attribute from `<stentry>` element.
  - Remove the `@spectitle` attribute.

**Convert conrefs to conkeyrefs**

Use this operation to convert `@conref` attributes to `@conkeyref` attributes. For more information and instructions for using this operation, see Converting Conrefs to Conkeyrefs (on page 3041).

**Convert Nested Topics to New Topics (Available from the contextual menu of editable maps/nodes in the DITA Maps Manager (on page 2897))**
Use this operation on topics that contain nested `<topic>` elements to convert each nested topic to a new topic. Also, the new topics are added in the **DITA Maps Manager** as the first child topics of the original topic.

**Convert Sections to New Topics (Available from the contextual menu of editable maps/nodes in the DITA Maps Manager (on page 2897))**

Use this operation on topics that contain multiple sections to convert each section to a new topic. Also, the new topics are added in the **DITA Maps Manager** as the first child topics of the original topic.

**Convert simple tables to CALS tables**

Use this operation to convert DITA simple tables to CALS tables.

**Convert to Concept**

Use this operation to convert a DITA topic (of any type) to a DITA Concept topic type (for example, Topic to Concept). For more information, see **Converting DITA Topics to Another Type (on page 2970).**

**Convert to General Task**

Use this operation to convert a DITA topic (of any type) to a DITA General Task topic type (for example, Task to General Task). For more information, see **Converting DITA Topics to Another Type (on page 2970).**

**Convert to Reference**

Use this operation to convert a DITA topic (of any type) to a DITA Reference topic type (for example, Topic to Reference). For more information, see **Converting DITA Topics to Another Type (on page 2970).**

**Convert to Task**

Use this operation to convert a DITA topic (of any type) to a DITA Task topic type (for example, Topic to Task). For more information, see **Converting DITA Topics to Another Type (on page 2970).**

**Convert to Topic**

Use this operation to convert a DITA topic (of any type) to a DITA Topic (for example, Task to Topic). For more information, see **Converting DITA Topics to Another Type (on page 2970).**

**Convert to Troubleshooting**

Use this operation to convert a DITA topic (of any type) to a DITA Troubleshooting topic type (for example, Topic to Troubleshooting). For more information, see **Converting DITA Topics to Another Type (on page 2970).**

**Rename Key**

Use this operation to rename a key. It also updates all references to it.
**Note:**
It does not work on DITA 1.3 key scopes.

**Generate IDs**

Use this operation to automatically generate unique IDs for elements.

**Scope and Filters:**

All of the DITA refactoring actions allow you to choose a scope for the operation and some filters:

**Scope**

Select from a variety of options to define the scope that will have resources affected by the operation. For example, you can choose to affect all resources in the **Project**, **All opened files**, **Current DITA map hierarchy**, or just the **Current file**.

**Filters section**

- **Include files**
  
  Specifies files to be excluded from the operation. You can specify multiple files by separating them with commas and the patterns can include wildcards (such as * or ?).

- **Restrict to known XML file types only**
  
  Excludes non-XML file types from the operation.

- **Look inside archives**
  
  If this option is selected, the scope of the operation will include files inside archives.

**Refactoring Operations for DITA Maps**

- **Convert DITA Bookmap to Map**
  
  Convert a DITA bookmap to a DITA map.

- **Convert DITA Map to Bookmap**
  
  Convert a DITA map to a DITA bookmap.

- **Change or remove profiling attribute value**
  
  Change or remove a value from a DITA profiling attribute. A profiling attribute can have multiple values, separated by spaces (e.g. for `platform="windows redhat"`, you can change the current `redhat` value to `linux`). Select the name of the profiling attribute, the current value to replace, and the new value. If the new value is left empty, the current value is removed from the profiling attribute.

- **Define keys for all topic references**
This refactoring action is useful for converting links inside a DITA project from direct to indirect key-based addressing. When applied on DITA resources from your project (DITA maps and topics), this refactoring action defines keys for all of a DITA map's topic references based on the referenced file name and converts each direct reference to a key reference in each DITA topic. If a topic references already has keys defined, the action does not define new ones. Inside the DITA topics, whenever there is a link element (<xref> or <link>) with a direct reference to another DITA topic or an element with a @conref, the action attempts to convert them to indirect key-based addressing. The refactoring action may introduce linking errors or create duplicate keys so it is advised to run the Validate and check for completeness action from the DITA Maps Manager toolbar to manually fix those problems. You can enable the Report duplicate keys checkbox to also report any keys that are defined more than once.

Scope and Filters:

All of the DITA refactoring actions allow you to choose a scope for the operation and some filters:

Scope

Select from a variety of options to define the scope that will have resources affected by the operation. For example, you can choose to affect all resources in the Project, All opened files, Current DITA map hierarchy, or just the Current file.

Filters section

Include files

Specify files to be excluded from the operation. You can specify multiple files by separating them with commas and the patterns can include wildcards (such as * or ?).

Restrict to known XML file types only

Excludes non-XML file types from the operation.

Look inside archives

If this option is selected, the scope of the operation will include files inside archives.

Refactoring Operations for Elements

Delete element

Use this operation to delete elements. This operation requires you to specify the following parameter:

Element

The target element to be deleted, in the form of a local name from any namespace, a local name with a namespace prefix, or an XPath expression.

Delete element content
Use this operation to delete the content of elements. This operation requires you to specify the following parameter:

**Element**

The target element whose content is to be deleted, in the form of a local name from any namespace, a local name with a namespace prefix, or an XPath expression.

**Insert element**

Use this operation to insert new elements. This operation allows you to specify the following parameters:

**Element section**

**Local name**

The local name of the element to be inserted.

**Namespace**

The namespace of the element to be inserted.

**Location section**

**XPath**

An XPath expression that identifies an existing element to which the new element is relative, in the form of a local name from any namespace, a local name with a namespace prefix, or other XPath expressions.

**Position**

The position where the new element will be inserted, in relation to the specified existing element. The possible selections in the drop-down menu are: **After**, **Before**, **First child**, or **Last child**.

**Rename element**

Use this operation to rename elements. This operation requires you to specify the following parameters:

**Target elements (XPath)**

The target elements to be renamed, in the form of a local name from any namespace, a local name with a namespace prefix, or other XPath expressions.

**New local name**

The new local name of the element.

**Unwrap element**
Use this operation to remove the surrounding tags of elements, while keeping the content unchanged. This operation requires you to specify the following parameter:

**Target elements (XPath)**

The target elements whose surrounding tags will be removed, in the form of a local name from any namespace, a local name with a namespace prefix, or other XPath expressions.

**Wrap element**

Use this operation to surround elements with element tags. This operation allows you to specify the following parameters:

**Target elements (XPath)**

The target elements to be surrounded with tags, in the form of a local name from any namespace, a local name with a namespace prefix, or other XPath expressions.

**Wrapper element section**

- **Local name**
  The local name of the *Wrapper element*.

- **Namespace**
  The namespace of the *Wrapper element*.

**Wrap element content**

Use this operation to surround the content of elements with element tags. This operation allows you to specify the following parameters:

**Target elements (XPath)**

The target elements whose content will be surrounded with tags, in the form of a local name from any namespace, a local name with a namespace prefix, or other XPath expressions.

**Wrapper element section**

- **Local name**
  The local name of the *Wrapper element* that will surround the content of the target.

- **Namespace**
  The namespace of the *Wrapper element* that will surround the content of the target.
Refactoring Operations for Fragments

Insert XML fragment

Use this operation to insert an XML fragment. This operation allows you to specify the following:

**XML Fragment**

The XML fragment to be inserted.

**Location section**

**XPath**

An XPath expression that identifies an existing element to which the inserted fragment is relative, in the form of a local name from any namespace, a local name with a namespace prefix, or other XPath expressions.

**Position**

The position where the fragment will be inserted, in relation to the specified existing element. The possible selections in the drop-down menu are: After, Before, First child, or Last child.

Replace element content with XML fragment

Use this operation to replace the content of elements with an XML fragment. This operation allows you to specify the following parameters:

**Target elements (XPath)**

The target elements whose content will be replaced, in the form of a local name from any namespace, a local name with a namespace prefix, or other XPath expressions.

**XML Fragment**

The XML fragment with which to replace the content of the target element.

Replace element with XML fragment

Use this operation to replace elements with an XML fragment. This operation allows you to specify the following parameters:

**Target elements (XPath)**

The target elements to be replaced, in the form of a local name from any namespace, a local name with a namespace prefix, or other XPath expressions.

**XML Fragment**

The XML fragment with which to replace the target element.
Refactoring Operations for *JATSKit*

**Add BITS DOCTYPE - NLM/NCBI Book Interchange 2.0**
Use this operation to add an NLM 'BITS' 2.0 DOCTYPE declaration.

**Add Blue DOCTYPE - NISO JATS Publishing 1.1**
Use this operation to add a JATS 'Blue' 1.1 DOCTYPE declaration.

**Normalize IDs**
Use this operation to normalize assigned IDs and assigned IDs to elements that are missing them.

All of these *JATSKit* refactoring actions allow you to choose a scope for the operation and some filters:

**Scope**
Select from a variety of options to define the scope for the resources that will be affected by the operation. For example, you can choose to affect all resources in the Project, All opened files, or just the Current file.

**Filters section**

- **Include files**
  Specifies files to be excluded from the operation. You can specify multiple files by separating them with commas and the patterns can include wildcards (such as * or ?).

- **Restrict to known XML file types only**
  Excludes non-XML file types from the operation.

- **Look inside archives**
  If this option is selected, the scope of the operation will include files inside archives.

Refactoring Operations for *Processing Instructions*

**Accept all tracked changes, remove all Oxygen-specific comments and highlights**
Use this operation to accept all application-specific tracked changes (from elements and attributes) or remove all application-specific comments or highlights. There are several options to choose from:

- **Accept all tracked changes**
  Accepts all application-specific tracked changes (from elements and attributes).

- **Remove comments**
  Removes all application-specific comments.

- **Remove highlights**
Removes all application-specific highlights.

**Delete processing instructions**

Use this operation to delete all processing instructions that have a certain target name from the processed documents. This operation requires you to specify the following parameter:

**Processing instruction target**

The target name of the processing instructions to delete.

**Note:**

Processing instructions that are outside the root element are not deleted because the serializer preserves the content before and after the root.

**Refactoring Operations for Publishing Template**

These operations are for those who use *Oxygen Publishing Templates* for WebHelp Responsive output customization.

**Migrate HTML Page Layout Files to v21**

Use this operation to convert custom HTML page layout files (on page 1622) that are included in a custom Publishing Template that was created in Oxygen XML Editor version 20.0 or 20.1 so that they will be compatible with Oxygen XML Editor version 21.0.

**Migrate HTML Page Layout Files to v22**

Use this operation to convert custom HTML page layout files (on page 1622) that are included in a custom Publishing Template that was created in Oxygen XML Editor versions 20.0 - 21.1 so that they will be compatible with Oxygen XML Editor version 22.0.

**Update HTML Pages**

**Attention:**

This operation is only used by Oxygen XML Editor and should not be used manually.

**Additional Notes:**

- There are some operations that allow `<ANY>` for the **local name** and **namespace** parameters. This value can be used to select an element or attribute regardless of its local name or namespace. Also, the `<NO_NAMESPACE>` value can be used to select nodes that do not belong to a namespace.
- Some operations have parameters that accept XPath expressions to match elements or attributes. In these XPath expressions you can only use the prefixes declared in the **Options > Preferences > XML > XSLT-XQUERY > XPath** (on page 258) page. This preferences page can be
Custom Refactoring Operations

While Oxygen XML Editor includes a variety of built-in XML refactoring operations to help you accomplish particular tasks, you can also create custom operations according to your specific needs. For example, you could create a custom refactoring operation to convert an attribute to an element and insert the element as the first child of the parent element.

An XML Refactoring operation is defined as a pair of resources:

- An **XQuery Update script** or **XSLT stylesheet** that Oxygen XML Editor will run to refactor the XML files.
- An **XML Operation Descriptor** file that contains information about the operation (such as the name, description, and parameters).

**Figure 621. Diagram of an XML Refactoring Operation**

All the defined custom operations are loaded by the **XML Refactoring Tool** and presented in the **Refactoring Operations** wizard page (on page 841), along with the built-in operations.
After the user chooses an operation and specifies its parameters, Oxygen XML Editor processes an XQuery Update or XSLT transformation over the input file. This transformation is executed in a safe mode, which implies the following:

- **When loading the document:**
  - The XInclude mechanism is disabled. This means that the resources included by using XInclude will not be visible in the transformation.
  - The DTD entities will be processed without being expanded.
  - The associated DTD will be not loaded, so the default attributes declared in the DTD will not be visible in the transformation.

- **When saving the updated XML document:**
  - The DOCTYPE will be preserved.

Note:
This can be changed using Saxon extension functions in XSLT (on page 873).

- The DTD entities will be preserved as they are in the original document when the document is saved.
- The attribute values will be kept in their original form without being normalized.
- The spaces between attributes are preserved. Basically, the spaces are lost by a regular XML serialization since they are not considered important.

The result of this transformation overrides the initial input file.

Note:
To achieve some of the previous goals, the XML Refactoring mechanism adds several attributes that are interpreted internally. The attributes belong to the http://www.oxygenxml.com/ns/xmlRefactoring/additional_attributes namespace. These attributes should not be taken into account when processing the input XML document since they are discarded when the transformed document is serialized.

Restriction:
Comments or processing instructions that are in any node before or after the root element cannot be modified by an XML Refactoring operation. In other words, XML Refactoring operations can only be applied on the root element and the nodes inside it. However, as a work around to this limitation, you can use Saxon extension functions and the XSLT stylesheet method (on page 873) to implement the new custom XML refactoring operation.

Creating a Custom Refactoring Operation

To create a custom refactoring operation, follow these steps:
1. Create an XQuery Update script (on page 863) or XSLT stylesheet file (on page 868).
2. Create an XML Refactoring Operation Descriptor file contains the path to the XQuery Update script (on page 866) or XSLT stylesheet (on page 871).
3. Store both files in one of the locations that Oxygen XML Editor (on page 875) scans when loading the custom operations.

**Result:** Once you run the XML Refactoring tool again, the custom operation appears in the Refactoring Operations wizard page (on page 841).

**Related information**
Storing and Sharing Refactoring Operations (on page 875)

### Custom Refactoring Script

The first step in creating a custom refactoring operation is to create an XQuery Update script (on page 863) or XSLT stylesheet (on page 868) that is needed to process the refactoring operations. The easiest way to create this script file is to use the New document wizard to create a new XQuery or XSLT file and you can use the XQuery method example (on page 863) or XSLT method example (on page 868) to help you with the content.

There are cases when it is necessary to add parameters in the XQuery script (on page 863) or XSLT stylesheet (on page 868). For instance, if you want to rename an element, you may want to declare an external parameter associated with the name of the element to be renamed. To allow you to specify the value for these parameters, they need to be declared in the refactoring operation descriptor file that is associated with this operation.

**Note:**
The XQuery Update processing is disabled by default in Oxygen XML Editor. Thus, if you want to create or edit an XQuery Update script you have to enable this mechanism by creating an XQuery transformation scenario (on page 1536) and choose Saxon EE as the transformation engine. Also, you need to make sure the Enable XQuery update option is selected in the Saxon processor advanced options (on page 1485).

**Note:**
If you are using an XSLT file, XPath expressions that are passed as parameters will automatically be rewritten to conform with the mapping of the namespace prefixes declared in the XML /XSLT-XQuery / XPath preferences page (on page 258).

The next step in creating a custom refactoring operation is to create an XML Refactoring Operation Descriptor file contains the path to the XQuery Update script (on page 866) or XSLT stylesheet (on page 871).
Custom Refactoring Operation Descriptor File

The second step in creating a custom refactoring operation is to create an operation descriptor file. The easiest way to do this is to use the New document wizard and choose the XML Refactoring Operation Descriptor template.

Introduction to the Descriptor File

This file contains information (such as name, description, and id) that is necessarily when loading an XML Refactoring operation. It also contains the path to the XQuery Update script (on page 863) or XSLT stylesheet (on page 868) that is associated with the particular operation through the <script> element.

You can specify a category for your custom operations to logically group certain operations. The <category> element is optional and if it is not included in the descriptor file, the default name of the category for the custom operations is Other operations.

The descriptor file is edited and validated against the following schema: frameworks/xml_refactoring/operation_descriptor.xsd.

Declaring Parameters in the Descriptor File

If the XQuery Update script or XSLT stylesheet includes parameters, they should be declared in the parameters section of the descriptor file. All the parameters specified in this section of the descriptor file will be displayed in the XML Refactoring tool within the Configure Operation Parameters wizard page (on page 842) for that particular operation.

The value of the first <description> element in the <parameters> section will be displayed at the top of the Configure Operation Parameters wizard page (on page 842).

To declare a parameter, specify the following information:

- **label** - This value is displayed in the user interface for the parameter.
- **name** - The parameter name used in the XQuery Update script or XSLT stylesheet and it should be the same as the one declared in the script.
- **type** - Defines the type of the parameter and how it will be rendered. There are several types available:
  - **TEXT** - Generic type used to specify a simple text fragment.
  - **XPATH** - Type of parameter whose value is an XPATH expression. For this type of parameter, Oxygen XML Editor will use a text input with corresponding content completion and syntax highlighting.
Note:
The value of this parameter is transferred as plain text to the XQuery Update or XSLT transformation without being evaluated. You should evaluate the XPath expression inside the XQuery Update script or XSLT stylesheet. For example, you could use the `saxon:evaluate` Saxon extension function.

Note:
A relative XPath expression is converted to an absolute XPath expression by adding `//` before it (`//XPathExp`). This conversion is done before transferring the XPath expression to the XML refactoring engine.

Note:
When writing XPath expressions, you can only use prefixes declared in the Options > Preferences > XML > XSLT-XQuery > XPath (on page 258) options page.

- **NAMESPACE** - Used for editing namespace values.
- **REG_EXP_FIND** - Used when you want to match a certain text by using Perl-like regular expressions.
- **REG_EXP_REPLACE** - Used along with **REG_EXP_FIND** to specify the replacement string.
- **XML_FRAGMENT** - This type is used when you want to specify an XML fragment. For this type, Oxygen XML Editor will display a text area specialized for inserting XML documents.
- **NC_NAME** - The parameter for **NC_NAME** values. It is useful when you want to specify the local part of a **QName** (on page 3223) for an element or attribute.
- **BOOLEAN** - Used to edit boolean parameters.
- **TEXT_CHOICE** - It is useful for parameters whose value should be from a list of possible values. Oxygen XML Editor renders each possible value as a radio button option.

- **description** - The description of the parameter. It is used by the application to display a tooltip when you hover over the parameter.

- **possibleValues** - Contains the list with possible values for the parameter and you can specify the default value, as in the following example:

```xml
<possibleValues onlyPossibleValuesAllowed="true">
  <value name="before">Before</value>
  <value name="after" default="true">After</value>
  <value name="firstChild">First child</value>
  <value name="lastChild">Last child</value>
</possibleValues>
```
Specialized Parameters to Match Elements or Attributes

If you want to match elements or attributes, you can use some specialized parameters, in which case Oxygen XML Editor will propose all declared elements or attributes based on the schema associated with the currently edited file. The following specialized parameters are supported:

**elementLocation**

This parameter is used to match elements. For this type of parameter, the application displays a text field where you can enter the element name or an XPath expression. The text from the `@label` attribute is displayed in the application as the label of the text field. The `@name` attribute is used to specify the name of the parameter from the XQuery Update script or XSLT stylesheet. If the value of the `@useCurrentContext` attribute is set to `true`, the element name from the cursor position is used as proposed values for this parameter.

Example of an `<elementLocation>`:

```xml
<elementLocation name="elem_loc" useCurrentContext="false">
  <element label="Element location">
    <description>Element location description.</description>
  </element>
</elementLocation>
```

**attributeLocation**

This parameter is used to match attributes. For this type of parameter, the application displays two text fields where you can enter the parent element name and the attribute name (both text fields accept XPath expressions for a finer match). The text from the `@label` attributes is displayed in the application as the label of the associated text fields. The `@name` attribute is used to specify the name of the parameter from the XQuery Update script or XSLT stylesheet. The value of this parameter is an XPath expression that is computed by using the values of the expression from the `element` and `attribute` text fields. For example, if `section` is entered for the element and a `title` is entered for the attribute, the XPath expression would be computed as `//section/@title`. If the value of the `useCurrentContext` attribute is set to `true`, the element and attribute name from the cursor position is used as proposed values for the operation parameters.

Example of an `<attributeLocation>`:

```xml
<attributeLocation name="attr_xpath" useCurrentContext="true">
  <element label="Element path">
    <description>Element path description.</description>
  </element>
  <attribute label="Attribute">
    <description>Attribute path description.</description>
  </attribute>
</attributeLocation>
```
**elementParameter**

This parameter is used to specify elements by local name and namespace. For this type of parameter, the application displays two combo boxes with elements and namespaces collected from the associated schema of the currently edited file. The text from the @label attribute is displayed in the application as label of the associated combo. The @name attribute is used to specify the name of the parameter from the XQuery Update script or XSLT stylesheet. If you specify the @allowsAny attribute, the application will propose <ANY> as a possible value for the Name and Namespace combo boxes. You can also use the @useCurrentContext attribute and if its value is set to true, the element name and namespace from the cursor position is used as proposed values for the operation parameters.

**Example of an elementParameter:**

```xml
<elementParameter id="elemID" useCurrentContext="true">
  <localName label="Name" name="element_localName" allowsAny="true">
    <description>Local name of the parent element.</description>
  </localName>
  <namespace label="Namespace" name="element_namespace" allowsAny="true">
    <description>Local name of the parent element</description>
  </namespace>
</elementParameter>
```

**attributeParameter**

This parameter is used to specify attributes by local name and namespace. For this type of parameter, the application displays two combo boxes with attributes and their namespaces collected from the associated schema of the currently edited file. The text from the @label attribute is displayed in the application as the label of the associated combo box. You can also use the @useCurrentContext attribute and if its value is set to true, the attribute name and namespace from the cursor position is used as proposed values for the operation parameters.

**Note:**

An `attributeParameter` is dependant upon an `elementParameter`. The list of attributes and namespaces are computed based on the selection in the `elementParameter` combo boxes.

**Example of an attributeParameter:**

```xml
<attributeParameter dependsOn="elemID" useCurrentContext="true">
  <localName label="Name" name="attribute_localName">
    <description>The name of the attribute to be converted.</description>
  </localName>
  <namespace label="Namespace" name="attribute_namespace" allowsAny="true">
    <description>Namespace of the attribute to be converted.</description>
  </namespace>
</attributeParameter>
```
Note:
All built-in operations are loaded from the 
\[OXYGEN_INSTALL_DIR]/refactoring folder.

Related information
Example of an Operation Descriptor File with an XSLT Stylesheet (on page 871)
Example of an Operation Descriptor File with an XQuery Update script (on page 866)

**XSLT Stylesheet for Creating a Custom Operation**

To demonstrate creating a custom operation, suppose that you have a task where you need to convert an attribute into an element and insert it inside another element. A specific example would be if you have a project with a variety of `<image>` elements where a deprecated `@alt` attribute was used for the description and you want to convert all instances of that attribute into an element with the same name and insert it as the first child of the `<image>` element.

Thus, the task is to convert this attribute into an element with the same name and insert it as the first child of the image element.

**Figure 622. Example: Custom XML Refactoring Operation**

An XSLT stylesheet can be used to implement the new custom XML refactoring operation. The second requirement is an XML Refactoring operation descriptor file (on page 2605) that contains the path to the XSLT stylesheet.
Example of an XSLT Script for Creating a Custom Operation to *Convert an Attribute to an Element*

The XSLT stylesheet does the following:

- Iterates over all elements from the document that have the specified local name and namespace.
- Finds the attribute that will be converted to an element.
- Adds the new element as the first child of the parent element.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
                   xmlns:xs="http://www.w3.org/2001/XMLSchema"
                   exclude-result-prefixes="xs"
                   xmlns:xr="http://www.oxygenxml.com/ns/xmlRefactoring"
                   version="2.0">


  <xsl:param name="element_localName" as="xs:string" required="yes"/>
  <xsl:param name="element_namespace" as="xs:string" required="yes"/>
  <xsl:param name="attribute_localName" as="xs:string" required="yes"/>
  <xsl:param name="attribute_namespace" as="xs:string" required="yes"/>
  <xsl:param name="new_element_localName" as="xs:string" required="yes"/>
  <xsl:param name="new_element_namespace" as="xs:string" required="yes"/>

  <xsl:template match="node() | @*">
    <xsl:copy>
      <xsl:apply-templates select="node() | @*"/>
    </xsl:copy>
  </xsl:template>

  <xsl:template match="/*[xr:check-local-name($element_localName, ., true())
                           and
                           xr:check-namespace-uri($element_namespace, .)]">
    <xsl:variable name="attributeToConvert"
                  select="@*[xr:check-local-name($attribute_localName, ., true())
                         and
                         xr:check-namespace-uri($attribute_namespace, .)]"/>

    <xsl:choose>
      <xsl:when test="empty($attributeToConvert)">
        <xsl:copy>$attributeToConvert</xsl:copy>
      </xsl:when>
      <xsl:otherwise>
        <xsl:apply-templates select="node() | @*"/>
      </xsl:otherwise>
    </xsl:choose>
  </xsl:template>
</xsl:stylesheet>
```
Note:
The XSLT stylesheet imports a module library that contains utility functions and variables. The location of this module is resolved via an XML Catalog (on page 3225) set in the XML Refactoring framework (on page 3219).

Example of an Operation Descriptor File That References the XSLT Stylesheet for Creating a Custom Operation to Convert an Attribute to an Element

After you have developed the XSLT stylesheet, you have to create an XML Refactoring operation descriptor. This descriptor is used by the application to load the operation details such as name, description, or parameters.

<?xml version="1.0" encoding="UTF-8"?>
<refactoringOperationDescriptor>
<description>Converts the specified attribute to an element. The new element will be inserted as first child of the attribute's parent element.</description>

<!-- For the XSLT stylesheet option uncomment the following line and comment the line referring the XQuery Update script -->

<!-- <script type="XSLT" href="convert-attribute-to-element.xsl"/> -->

<script type="XQUERY_UPDATE" href="convert-attribute-to-element.xq"/>

<parameters>

<description>Specify the attribute to be converted to element.</description>

<section label="Parent element">

<elementParameter id="elemID">

<localName label="Name" name="element_localName" allowsAny="true">

<description>Local name of the parent element.</description>

</localName>

<namespace label="Namespace" name="element_namespace" allowsAny="true">

<description>Local name of the parent element</description>

</namespace>

</elementParameter>

</section>

<section label="Attribute">

<attributeParameter dependsOn="elemID">

<localName label="Name" name="attribute_localName">

<description>Name of the attribute to be converted.</description>

</localName>

<namespace label="Namespace" name="attribute_namespace" allowsAny="true">

<description>Namespace of the attribute to be converted.</description>

</namespace>

</attributeParameter>

</section>

<section label="New element">

<elementParameter>

<localName label="Name" name="new_element_localName">

<description>The name of the new element.</description>

</localName>

<namespace label="Namespace" name="new_element_namespace">

<description>The namespace of the new element.</description>

</namespace>

</elementParameter>

</section>
Note:

If you are using an XSLT file, the line with the `<script>` element would look like this:

```xml
<script type="XSLT" href="convert-attribute-to-element.xsl"/>
```

The code exemplified above and other refactoring examples can be found on the DITA Refactoring GitHub sample project.

Results

After you have created these files, copy them into a folder scanned by Oxygen XML Editor when it loads the custom operation (on page 875). When the XML Refactoring tool is started again, you will see the created operation.

Since various parameters can be specified, this custom operation can also be used for other similar tasks. The following image shows the parameters that can be specified in the example of the custom operation to convert an attribute to an element:

![Figure 623. Example: XML Refactoring Wizard for a Custom Operation](image)

Using Saxon Extension Functions to Allow Custom Refactoring Operations to Read and Modify Content Outside the Root Node

One advantage to using an XSLT stylesheet is that there is limitation when using an XQuery Update script (on page 863) in that refactoring operations can only be performed on comments or processing instructions that
are inside the root element. Thus, using the XQuery method, *comments or processing instructions* that are in any node before or after the root element cannot be modified by an XML Refactoring operation.

The XSLT stylesheet method offers a work-around to this limitation through the use of some Saxon extension functions.

To illustrate the use of these functions, consider the following sample XML file:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<!-- comment before root -->
<?pi before root ?>
<root>
    <child/>
</root>
<!-- comment after root -->
<?pi after root ?>
```

The following Saxon extension functions can be used to read and modify content outside the root node:

- **get-content-after-root()** - Returns the content after root as `xs:string`.
- **set-content-after-root(xs:string)** - Updates the content that will be serialized in the refactored document after the root node.
- **get-content-before-root()** - Returns the content before root as `xs:string`.

For the XML above, the call of this function will return the following string value:

```xml
<!-- comment after root -->
<?pi after root ?>
```

The function call `set-content-after-root('<!-- Inserted comment -->')` will result in replacing the nodes after the root element with the comment passed as string argument. The XML document will be modified as follows:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<!-- comment before root -->
<?pi before root ?>
<root>
    <child/>
</root>
<!-- Inserted comment -->
```

For the XML above, the call of this function will return the following string value:
• **set-content-before-root(xs:string)** - Updates the content that will be serialized in the refactored document after the root node.

The function call `set-content-before-root('<!-- Inserted comment -->')` will result in replacing the nodes before the root element with the comment passed as string argument. The XML document will be modified as follows:

```xml
<!-- Inserted comment -->
<root>
  <child>
</child>
</root>
</?pi after root ?>
```

**XSLT Example:**

To process content after the root node, the XSLT would look like this:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  xmlns:xs="http://www.w3.org/2001/XMLSchema" exclude-result-prefixes="xs"
  xmlns:xrf="http://www.oxygenxml.com/ns/xmlRefactoring/functions" version="3.0">
  <xsl:template match="/">
    <!-- The comment content that will be inserted after the root element -->
    <xsl:variable name="commentAsText"><!-- COMMENT ADDED FROM XR OPERATION-->
    </xsl:variable>
    <!-- Retrieve the content after the root element as is -->
    <xsl:variable name="after-root-content" as="xs:string"
      select="xrf:get-content-after-root()"/>
    <!-- Update the content after the root element -->
    <xsl:value-of select="xrf:set-content-after-root($processedContent)"/>
  </xsl:template>
</xsl:stylesheet>
```
Note:
The above XSLT retrieves the nodes after the root element as string, appends a new comment, and then sets back the updated content into the XML document.

Storing and Sharing Refactoring Operations

Oxygen XML Editor scans the following locations when looking for XML Refactoring operations to provide flexibility:

- A refactoring folder, created inside a directory that is associated to a framework you are customizing. To associate the parent directory to the framework, you need to add it in the Classpath tab of the Document type configuration dialog box (on page 144).
- A folder that you specify in the Load additional refactoring operations from text box (on page 269) in the XML Refactoring preferences page (on page 269).

Note:
If you share a project with your team, you can also share the custom operation by doing the following:
1. Save the custom operation in a folder that is part of your project.
2. Switch the XML Refactoring option page to project level (on page 3222):
   a. Open the Preferences dialog box (Options > Preferences) (on page 124) and go to XML > XML Refactoring (on page 269).
   b. Select Project Options (on page 3222) at the bottom of the dialog box.
3. In the Load additional refactoring operations from text box (on page 269), use the ${pd} editor variable (on page 331) so that the folder path is declared relative to the project.

- A folder specified by the XML Refactoring Operations Plugin Extension (on page 2432).
- The refactoring folder from the Oxygen XML Editor installation directory ([OXYGEN_INSTALL_DIR]/refactoring/).

Sharing Custom Refactoring Operations

The purpose of Oxygen XML Editor scanning multiple locations for the XML Refactoring operations is to provide more flexibility for developers who want to share the refactoring operations with the other team members. Depending on your particular use case, you can attach the custom refactoring operations to other resources, such as framework (on page 3219) or projects.

After storing custom operations, you can share them with other users by sharing the resources.
Localizing XML Refactoring Operations

Oxygen XML Editor includes localization support for the XML refactoring operations.

The translation keys for the built-in refactoring operations are located in 
{OXYGEN_INSTALL_DIR}/refactoring/i18n/translation.xml.

The localization support is also available for custom refactoring operations. The following information can be translated:

- The operation name, description, and category.
- The \textit{<description>} of the \textit{<parameters>} element.
- The label, description, and possibleValues for each parameter.

Translated refactoring information uses the following form:

\$\{i18n(translation_key)\}

Oxygen XML Editor scans the following locations to find the translation.xml files that are used to load the translation keys:

- A \texttt{refactoring/i18n} folder, created inside a directory that is associated to a customized \textit{framework}.
- A \texttt{i18n} folder, created inside a directory that is associated to a customized \textit{framework}.
- An \texttt{i18n} folder inside any specified folder. In this case, you need to open the Preferences dialog box (Options > Preferences) (on page 124), go to XML > XML Refactoring, and specify the folder in the Load additional refactoring operations from text box.
- An \texttt{i18n} folder located in directories specified through the XML Refactoring Operations Plugin Extension (on page 2432).
- The \texttt{refactoring/i18n} folder from the Oxygen XML Editor installation directory (/OXYGEN_INSTALL_DIR/refactoring/i18n).

\textbf{Example: Refactoring Operation Descriptor File with i18n Support}

```xml
<?xml version="1.0" encoding="UTF-8"?>

<refactoringOperationDescriptor
   xmlns="http://www.oxygenxml.com/ns/xmlRefactoring"
   id="remove_text_content"
   name="\$\{i18n(Remove_text_content)\}"
>
   <description>\$\{i18n(Remove_text_content_description)\}</description>

   <script type="XQUERY_UPDATE" href="remove_text_content.xq"/>

   <parameters>
       <description>\$\{i18n(parameters_description)\}</description>

       <parameter
           label="\$\{i18n(Element_name)\}" name="element_localName"
           type="NC_NAME">
           <description>\$\{i18n(Element_name_descriptor)\}</description>
       </parameter>

       <possibleValues>
       </possibleValues>
   </parameters>
</refactoringOperationDescriptor>
```
Generating Sample XML Files

Oxygen XML Editor offers support to generate sample XML files both from XML schema 1.0 and XML schema 1.1, depending on the XML schema version set in XML Schema preferences page (on page 238).

To generate sample XML files from an XML Schema, use the Generate Sample XML Files action from the Tools menu. This action is also available in the contextual menu of the schema Design mode (on page 952). The action opens the Generate Sample XML Files dialog box that allows you to configure a variety of options for generating the files.

The Generate Sample XML Files dialog box contains three tabs with various configurable options. Default values for these options can be set in the Sample XML Files Generator preferences page (on page 242). You can also run the tool from the command line using exported options.

Schema Tab

The first set of options for the Generate Sample XML Files tool are found in the Schema tab.
Figure 624. Generate Sample XML Files Dialog Box (Schema Tab)

This tab includes the following options:

**URL**

Specifies the URL of the Schema location. You can specify the path by using the text field, the history drop-down menu, or the browsing actions in the **Browse** drop-down list.

**Namespace**

Displays the namespace of the selected schema.

**Root Element**

After the schema is selected, this drop-down menu is populated with all root candidates gathered from the schema. Choose the root of the output XML documents.

**Output folder**

Path to the folder where the generated XML instances will be saved.

**Filename prefix and Extension**

You can specify the prefix and extension for the file name that will be generated. Generated file names have the following format: prefixN.extension, where \( N \) represents an incremental number from 0 up to the specified **Number of instances**.

**Number of instances**

The number of XML files to be generated.
Open first instance in editor

When selected, the first generated XML file is opened in the editor.

Namespaces section

You can specify the Default Namespace, as well as the prefixes for the namespaces.

Export settings

Use this button to save the current settings for future use.

Import settings

Use this button to load previously exported settings.

You can click OK at any point to generate the sample XML files.

Options Tab

The Options tab allows you to set specific options for namespaces and elements.

Figure 625. Generate Sample XML Files Dialog Box (Options Tab)

This tab includes the following options:
Namespace / Element table

Allows you to set a namespace for each element name that appears in an XML document instance. The following prefix-to-namespace associations are available:

- All elements from all namespaces (<ANY> - <ANY>). This is the default setting.
- All elements from a specific namespace.
- A specific element from a specific namespace.

Settings subtab

Namespace
Displays the namespace specified in the table at the top of the dialog box.

Element
Displays the element specified in the table at the top of the dialog box.

Generate optional elements
When selected, all elements are generated, including the optional ones (having the minOccurs attribute set to 0 in the schema).

Generate optional attributes
When selected, all attributes are generated, including the optional ones (having the use attribute set to optional in the schema).

Values of elements and attributes
Controls the content of generated attribute and element values. The following choices are available:

- **None** - No content is inserted.
- **Default** - Inserts a default value depending on the data type descriptor of the particular element or attribute. The default value can be either the data type name or an incremental name of the attribute or element (according to the global option from the Sample XML Files Generator preferences page). Note that type restrictions are ignored when this option is selected. For example, if an element is of a type that restricts an xs:string with the xs:maxLength facet to allow strings with a maximum length of 3, the XML instance generator tool may generate string element values longer than 3 characters.
- **Random** - Inserts a random value depending on the data type descriptor of the particular element or attribute.

⚠️ Important:
If all of the following are true, the Generate Sample XML Files tool outputs invalid values:
Preferred number of repetitions

Allows you to set the preferred number of repeating elements related to `minOccurs` and `maxOccurs` facets defined in the XML Schema.

- If the value set here is between `minOccurs` and `maxOccurs`, then that value is used.
- If the value set here is less than `minOccurs`, then the `minOccurs` value is used.
- If the value set here is greater than `maxOccurs`, then `maxOccurs` is used.

Maximum recursion level

If a recursion is found, this option controls the maximum allowed depth of the same element.

Type alternative strategy

Used for the `<xs:alternative>` element from XML Schema 1.1. The possible strategies are:

- **First** - The first valid alternative type is always used.
- **Random** - A random alternative type is used.

Choice strategy

Used for `<xs:choice>` or `<substitutionGroup>` elements. The possible strategies are:

- **First** - The first branch of `<xs:choice>` or the head element of `<substitutionGroup>` is always used.
- **Random** - A random branch of `<xs:choice>` or a substitute element or the head element of a `<substitutionGroup>` is used.

Generate the other options as comments

If selected, generates the other possible choices or substitutions (for `<xs:choice>` and `<substitutionGroup>`). These alternatives are generated inside comments groups so you can uncomment and use them later. Use this option with care (for example, on a restricted namespace and element) as it may generate large result files.

Element values subtab

Allows you to add values that are used to generate the content of elements. If there are multiple values, then the values are used in a random order.

Attribute values subtab
Allows you to add values that are used to generate the content of attributes. If there are multiple values, then the values are used in a random order.

**Export settings**

Use this button to save the current settings for future use.

**Import settings**

Use this button to load previously exported settings.

You can click **OK** at any point to generate the sample XML files.

**Advanced Tab**

The **Advanced** tab allows you to set some options regarding output values and performance.

**Figure 626. Generate Sample XML Files Dialog Box (Advanced Tab)**

This tab includes the following options:

**Use incremental attribute / element names as default**

If selected, the value of an element or attribute starts with the name of that element or attribute. For example, for an `<a>` element the generated values are: `a1, a2, a3`, and so on. If not selected, the value is the name of the type of that element / attribute (for example: `string, decimal`, etc.)

**Maximum length**

The maximum length of string values generated for elements and attributes.

**Discard optional elements after nested level**

The optional elements that exceed the specified nested level are discarded. This option is useful for limiting deeply nested element definitions that can quickly result in very large XML documents.

**Export settings**

Use this button to save the current settings for future use.

**Import settings**

Use this button to load previously exported settings.
Converting Schema to Another Schema Language

The **Generate/Convert Schema** tool allows you to convert a DTD or Relax NG (full or compact syntax) schema or a set of XML files to an equivalent XML Schema, DTD or Relax NG (full or compact syntax) schema. Where perfect equivalence is not possible due to limitations of the target language, Oxygen XML Editor generates an approximation of the source schema. Oxygen XML Editor uses the **Trang multiple format converter** to perform the actual schema conversions.

To use this tool, select the **Generate/Convert Schema (Alt + Shift + C (Command + Alt + C on macOS))** action from the **Tools** menu or from the **Open with** submenu when invoking the contextual menu in the **Project view (on page 402)**. This action opens the **Generate/Convert Schema** dialog box that allows you to configure various options for conversion.

### Figure 627. Generate/Convert Schema Dialog Box

The **Generate/Convert Schema** dialog box includes the following options:

**Input section**
Allows you to select the language of the source schema. If the conversion is based on a set of XML files, rather than just a single XML file, select the **XML Documents** option and use the file selector to add the XML files involved in the conversion.

**Output section**

Allows you to select the language of the target schema.

**Options**

You can choose the **Encoding**, the maximum **Line width**, and the **Indent size** (in number of spaces) for one level of indentation.

**Output file**

Specifies the path for the output file that will be generated.

**Close dialog when finished**

If you deselect this option, the dialog box will remain open after the conversion so that you can easily continue to convert more files.

**Advanced options**

If you select **XML 1.0 DTD** for the input, you can click this button to access more advance options to further fine-tune the conversion. The following advanced options are available:

**XML 1.0 DTD Input section**

These options apply to the source DTD:

- **xmlns** - Specifies the default namespace, that is the namespace used for unqualified element names.
- **attlist-define** - Specifies how to construct the name of the definition representing an attribute list declaration from the name of the element. The specified value must contain exactly one percent character. This percent character is replaced by the name of element (after colon replacement) and the result is used as the name of the definition.
- **colon-replacement** - Replaces colons in element names with the specified chars when constructing the names of definitions used to represent the element declarations and attribute list declarations in the DTD.
- **any-name** - Specifies the name of the definition generated for the content of elements declared in the DTD as having a content model of ANY.
- **element-define** - Specifies how to construct the name of the definition representing an element declaration from the name of the element. The specified value must contain exactly one percent character. This percent character is replaced by the name of element (after colon replacement) and the result is used as the name of the definition.
- **annotation-prefix** - Default values are represented using a `@prefix:defaultValue` annotation attribute where prefix is the specified value
and is bound to http://relaxng.org/ns/compatibility/annotations/1.0 as defined by the RELAX NG DTD Compatibility Committee Specification. By default, the conversion engine will use a for prefix unless that conflicts with a prefix used in the DTD.

- **inline-attlist** - Instructs the application not to generate definitions for attribute list declarations, but instead move attributes declared in attribute list declarations into the definitions generated for element declarations. This is the default behavior when the output language is XSD.

- **strict-any** - Preserves the exact semantics of ANY content models by using an explicit choice of references to all declared elements. By default, the conversion engine uses a wildcard that allows any element

- **generate-start** - Specifies whether or not the conversion engine should generate a start element. DTD's do not indicate what elements are allowed as document elements. The conversion engine assumes that all elements that are defined but never referenced are allowed as document elements.

- **xmlns mappings** table - Each row specifies the prefix used for a namespace in the input schema.

### W3C XML Schema Output section

This section is available if you select **W3C XML Schema** for the output.

- **disable-abstract-elements** - Disables the use of abstract elements and substitution groups in the generated XML Schema. This can also be controlled using an annotation attribute.

- **any-process-contents** - One of the values: strict, lax, skip. Specifies the value for the @processContents attribute of any elements. The default is skip (corresponding to RELAX NG semantics) unless the input format is DTD, in which case the default is strict (corresponding to DTD semantics).

- **any-attribute-process-contents** - Specifies the value for the @processContents attribute of &lt;anyAttribute&gt; elements. The default is skip (corresponding to RELAX NG semantics).

### Converting Database to XML Schema

Oxygen XML Editor includes a tool that allows you to create an XML Schema from the structure of a database.

To convert a database structure to an XML Schema, use the following procedure:

1. Select the **Convert DB Structure to XML Schema** action from the **Tools** menu.

   **Result:** The **Convert DB Structure to XML Schema** dialog box is opened and your current database connections are displayed in the **Connections** section.

2. If the database source is not listed, click the **Configure Database Sources** button to open the **Data Sources preferences page** where you can configure data sources and connections.
3. In the **Format for generated schema** section, select one of the following formats:
   - **Flat schema** - A flat structure that resembles a tree-like view of the database without references to elements.
   - **Hierarchical schema** - Display the table dependencies visually, in a type of tree view where dependent tables are shown as indented child elements in the content model. Select this option if you want to configure the database columns of the tables to be converted.

4. Click **Connect**.

   **Result:** The database structure is listed in the **Select database tables** section according to the format you chose.

5. Select the database tables that you want to be included in the XML Schema.

6. If you selected **Hierarchical schema** for the format, you can configure the database columns.
   a. Select the database column you want to configure.
   b. In the **Criterion** section you can choose to convert the selected database column as an **Element**, **Attribute**, or to be **Skipped** in the resulting XML Schema.
   c. You can also change the name of the selected database column by changing it in the **Name** text field.

7. Click **Generate XML Schema**.

   **Result:** The database structure is converted to an XML Schema and it is opened for viewing and editing.

### Flatten an XML Schema

You can organize an XML schema linked by `<xs:include>` and `<xs:import>` statements on several levels. In some cases, working on such a schema as if it were a single file is more convenient than working on multiple files separately. The **Flatten Schema** operation allows you to flatten an entire hierarchy of XML schemas. Starting with the main XML schema, Oxygen XML Editor calculates its hierarchy by processing the `<xs:include>` and `<xs:import>` statements.

The **Flatten Schema** action is available from the **Tools** menu or the contextual menu in **Text** mode. This action opens the **Flatten Schema** dialog box that allows you to configure the operation.
For the main schema file and for each imported schema, a new flattened schema is generated in the specified output folder. These schemas have the same name as the original ones.

**Note:**
If necessary, the operation renames the resulted schemas to avoid duplicated file names.

A flattened XML schema is obtained by recursively adding the components of the included schemas into the main one. This means Oxygen XML Editor replaces the `<xs:include>`, `<xs:redefine>`, and `<xs:override>` elements with the ones coming from the included files.

**Options in the Flatten Schema Dialog Box**

The following options are available in the **Flatten Schema** dialog box:

- **File name**
  The name of the output file.

- **Output directory**
  The path of the output directory where the flattened schema file will be saved.

- **Open the flattened XML Schema file in editor**
Opens the main flattened schema in the editing area after the operation completes.

**Use the XML Catalogs when collecting the referenced XML Schemas**

Enables the imported and included schemas to be resolved through the available XML Catalogs (on page 3225).

**Note:**
Changing this option triggers the recalculation of the dependencies graph for the main schema.

**Process the imported XML Schemas resolved through the XML Catalogs**

Specifies whether or not the imported schemas that were resolved through an XML Catalog (on page 3225) are also processed.

**Flatten the imported XML Schema(s)**

Specifies whether or not the imported schemas are flattened.

**Note:**
For the schemas skipped by the flatten operation, no files are created in the output folder and the corresponding import statements remain unchanged.

**Tip:**
This function can be executed from an automated command-line script, for more details, see Scripting Oxygen (on page 3196).

**Generating Java Classes from XML Schema**

Oxygen XML Editor includes a tool for generating Java classes from an XML Schema (XSD) file. The Generate Java classes from XML Schema (XSD) action for invoking the tool can be found in the Tools menu. It requires an additional add-on to be installed, so the first time you invoke the action, Oxygen XML Editor will present a dialog box asking if you want to install it. Once installed, you need to restart Oxygen XML Editor and the action will invoke the Java class generator tool.

**Quick Installation**

You can drag the following Install button and drop it into the main editor in Oxygen (version 24.1 or newer) to quickly initiate the installation process:

Install

**Manual Installation**

To manually install it the add-on, follow these instructions:
1. Go to Help > Install new add-ons to open an add-on selection dialog box.
2. Enter or paste https://www.oxygenxml.com/InstData/Addons/default/updateSite.xml in the Show add-ons from field or select it from the drop-down menu.
3. Select the Generate Java classes from XML Schema (XSD) add-on and click Next.
4. Read the end-user license agreement. Then select the I accept all terms of the end-user license agreement option and click Finish.
5. Restart the application.

Result: The Generate Java classes from XML Schema (XSD) dialog box is now available and can be selected from the Tools menu.

Generating Java Classes

To generate Java classes, follow these steps:

1. Select the Generate Java classes from XML Schema (XSD) action from the Tools menu.

   Step Result: The Generate Java classes from XML Schema (XSD) dialog box is displayed:

   **Figure 629. Generate Java Classes from XML Schema (XSD) Dialog Box**

   ![Generate Java classes from XML Schema (XSD) Dialog Box]

2. Choose or enter the XSD URL of the XML Schema document.
3. Choose the path for the Output folder where the generated files will be stored.
4. [Optional] You can choose the Package name for the Java package that will contain the generated source files. If not specified, the name will be generated based on the value of the @targetNamespace attribute.
5. [Optional] You can select the Open in Editor option to open the ObjectFactory.java file in the editor. This java class contains factory methods for all other classes in the package.
6. Click the Generate button.

Result: The Java class files will be generated inside the new package, located in the specified output folder.
Compiling an XSL Stylesheet for Saxon

As of Saxon 10.6, it is possible to export a compiled form of a stylesheet as a JSON or XML file (called a stylesheet export file or SEF). Oxygen XML Editor includes a simple tool called Compile XSL Stylesheet for Saxon (found in the Tools menu) that does this for you.

Use-Cases for a Stylesheet Export File (SEF)

- **Use Saxon-JS to run transformations in a browser** - A stylesheet export file (SEF) is needed if you want to use the Saxon-JS product to run transformations in a browser, as in the following example:

```html
<script type="text/javascript" src="SaxonJS/SaxonJS.min.js"></script>
<script>
    window.onload = function () {
        SaxonJS.transform({
            stylesheetLocation: "books.sef",
            sourceLocation: "books.xml"
        });
    }
</script>
```

- **Use SEF to run transformations in Oxygen XML Editor** - You can also use a stylesheet export file (SEF) in Oxygen XML Editor to apply an XSLT transformation over an XML file. This requires Saxon-EE or Saxon-PE versions of the Saxon product and you must select one of those two versions for the Target when you configure the SEF file (on page 2626). When configuring the XSLT transformation, you will specify the SEF file in the XSL URL field (on page 1465).

Compiling an SEF File

The Compile XSL Stylesheet for Saxon tool can be found in the Tools menu and it compiles a specified stylesheet as a JSON or an XML file (stylesheet export file).

If you choose Saxon-JS as the Target (the type of Saxon product that the export file will be used with), then the compiled stylesheet will be a JSON file with a file extension of .sef by default.

If you choose Saxon-EE, Saxon-PE, or Saxon-HE for the Target, then the compiled stylesheet will be an XML file with a file extension of .xsef by default.

Selecting this tool opens the Compile XSL Stylesheet for Saxon dialog box that allows you to configure some options for conversion.
This dialog box includes the following options:

**XSL URL**

Allows you to select URL of the source XSL stylesheet. You can specify the URL by using the text field, the history drop-down, or the browsing actions in the **Browse** drop-down list.

**Output file**

You can specify the path where the output file will be saved by entering it in the text field, using the **Insert Editor Variables** button, or using the browsing actions in the **Browse** drop-down list.

**Open in Editor**

Select this option to open the resulting **stylesheet export file** in the main Oxygen XML Editor editing pane.

**Target**

Allows you to select the type of Saxon product that the export file will be used with. You can choose **Saxon-JS, Saxon-EE, Saxon-PE, or Saxon-HE.**

**Relocatable**

Can be used to control the Saxon **-relocate** parameter. You can select this option to produce a **relocatable** export package (SEF) that can be deployed to a different location, with a different base URI.

**Set the default namespace for unprefixed element names ("-ns")**
Can be used to control the $ns:(uri|##any|##html5) Saxon parameter that defines the handling of unprefixed element names that appear as name tests in path expressions and match patterns in the stylesheet:

- The ##any value declares that unprefixed names are treated as a test on the local name of the element only. They will match regardless of namespace.
- The ##html5 value declares that an unprefixed element name will match either a name in the XHTML namespace or a name in no namespace. This option is primarily intended for use when generating stylesheets to run under Saxon-JS in the browser since the resulting behavior is close to that defined by the special rules in the HTML5 specification for XSLT and XPath running against an HTML5 DOM.
- You can also specify a valid URI by editing the value in the combo box. Specifying a URI sets the default namespace for elements and types (effectively a default value for xpath-default-namespace). Note that an explicit value for this attribute takes precedence.

Use a configuration file (“-config”)

Select this option if you want to use a Saxon 10.6 configuration file that will be executed for the XSLT transformation and validation processes. You can specify the path to the configuration file by entering it in the URL field, or by using the Insert Editor Variables button, or using the browsing actions in the Browse drop-down list.

Compile

Use this button to generate the stylesheet export file according the options selected in this dialog box.

JSON Tools

Oxygen XML Editor includes some useful tools for converting JSON to and from XML, converting XSD to JSON Schema, and generating JSON instances or a JSON Schema.

Generating Sample JSON Files from a JSON Schema

Oxygen XML Editor includes a tool for generating sample JSON files. To generate sample JSON files from a JSON Schema, select Generate Sample JSON Files from the Tools > JSON Tools menu. The action opens a dialog box where you can configure a variety of options for generating the files.
The **Generate Sample JSON Files** dialog box includes the following fields and options:

### Schema URL

The URL of the Schema location. You can specify the path by using the text field, the history drop-down menu, or the browsing actions in the `Browse` drop-down list.

### Associate schema in the document

If enabled, the specified schema will be associated with the generated files.

### Output folder

Path to the folder where the generated JSON instances will be saved.

### File name

The name of the instance(s) that will be generated. By default, `instance.json` is used.

### Number of instances

The desired number of JSON instances to be generated. When more than one instance is generated, the index of the instance will be added to its file name.

### Property value

You can specify the way the values of the properties are generated. The following options are available:
• *None* - Assigns empty values for properties (a template file will be generated). This is the default value.
• *Default* - Assigns the name of the property as the value (for strings) or assigns the specified minimum value (for numbers).
• *Random* - Assigns random values according to schema restrictions.

**Generate optional properties**

If selected, the JSON instance will be generated with optional properties that are defined in the JSON schema. Otherwise, only the required properties will be generated.

**Generate additional content**

If selected, the JSON instance will be generated with additional properties that are defined in the JSON schema as `additionalProperties` and additional items that are defined as `additionalItems` (in the case of an Array).

**Choice strategy**

You can specify the way an instance will be generated from a schema that contains a `CombinedSchema` (with either `oneOf` or `anyOf`). The following options are available:

• *First* - The first defined schema in `oneOf` or `anyOf` will be used.
• *Random* - A random schema defined in `oneOf` or `anyOf` will be used.

**Recursion level**

This option controls the maximum allowed depth (must be a number), in case the selected schema contains recursive calls of `@ref` schemas referencing one another. By default, it is set to 1, meaning that the generation for the recursive calls will stop after the first iteration.

**Open first instance in editor**

If selected, the first generated instance is opened in the editor.

You can click **OK** at any point to generate the sample JSON files.

**Generating JSON Schema from a JSON File**

Oxygen XML Editor includes a tool for generating a sample JSON Schema from a JSON file. To generate a sample JSON Schema, select **Generate JSON Schema** from the **Tools > JSON Tools** menu. The action opens a dialog box where you can configure some options for generating the JSON Schema.
Figure 632. Generate JSON Schema Dialog Box

The **Generate JSON Schema** dialog box includes the following fields and options:

**JSON Document URL**

The URL of the JSON file. You can specify the path by using the text field, the history drop-down menu, or the browsing actions in the **Browse** drop-down list.

**Output JSON Schema**

The path to the folder where the generated JSON Schema will be saved.

**Open in Editor**

If selected, the generated JSON Schema is opened in the editor.

**JSON Schema version**

The version of the resulting JSON schema. The possible choices are: **Draft 4**, **Draft 6**, **Draft 7**, and **Draft 2019-09**.

**Extract matching format for strings**

If selected, the generator will attempt to find a format that matches the string values from the JSON Document.

**Restrict additional content**

If selected, **additionalProperties** (for objects) and **additionalItems** (for arrays) will be set to **false** in the resulting schema. By default, these keys are not in the schema, meaning that providing additional content (according to the schema) is allowed.

**Add default values for simple types**

If selected, the **default** values (0 for number, "" for string, **false** for boolean) and **examples** for strings will be added.
**Make all properties required**

If selected, the generator will mark all the properties as required in the resulting schema.

You can click **Generate** at any point to generate the JSON Schema.

**JSON to YAML Converter**

**Converting JSON to YAML in Oxygen**

Oxygen XML Editor includes a useful and simple tool for converting JSON files to YAML. The **JSON to YAML** action for invoking the tool can be found in the **Tools > JSON Tools** menu.

To convert a JSON document to YAML, follow these steps:

1. Select the **JSON to YAML** action from the **Tools > JSON Tools** menu.
   
   The **JSON to YAML** dialog box is displayed:

   ![JSON to YAML Dialog Box]

   2. Choose or enter the **JSON URL** for the document you want to convert.
   
   3. Choose the path of the **Output file** that will contain the resulting YAML document.
   
   4. **[Optional]** Select the **Open in Editor** option to open the resulting YAML document in the main editing pane.
   
   5. Click the **Convert** button.

**Result**: The original JSON document is now converted to a YAML document.

**Related Information:**

YAML to JSON Converter (on page 1137)
YAML to JSON Converter

Converting YAML to JSON in Oxygen

Oxygen XML Editor includes a useful and simple tool for converting YAML files to JSON. It even works on files that consist of multiple YAML documents, each separated by three dashes (---), in which case the conversion creates multiple JSON files with a number in the name.

The YAML to JSON action for invoking the tool can be found in the Tools > JSON Tools menu.

To convert a YAML document to JSON, follow these steps:

1. Select the YAML to JSON action from the Tools > JSON Tools menu.

   The YAML to JSON dialog box is displayed:

   ![YAML to JSON Dialog Box](image)

   Figure 634. YAML to JSON Dialog Box

   2. Choose or enter the YAML URL for the document you want to convert.
   3. Choose the path of the Output file that will contain the resulting JSON document.
   4. [Optional] Select the Open in Editor option to open the resulting JSON document in the main editing pane.
   5. Click the Convert button.

Result: The original YAML document is now converted to a JSON document.

Related Information:
- JSON to YAML Converter (on page 1137)

JSON to XML Converter

Online JSON to XML Converter

Attention:

For a simple ONLINE tool for converting a single JSON file to XML, or vice versa, go to: https://www.oxygenxml.com/xml_json_converter.html.
Converting JSON to XML in Oxygen

Oxygen XML Editor includes a useful and simple tool for converting JSON files to XML. The JSON to XML action for invoking the tool can be found in the Tools > JSON Tools menu.

To convert a JSON document to XML, follow these steps:

1. Select the JSON to XML action from the Tools > JSON Tools menu.

   The JSON to XML dialog box is displayed:

   ![JSON to XML Dialog Box](image)

2. Choose or enter the Input URL of the JSON document.
3. Choose the path of the Output file that will contain the resulting XML document.
4. Select the Open in Editor option to open the resulting XML document in the main editing pane.
5. Click the Convert button.

Result: The original JSON document is now converted to an XML document.
Conversion Details

• If the JSON document has more than one property on the first level, the converted XML document will have an additional root element called `<JSON>`.

For example, the following JSON document:

```json
{
    "personnel": {
        "person": [
            {
                "id": "Big.Boss",
                "name": "Boss",
                "given": "Big"
            },
            {
                "name": "Worker"
            }
        ],
        "id": "personnel-id"
    }
}
```

it is converted to:

```xml
<?xml version="1.0" encoding="UTF-8"?>

<JSON>
    <personnel>
        <person>
            <name>Boss</name>
        </person>
    </personnel>
</JSON>
```
• If the JSON document is an array, the converted XML document will have a root element called `<array>` and for each item within the array, another `<array>` is created.

```json
[
  {"name": "Boss"},
  {"name": "Worker"}
]
```

it is converted to:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<array>
  <array>
    <name>Boss</name>
  </array>
  <array>
    <name>Worker</name>
  </array>
</array>
```

• If the name of a JSON property contains characters that are not valid in XML element names (for example, `$`), then the invalid characters will be escaped as its hexadecimal equivalent in the converted XML.

```json
{"$id": "personnel-id"}
```

is converted to:

```xml
<X24_id>personnel-id</X24_id>
```

Related Information:

XML to JSON Converter (on page 1134)
XML to JSON Converter

Online XML to JSON Converter

Attention:

For a simple ONLINE tool for converting a single XML file to JSON, or vice versa, go to: https://www.oxygenxml.com/xml_json_converter.html.

Converting XML to JSON in Oxygen

Oxygen XML Editor includes a useful and simple tool for converting XML files to JSON. The XML to JSON action for invoking the tool can be found in the Tools > JSON Tools menu.

To convert an XML document to JSON, follow these steps:

1. Select the XML to JSON action from the Tools > JSON Tools menu.

   **Step Result:** The XML to JSON dialog box is displayed:

   ![XML to JSON Dialog Box](image)

2. Choose or enter the Input URL of the XML document.
3. Choose the path of the Output file that will contain the resulting JSON document.
4. Select how you want empty elements to be converted (default is object).
5. Select the Open in Editor option to open the resulting JSON document in the main editing pane.
6. Click the Convert button.

**Result:** The original XML document is now converted to a JSON document.
Conversion Details

- Some XML components are ignored (e.g. comments and processing instructions).

- If any elements contain attributes in the XML document, the attributes are converted to properties in the converted JSON document. If the XML document contains more than one element with the same name, they will be converted into an array of object in the converted JSON document.

For example, the following XML document:

```xml
<personnel>
  <person id="person.one">
    <name>Boss</name>
  </person>

  <person id="person.two">
    <name>Worker</name>
  </person>
</personnel>
```

it is converted to:

```json
{  "personnel": {  "person": [  ]  }  }
```
• If the XML document contains elements with mixed content (text plus elements), the converted JSON document will contain a 
<text>property with its value set as the text content. If there are multiple text nodes, the subsequent 
text properties will contain a number (e.g. #text1, #text2). If there are multiple elements with the same name, the first property will have the element name and the subsequent properties will contain a number (e.g. b, b#1, b#2).

This <b>is</b> an <b>example</b>!

is converted to:

```json
{
  "p": {
    "#text": "This ",
    "b": "is",
    "#text1": " an ",
    "b#1": "example",
    "#text2": "!"
  }
}
```

• If the XML document contains element names that contains hexadecimal codes (for example, if they were escaped during a JSON to XML conversion (on page 1131)), it will be converted to the normal character value in the converted JSON document.

<_X24_id>personnel-id</_X24_id>

is converted to:

```json
{"$id": "personnel-id"}
```
XSD to JSON Schema Converter

Oxygen XML Editor includes a tool for converting an XML Schema file (XSD) to a JSON Schema file. The XSD to JSON Schema action for invoking the tool can be found in the Tools > JSON Tools menu. It requires an additional add-on to be installed, so the first time you invoke the action, Oxygen XML Editor will present a dialog box asking if you want to install it. Once installed, you need to restart Oxygen XML Editor and the XSD to JSON Schema action will invoke the tool.

Quick Installation

You can drag the following Install button and drop it into the main editor in Oxygen (version 24.1 or newer) to quickly initiate the installation process:

Install

Manual Installation

To manually install the add-on, follow these instructions:

1. Go to Help > Install new add-ons to open an add-on selection dialog box.
2. Enter or paste https://www.oxygenxml.com/InstData/Addons/default/updateSite.xml in the Show add-ons from field or select it from the drop-down menu.
3. Select the XSD to JSON Schema add-on and click Next.
4. Read the end-user license agreement. Then select the I accept all terms of the end-user license agreement option and click Finish.
5. Restart the application.

Result: The XSD to JSON Schema dialog box is now available and can be selected from the Tools > JSON Tools menu.

Converting XSD to JSON Schema

To convert an XML Schema (XSD) to a JSON Schema, follow these steps:

1. Select the XSD to JSON Schema action from the Tools > JSON Tools menu.

   Step Result: The XSD to JSON Schema dialog box is displayed:
In the **XSD URL** field, choose or enter the URL of the XML Schema document. The conversion supports XSD versions 1.0 and 1.1.

3. In the **Output file** field, choose the path for the resulting output file.

4. [Optional] You can select the **Open in Editor** option to open the resulting JSON Schema document in the main editing pane.

5. For the **JSON Schema version** option, choose the version of the resulting JSON schema. The possible choices are: **Draft 4**, **Draft 6**, **Draft 7**, and **Draft 2019-09**.

6. [Optional] If you select the **Restrict additional content** option, then `additionalProperties` (for objects) and `additionalItems` (for arrays) will be set to `false` in the resulting schema. By default, these keys are not in the schema, meaning that providing additional content (according to the schema) is allowed.

7. [Optional] You can select the **Preserve case of names from the XSD** option if you want the names from the XSD to remain unchanged in the resulting JSON Schema. Otherwise, the default JAXB naming algorithm will be applied (for example, "some.name" is changed to "SomeName", or "some_oth3r_name" is changed to "SomeOth3RName").

8. Click the **Convert** button.

**Result:** The original XSD document is now converted to a JSON Schema document. The resulting JSON Schema will be the specified draft and will contain:

- The `$id` of the schema, generated from XSD `targetNamespace`.

- The `$definitions` section, which declares `complex` and `enum` types.

- The `anyOf` section, which lists possible top-level elements as an array of objects.

**Other Possible Results:**
• If an XSD type extends another type, then its schema is combined with the schema of the base type using the `allOf` keyword.
• If an extension in XSD defines an element with the same name as an attribute in the base, a property named `rest` is generated to avoid name conflicts in JSON.
• If a property of a complex type is a collection property, the schema of the collection items will be wrapped in the JSON array schema.

Conversion Mappings

The following table lists the specific conversion mapping details.

<table>
<thead>
<tr>
<th>XML Schema Type</th>
<th>JSON Schema Representation</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>anySimpleType</code></td>
<td>string, number, integer, boolean, null</td>
</tr>
<tr>
<td><code>anyType</code></td>
<td>string, number, integer, boolean, null, object, array</td>
</tr>
<tr>
<td><code>string</code></td>
<td>string</td>
</tr>
<tr>
<td><code>normalizedString</code></td>
<td>string</td>
</tr>
<tr>
<td><code>token</code></td>
<td>string</td>
</tr>
<tr>
<td><code>language</code></td>
<td>string</td>
</tr>
<tr>
<td><code>Name</code></td>
<td>string</td>
</tr>
<tr>
<td><code>NCName</code></td>
<td>string</td>
</tr>
<tr>
<td><code>ID</code></td>
<td>string</td>
</tr>
<tr>
<td><code>IDREF</code></td>
<td>string</td>
</tr>
<tr>
<td><code>IDREFS</code></td>
<td>array of strings</td>
</tr>
<tr>
<td><code>ENTITY</code></td>
<td>string</td>
</tr>
<tr>
<td><code>ENTITIES</code></td>
<td>array of strings</td>
</tr>
<tr>
<td><code>NMToken</code></td>
<td>string</td>
</tr>
<tr>
<td><code>NMTOKENS</code></td>
<td>array of strings</td>
</tr>
<tr>
<td><code>boolean</code></td>
<td>boolean</td>
</tr>
<tr>
<td><code>base64Binary</code></td>
<td>array of integers</td>
</tr>
<tr>
<td><code>hexBinary</code></td>
<td>array of integers</td>
</tr>
<tr>
<td><code>float</code></td>
<td>number</td>
</tr>
<tr>
<td><code>decimal</code></td>
<td>number</td>
</tr>
<tr>
<td><code>integer</code></td>
<td>integer</td>
</tr>
<tr>
<td><code>nonPositiveInteger</code></td>
<td>integer</td>
</tr>
<tr>
<td>XML Schema Type</td>
<td>JSON Schema Representation</td>
</tr>
<tr>
<td>----------------------</td>
<td>------------------------------</td>
</tr>
<tr>
<td>negativeInteger</td>
<td>integer</td>
</tr>
<tr>
<td>long</td>
<td>integer</td>
</tr>
<tr>
<td>int</td>
<td>integer</td>
</tr>
<tr>
<td>short</td>
<td>integer</td>
</tr>
<tr>
<td>byte</td>
<td>integer</td>
</tr>
<tr>
<td>nonNegativeInteger</td>
<td>integer</td>
</tr>
<tr>
<td>unsignedLong</td>
<td>integer</td>
</tr>
<tr>
<td>unsignedInt</td>
<td>integer</td>
</tr>
<tr>
<td>unsignedShort</td>
<td>integer</td>
</tr>
<tr>
<td>unsignedByte</td>
<td>integer</td>
</tr>
<tr>
<td>positiveInteger</td>
<td>integer</td>
</tr>
<tr>
<td>double</td>
<td>number</td>
</tr>
<tr>
<td>anyURI</td>
<td>string with &quot;format&quot;:&quot;uri&quot;</td>
</tr>
<tr>
<td>QName</td>
<td>object with &quot;namespaceURI&quot;, &quot;localPart&quot;, &quot;prefix&quot;</td>
</tr>
<tr>
<td>duration</td>
<td>string</td>
</tr>
<tr>
<td>dateTime</td>
<td>string with &quot;format&quot;:&quot;date-time&quot;</td>
</tr>
<tr>
<td>date</td>
<td>string with &quot;format&quot;:&quot;date&quot;</td>
</tr>
<tr>
<td>time</td>
<td>string with &quot;format&quot;:&quot;time&quot;</td>
</tr>
</tbody>
</table>

**Conversion Limitations**

In most cases, the conversion creates an equivalent schema, but there are some limitations:

- Restrictions/facets are not taken into consideration when converting (fractionDigits, pattern, totalDigits, whitespace, minInclusive, maxInclusive, and the restrictions for length, except enumeration). However, extensions and indicators are properly converted (minOccurs, maxOccurs, group, sequence, choice).
- The `<documentation>` element is not converted into `<description>`.
- The `@substitutionGroup` attribute for an element that has no declared type becomes a reference to the element that can substitute it.
- The `@block` attribute is not taken into consideration during the conversion.
OpenAPI Tester

Oxygen XML Editor includes a testing tool for OpenAPI files. The tool provides the ability to inspect OpenAPI request responses and to ensure that they work as expected. It can be used for OpenAPI 3.x in JSON or YAML format.

To use the tool, select OpenAPI Tester from the Tools > JSON Tools menu. This opens a dialog box where you can specify the location of the OpenAPI file that you want to test.

This tool requires an additional add-on to be installed, so the first time you invoke the action, Oxygen XML Editor presents a dialog box asking if you want to install it. Once installed, you need to restart Oxygen XML Editor and the OpenAPI Tester action will invoke the tool.

Quick Installation

You can drag the following Install button and drop it into the main editor in Oxygen (version 24.1 or newer) to quickly initiate the installation process:

Install

Manual Installation

To manually install it the add-on, follow these instructions:

1. Go to Help > Install new add-ons to open an add-on selection dialog box.
2. Enter or paste https://www.oxygenxml.com/InstData/Addons/default/updateSite.xml in the Show add-ons from field or select it from the drop-down menu.
3. Select the OpenAPI Tester add-on and click Next.
4. Read the end-user license agreement. Then select the I accept all terms of the end-user license agreement option and click Finish.
5. Restart the application.

Result: The OpenAPI Tester dialog box is now available and can be selected from the Tools > JSON Tools menu.

OpenAPI Tester Dialog Box

Figure 640. OpenAPI Tester Dialog Box
After selecting the URL for the OpenAPI document (either local or remote) and clicking **OK**, the **OpenAPI Tester** view becomes visible on the right side of the editor. The view can also be opened by selecting **OpenAPI Tester** from the **Window > Show View** menu.

**Figure 641. OpenAPI Tester View**

The tester loads the selected OpenAPI document and fills in the corresponding fields. The tester fields are as follows:

**URL**

The URL of the OpenAPI file. You can specify the path by using the text field or the **Browse** button. If you specify the path directly in the text field, you need to click the **Reload** button. You can use the **Clear** button if you want to remove all the content from the view.

**Server**

The list of servers defined by the OpenAPI document. The global "servers" array can be overridden on the path level or operation level. If it is not provided or is empty, the server URL defaults to "/".

**Path**

The list of individual endpoints (paths) in the API. The full request URL is constructed as `<server-url>/path`.

**Operation**

The list of HTTP operations supported by the selected path. OpenAPI 3.0 supports get, post, put, patch, delete, head, options, and trace.
Parameters tab

The definition of the parameters for the selected operation. OpenAPI 3.0 supports parameters passed via path, query string, headers, and cookies. The required parameters will be marked with a red asterisk. For array parameters, items must be separated by a comma.

Authorization tab

The list of available authentication types. The authentication data is persistent and can be removed from the contextual menu.

Request Body tab

The media type and the body of the request (if the selected operation allows it). For example, GET will not allow specifying a body since that HTTP method disallows it. Oxygen XML Editor will create a sample request body based on the JSON Schemas defined in the OpenAPI document. Usually, you just need to change a few values for the request to be valid.

Variables tab

The list of variables used for server templating (if applicable). Variables are indicated by (curly brackets) in the server URL.

Send button

Executes the request by creating a HTTP client with all the information extracted from the view.

Open response in editor

If selected, the response will be opened in the main editing pane.

Response area

Initially this area is empty. After using the Send button, it presents the message received from the server in response to the request. The expected content type of the message is JSON. The response status and possible errors that may occur are presented right below this area. The status has a green foreground for successful requests and a red foreground otherwise.

Format and Indent (Pretty-Print) Multiple Files

Oxygen XML Editor provides support for formatting and indenting (pretty-print (on page 3222)) multiple files at once. This action is available for any document in XML format, as well as for XQuery, CSS, JavaScript, and JSON documents.

To format and indent multiple files, use the Format and Indent Files action that is available in the contextual menu of the Project view (on page 402) or from the Tools menu. This opens the Format and Indent Files dialog box that allows you to configure options for the action.
Figure 642. Format and Indent Files Dialog Box

The **Scope** section allows you choose from the following scopes:

- **All opened files** - The *pretty-print (on page 3222)* is performed in all opened files.
- **Directory of the current file** - All the files in the folder of the currently edited file.
- **Project files** - All files from the current project.
- **Selected project files** - The selected files from the current project.
- **Specified path** - The *pretty-print (on page 3222)* is performed in the files located at a specified path.

The **Options** section includes the following options:

- **File filter** - Allow you to filter the files from the selected scope.
- **Recurse subdirectories** - When selected, the *pretty-print (on page 3222)* is performed recursively for the specified scope. The one exception is that this option is ignored if the scope is set to **All opened files**.
- **Include hidden files** - When selected, the *pretty-print (on page 3222)* is also performed in the hidden files.
- **Make backup files with extension** - When selected, Oxygen XML Editor makes backup files of the modified files. The default extension is `.bak`, but you can change the extension as you prefer.

**Generate Documentation**

Oxygen XML Editor includes a tool for generating documentation for XSLT, XML Schema, XQuery, and WSDL documents.
Generating Documentation for an XML Schema

Oxygen XML Editor can generate detailed documentation for the components of an XML Schema in HTML, PDF, DocBook, or other custom formats. You can select the components and the level of detail. The components are hyperlinked in both HTML and DocBook documents.

**Note:**
You can generate documentation for both XML Schema version 1.0 and 1.1.

To generate documentation for an XML Schema document, select **XML Schema Documentation** from the **Tools > Generate Documentation** menu or from the **Generate Documentation** submenu in the contextual menu of the **Project view (on page 402)**. You can also open the tool by using the **Generate Documentation** toolbar button.

**Figure 643. XML Schema Documentation Dialog Box**

![XML Schema Documentation Dialog Box](image)

The **Schema URL** field of the dialog box must contain the full path to the XML Schema (XSD) file that will have documentation generated. The schema may be a local or a remote file. You can specify the path to the
schema by entering it in the text field, or by using the ✉️ Insert Editor Variables button or the options in the 📖 Browse drop-down menu.

Output Tab

The following options are available in the Output tab:

- **Format** - Allows you to choose between the following formats:
  - **HTML** - The documentation is generated in HTML output format (on page 1016).
  - **PDF** - The documentation is generated in PDF output format (on page 1019).
  - **DocBook** - The documentation is generated in DocBook output format (on page 1019).
  - **DITA** - The documentation is generated in DITA output format (on page 1019).
  - **Custom** - The documentation is generated in a custom output format (on page 1019), allowing you to control the output. Click the Options button to open a Custom format options dialog box where you can specify a custom stylesheet for creating the output. There is also an option to Copy additional resources to the output folder and you can select the path to the additional Resources that you want to copy. You can also choose to keep the intermediate XML files created during the documentation process by deselecting the Delete intermediate XML file option.

- **Output file** - You can specify the path of the output file by entering it in the text field, or by using the ✉️ Insert Editor Variables button or the options in the 📖 Browse drop-down menu.

- **Split output into multiple files** - Instructs the application to split the output into multiple files. You can choose to split them by namespace, location, or component name.

- **Open in Browser/System Application** - Opens the result in the system application associated with the output file type. For DITA and DocBook documents, this option appears as Open in Editor and the result will be opened in Oxygen XML Editor (in the current editor).

> **Note:**
To set the browser or system application that will be used, open the Preferences dialog box (Options > Preferences) (on page 124), go to Global, and set it in the Default Internet browser field. This will take precedence over the default system application settings.

- **Keep only the annotations with xml:lang set to** - The generated output will contain only the annotations with the @xml:lang attribute set to the selected language. If you choose a primary language code (for example, en for English), this includes all its possible variations (en-us, en-uk, etc.).

Settings Tab

When you generate documentation for an XML schema you can choose what components to include in the output and the details to be included in the documentation.
The **Settings** tab allows you to choose whether or not to include the following components: **Global elements**, **Global attributes**, **Local elements**, **Local attributes**, **Simple Types**, **Complex Types**, **Groups**, **Attribute Groups**, **Redefines**, **Referenced schemas**, and **Include notations**.

You can choose whether or not to include the following other details:

- **Diagram** - Displays the diagram for each component. You can choose the image format (JPEG, PNG, GIF, SVG) to use for the diagram section. The generated diagrams are dependent on the options from the [Schema Design Properties](on page 198) page.
- **Diagram annotations** - This option controls whether or not the annotations of the components presented in the diagram sections are included.
- **Namespace** - Displays the namespace for each component.
- **Location** - Displays the schema location for each component.
- **Type** - Displays the component type if it is not an anonymous one.
- **Type hierarchy** - Displays the types hierarchy.
- **Model** - Displays the model (sequence, choice, all) presented in BNF form. The separator characters that are used depend upon the information item used:
- **xs:all** - Its children will be separated by space characters.
- **xs:sequence** - Its children will be separated by comma characters.
- **xs:choice** - Its children will be separated by / characters.

- **Children** - Displays the list of component's children.
- **Instance** - Displays an XML instance generated based on each schema element.
- **Used by** - Displays the list of all the components that reference the current one. The list is sorted by component type and name.
- **Properties** - Displays some of the component's properties.
- **Facets** - Displays the facets for each simple type.
- **Identity constraints** - Displays the identity constraints for each element. For each constraint there are presented the name, type (unique, key, keyref), reference attribute, selector and field(s).
- **Attributes** - Displays the attributes for the component. For each attribute there are presented the name, type, fixed or default value, usage and annotation.
- **Asserts** - Displays the `assert` elements defined in a complex type. The test, XPath default namespace, and annotation are presented for each assert.
- **Annotations** - Displays the annotations for the component. If you choose **Escape XML Content**, the XML tags are present in the annotations.
- **Source** - Displays the text schema source for each component.
- **Generate index** - Displays an index with the components included in the documentation.
  - **Include local elements and attributes** - If selected, local elements and attributes are included in the documentation index.
  - **Include resource hierarchy** - Specifies whether or not the resource hierarchy for an XML Schema documentation is generated. It is deselected by default.

**Export settings** - Save the current settings in a settings file for further use (for example, if you need the exported settings file for generating the documentation from the command-line interface).

**Import settings** - Reloads the settings from the exported file.

**Generate** - Use this button to generate the XML Schema documentation.

---

**Tip:**

This function can be executed from an automated command-line script, for more details, see Scripting Oxygen (on page 3196).

---

**Related Information:**

Customizing PDF or DocBook Output of Generated XML Schema Documentation (on page 1020)

---

**Generating Documentation for an XSLT Stylesheet**

You can use Oxygen XML Editor to generate detailed documentation in HTML format for the elements (top-level elements whose names are in the XSLT namespace) of an XSLT stylesheet. You can select what XSLT elements to include in the generated documentation and also the level of details to present for each of them.
The elements are hyperlinked. To generate documentation in a custom output format (on page 931), you can edit the XSLT stylesheet used to generate the documentation, or create your own stylesheet.

To open the XSLT Stylesheet Documentation dialog box, select XSLT Stylesheet Documentation from the Tools > Generate Documentation menu or from the Generate Documentation submenu in the contextual menu of the Project view (on page 402). You can also open the tool by using the Generate Documentation toolbar button.

Figure 645. XSLT Stylesheet Documentation Dialog Box

The XSL URL field of the dialog box must contain the full path to the XSL Stylesheet file you want to generate documentation for. The stylesheet may be a local or a remote file. You can specify the path to the stylesheet by entering it in the text field, or by using the Insert Editor Variables button or the options in the Browse drop-down menu.

Output Tab

The following options are available in the Output tab:

- **Format** - Allows you to choose between the following formats:
  - HTML - The documentation is generated in HTML output format (on page 929).
  - Custom - The documentation is generated in a custom output format (on page 931), allowing you to control the output. Click the Options button to open a Custom format options dialog box (on page 932) where you can specify a custom stylesheet for creating the output. There is also an option to Copy additional resources to the output folder and you can select the path to the
additional **Resources** that you want to copy. You can also choose to keep the intermediate XML files created during the documentation process by deselecting the **Delete intermediate XML file** option.

- **Output file** - You can specify the path of the output file by entering it in the text field, or by using the **Insert Editor Variables** button or the options in the **Browse** drop-down menu.
- **Split output into multiple files** - Instructs the application to split the output into multiple files. For large XSLT stylesheets, choosing another split criterion may generate smaller output files, providing faster documentation browsing. You can choose to split them by namespace, location, or component name.
- **Open in Browser/System Application** - Opens the result in the system application associated with the output file type.

**Note:**

To set the browser or system application that will be used, open the **Preferences** dialog box (Options > Preferences) (on page 124), go to **Global**, and set it in the **Default Internet browser** field. This will take precedence over the default system application settings.

**Settings Tab**

When you generate documentation for an XSLT stylesheet you can choose what XSLT elements to include in the output (templates, functions, global parameters, global variables, attribute sets, character maps, keys, decimal formats, output formats, XSLT elements from referenced stylesheets) and the details to include in the documentation.

**Figure 646. Settings Tab of the XSLT Stylesheet Documentation Dialog Box**
The **Settings** tab allows you to choose whether or not to include the following components: **Templates**, **Functions**, **Global parameters**, **Global variables**, **Attribute sets**, **Character maps**, **Keys**, **Decimal formats**, **Output formats**, **Referenced stylesheets**.

You can choose whether or not to include the following other details:

- **Documentation** - Shows the documentation for each XSLT element. For HTML format, the user-defined data elements that are recognized and transformed in documentation blocks of the XSLT elements they precede, are the ones from the following schemas:
  - Oxygen XML Editor built-in XSLT documentation schema.
  - A subset of DocBook 5 elements. The recognized elements are: `section`, `sect1` to `sect5`, `emphasis`, `title`, `ulink`, `programlisting`, `para`, `orderedlist`, `itemizedlist`.
  - A subset of DITA elements. The recognized elements are: `concept`, `topic`, `task`, `codeblock`, `p`, `b`, `i`, `ul`, `ol`, `pre`, `sli`, `step`, `steps`, `li`, `title`, `xref`.
  - Full XHTML 1.0 support.
  - XSLStyle documentation environment. XSLStyle uses DocBook or DITA languages inside its own user-defined data elements. The supported DocBook and DITA elements are the ones mentioned above.
  - DOXSL documentation framework (on page 3219). Supported elements are: `codefrag`, `description`, `para`, `docContent`, `documentation`, `parameter`, `function`, `docSchema`, `link`, `list`, `listitem`, `module`, `parameter`, `template`, `attribute-set`.

  Other XSLT documentation blocks that are not recognized will just be serialized inside an HTML `pre` element. You can change this behavior by using a custom format (on page 931) instead of the built-in HTML format (on page 929) and providing your own XSLT stylesheets.

- **Use comments** - Controls whether or not the comments that precede an XSLT element is treated as documentation for the element they precede. Comments that precede or succeed the `xsl:stylesheet` element, are treated as documentation for the whole stylesheet. Note that comments that precede an import or include directive are not collected as documentation for the imported/included module. Also, comments from within the body of the XSLT elements are not collected at all.

- **Namespace** - Shows the namespace for named XSLT elements.

- **Location** - Shows the stylesheet location for each XSLT element.

- **Parameters** - Shows parameters of templates and functions.

- **References** - Shows the named XSLT elements that are referenced from within an element.

- **Used by** - Shows the list of all the XSLT elements that reference the current named element.

- **Supersedes** - Shows the list of all the XSLT elements that are superseded the current element.

- **Overriding** - Shows the list of all the XSLT elements that override the current element.

- **Return type** - Shows the return type of the function.

- **Source** - Shows the text stylesheet source for each XSLT element.

- **Import precedence** - Shows the computed import precedence as declared in the XSL transformation specifications.

- **Generate index** - Creates an index with all the XSLT elements included in the documentation.
Export settings - Save the current settings in a settings file for further use (for example, if you need the exported settings file for generating the documentation from the command-line interface).

Import settings - Reloads the settings from the exported file.

Generate - Use this button to generate the XSLT documentation.

Tip:
This function can be executed from an automated command-line script, for more details, see Scripting Oxygen (on page 3196).

Related Information:
XSLT Stylesheet Component Documentation Support (on page 912)

Generating HTML Documentation for an XQuery Document

To generate HTML documentation for an XQuery document, use the XQuery Documentation dialog box. It is opened with the XQuery Documentation action that is available from the Tools > Generate Documentation menu or from the Generate Documentation submenu in the contextual menu of the Project view (on page 402). You can also open the tool by using the Generate Documentation toolbar button.

The dialog box allows you to configure a set of parameters for the process of generating the HTML documentation.
Figure 647. XQuery Documentation Dialog Box

The following options are available:

- **Input** - The full path to the XQuery file must be specified in one of the two fields in this section:
  - **URL** - The URL of the file to be used for generating the documentation.
  - **Folder** - The directory that contains the files to be used for generating the documentation. You can also specify the XQuery file extensions to be searched for in the specified directory.
- **Default function namespace** - Optional URI for the default namespace for the submitted XQuery.
- **Predefined function namespaces** - Optional, engine-dependent, predefined namespaces that the submitted XQuery refers to. They allow the conversion to generate annotation information to support the presentation component hypertext linking (only if the predefined modules have been loaded into the local xqDoc XML repository).
- **Open in Browser/System Application** - Select this option if you want the result to be opened in the system application associated with that file type.

**Note:**
To set the browser or system application that will be used, open the Preferences dialog box (Options > Preferences) on page 124, go to Global, and set it in the Default Internet browser field. This will take precedence over the default system application settings.

- **Output** - Allows you to specify where the generated documentation is saved on disk.
Generating Documentation for WSDL Documents

You can use Oxygen XML Editor to generate detailed documentation for the components of a WSDL document in HTML format. You can select the WSDL components to include in your output and the level of details to present for each of them. Also, the components are hyperlinked. You can also generate the documentation in a custom output format (on page 1072) by using a custom stylesheet.

Note:
The WSDL documentation includes the XML Schema components that belong to the internal or imported XML schemas.

To generate documentation for a WSDL document, select WSDL Documentation from the Tools > Generate Documentation menu or from the Generate Documentation submenu in the contextual menu of the Project view (on page 402). You can also open the tool by using the Generate Documentation toolbar button.

Figure 648. WSDL Documentation Dialog Box

The Input URL field of the dialog box must contain the full path to the WSDL document that you want to generate documentation for. The WSDL document may be a local or a remote file. You can specify the path to the WSDL file by entering it in the text field, or by using the Insert Editor Variables button or the options in the Browse drop-down menu.
Output Tab

The following options are available in the **Output** tab:

- **Format** - Allows you to choose between the following formats:
  - HTML - The documentation is generated in HTML output format *(on page 1071).*
  - Custom - The documentation is generated in a custom output format *(on page 1072)*, allowing you to control the output. Click the **Options** button to open a Custom format options dialog box where you can specify a custom stylesheet for creating the output. There is also an option to **Copy additional resources to the output folder** and you can select the path to the additional Resources that you want to copy. You can also choose to keep the intermediate XML files created during the documentation process by deselecting the **Delete intermediate XML file** option.

- **Output file** - You can specify the path of the output file by entering it in the text field, or by using the **Insert Editor Variables** button or the options in the **Browse** drop-down menu.

- **Split output into multiple files** - Instructs the application to split the output into multiple files. For large WSDL documents, choosing a different split criterion may generate smaller output files providing a faster documentation browsing. You can choose to split them by namespace, location, or component name.

- **Open in Browser/System Application** - Opens the result in the system application associated with the output file type.

**Note:**

To set the browser or system application that will be used, open the **Preferences** dialog box *(Options > Preferences)* *(on page 124)*, go to **Global**, and set it in the **Default Internet browser** field. This will take precedence over the default system application settings.

- **Keep only the annotations with xml:lang set to** - The generated output will contain only the annotations with the `@xml:lang` attribute set to the selected language. If you choose a primary language code (for example, `en` for English), this includes all its possible variations (`en-us`, `en-uk`, etc.).

Setting Tab

When you generate documentation for a WSDL document, you can choose what components to include in the output and the details to be included in the documentation.
The **Settings** tab allows you to choose whether or not to include the following:

**Components**
- **Services** - Specifies whether or not the generated documentation includes the WSDL services.
- **Bindings** - Specifies whether or not the generated documentation includes the WSDL bindings.
- **Port Types** - Specifies whether or not the generated documentation includes the WSDL port types.
- **Messages** - Specifies whether or not the generated documentation includes the WSDL messages.
- **XML Schema Components** - Specifies whether or not the generated documentation includes the XML Schema components.
- **Only global elements and types** - Specifies whether or not the generated documentation includes only global elements and types.

**Component Details**
- **Namespace** - Presents the namespace information for WSDL or XML Schema components.
- **Location** - Presents the location information for each WSDL or XML Schema component.
- **Used by** - Presents the list of components that reference the current one.
- **Documentation** - Presents the component documentation. If you choose **Escape XML Content**, the XML tags are presented in the documentation.
- **Source** - Presents the XML fragment that defines the current component.
- **Instance** - Generates a sample XML instance for the current component.
Note:
This option applies to the XML Schema components only.

- **XML Schema Diagram** - Displays the diagram for each XML Schema component. You can choose the image format (JPEG, PNG, GIF, SVG) to use for the diagram section.
- **Diagram annotations** - Specifies whether or not the annotations of the components presented in the diagram sections are included.

- **Generate index** - Displays an index with the components included in the documentation.
  - **Include local elements and attributes** - If selected, local elements and attributes are included in the documentation index.
  - **Include resource hierarchy** - Specifies whether or not the resource hierarchy for an XML Schema documentation is generated. It is deselected by default.

**Export settings** - Save the current settings in a settings file for further use (for example, if you need the exported settings file for generating the documentation from the command-line interface).

**Import settings** - Reloads the settings from the exported file.

**Generate** - Use this button to generate the WSDL documentation.

Tip:
This function can be executed from an automated command-line script, for more details, see Scripting Oxygen (on page 3196).

### Generating JSON Schema Documentation

Oxygen XML Editor includes a tool for generating documentation for a JSON Schema file in HTML format. To generate JSON Schema documentation, select **JSON Schema Documentation** from the **Tools > Generate Documentation** menu. You can also open the tool by using the **Generate Documentation** toolbar button. This opens a dialog box where you can specify the location of the JSON Schema file and HTML output file.

This tool requires an additional add-on to be installed, so the first time you invoke the action, Oxygen XML Editor presents a dialog box asking if you want to install it. Once installed, you need to restart Oxygen XML Editor and the **JSON Schema Documentation** action will invoke the tool.

### Quick Installation

You can drag the following **Install** button and drop it into the main editor in **Oxygen** (version 24.1 or newer) to quickly initiate the installation process:
Manual Installation

To manually install the add-on, follow these instructions:

1. Go to Help > Install new add-ons to open an add-on selection dialog box.
2. Enter or paste https://www.oxygenxml.com/InstData/Addons/default/updateSite.xml in the Show add-ons from field or select it from the drop-down menu.
3. Select the JSON Schema Documentation add-on and click Next.
4. Read the end-user license agreement. Then select the I accept all terms of the end-user license agreement option and click Finish.
5. Restart the application.

Result: The JSON Schema Documentation dialog box is now available and can be selected from the Tools > Generate Documentation menu.

JSON Schema Documentation Dialog Box

![JSON Schema Documentation Dialog Box](image)

The JSON Schema Documentation dialog box includes the following fields and options:

**JSON Schema URL**

The URL of the JSON Schema file. You can specify the path by using the text field, the history drop-down menu, or the browsing actions in the Browse drop-down list.

**Output file**

The path to the folder where the generated HTML file will be saved.

**Split output into multiple files**
If selected, the application splits the output into multiple files. You can choose between splitting them by **component** name or **location**.

**Open in Browser/System Application**

If selected, the generated result is opened in the system application associated with the output file type (HTML).

**Included component details**

This section can be used to specify whether or not the following components are shown in the generated documentation:

- **Annotations** - Displays the annotations (title, description) for each component (property or definition).
- **Constraints** - Displays the schema constraints for each component, according to its type.
- **Properties** - Displays the `properties` of an Object Schema.
- **Pattern Properties** - Displays the `patternProperties` of an Object Schema.
- **Enumerations** - Displays the enumerated values, if specified in the schema.
- **Source** - Displays the text schema source for each component.
- **Used By** - Displays the list of all the components that reference the current definition.
- **Composition** - Displays the `oneOf`, `anyOf`, and `allOf` Compositors that are used for combining schemas.
- **Diagram** - Displays the diagram image for each component. The diagrams are generated according to the options specified in the **Schema Design preferences page (on page 198)**. Diagrams are also generated for components within schemas from referenced files.
- **Image Map** - Diagrams will include hyperlinks that navigate to each particular component.
- **Location** - Displays the schema location for each component.

You can click **Generate** at any point to generate the JSON Schema documentation.

**Generated JSON Schema Documentation in HTML Format**

After generating the JSON Schema documentation, it is presented in a visual diagram style with various sections, hyperlinks, and options.
The generated documentation includes a Table of Contents on the left pane with links to particular sections in the right pane. You can collapse or expand details by using the Showing options or the Collapse or Expand buttons.

**Generating OpenAPI Documentation**

Oxygen XML Editor includes a tool for generating documentation for OpenAPI components in HTML format, including annotations and cross references. The documentation displays information about the servers, paths, components and tags defined in the OpenAPI 3.0 documents and it is presented in a visual diagram style with various sections, hyperlinks, and filtering options.

**Quick Installation**

You can drag the following Install button and drop it into the main editor in Oxygen (version 24.1 or newer) to quickly initiate the installation process:
Manual Installation

To manually install the add-on, follow these instructions:

1. Go to Help > Install new add-ons to open an add-on selection dialog box.
2. Enter or paste https://www.oxygenxml.com/InstData/Addons/default/updateSite.xml in the Show add-ons from field or select it from the drop-down menu.
3. Select the OpenAPI Documentation Generator add-on and click Next.
4. Read the end-user license agreement. Then select the I accept all terms of the end-user license agreement option and click Finish.
5. Restart the application.

Result: The OpenAPI Documentation dialog box is now available and can be selected from the Tools > Generate Documentation menu.

Generating OpenAPI Documentation

To generate OpenAPI documentation, select OpenAPI Documentation from the Tools > Generate Documentation menu. This opens a dialog box where you can specify the location of the OpenAPI file and the HTML output file.

This tool requires an additional add-on to be installed, so the first time you invoke the action, Oxygen XML Editor presents a dialog box asking if you want to install it. Once installed, you need to restart Oxygen XML Editor and the OpenAPI Documentation action will invoke the tool.

Figure 652. OpenAPI Documentation Dialog Box

The OpenAPI Documentation dialog box includes the following fields and options:
OpenAPI URL

The URL of the OpenAPI file. You can specify the path by using the text field or the browsing button ( Archive ).

Output file

The path to the folder where the generated HTML file will be saved.

Split output into multiple files

If selected, the application splits the output into multiple files. You can choose between splitting them by component name or location.

Included component details

This section can be used to specify whether or not details about the following components that belong to internal or imported schemas are shown in the generated documentation:

- **Annotations** - Displays the annotations (title, description) for each component (property or definition).
- **Constraints** - Displays the schema constraints for each component, according to its type.
- **Source** - Displays the text schema source for each component.
- **Location** - Displays the schema location for each component.
- **Used By** - Displays the list of all the components that reference the current definition.
- **Properties** - Displays the properties of an Object Schema.
- **Pattern Properties** - Displays the patternProperties of an Object Schema.
- **Enumerations** - Displays the enumerated values, if specified in the schema.
- **Diagram** - Displays the diagram image for each component. The diagrams are generated according to the options specified in the Schema Design preferences page (on page 198). Diagrams are also generated for components within schemas from referenced files.
- **Composition** - Displays the oneOf, anyOf, and allOf Compositors that are used for combining schemas.
- **Image Map** - Diagrams will include hyperlinks that navigate to each particular component.

You can click **Generate** at any point to generate the OpenAPI documentation.

Generated OpenAPI Documentation in HTML Format

After generating the OpenAPI documentation, it is presented in a visual diagram style with various sections, hyperlinks, and options.
The generated documentation includes a Table of Contents on the left pane with links to particular sections in the right pane. You can collapse or expand details by using the Showing options or the Collapse or Expand buttons.

**Canonicalizing Files**

You can select the [canonicalization](on page 3217) algorithm to be used for a document from the dialog box that is displayed by using the Canonicalize action that is available from the Source submenu when invoking the contextual menu in Text mode or from the Tools menu.
The **Canonicalize** dialog box allows you to set the following options:

- **Input URL** - Available if the **Canonicalize** action was selected from the **Tools** menu. It allows you to specify the location of the input file.

- **Exclusive** - If selected, the exclusive (uncommented) canonicalization method is used.

  **Note:** *Exclusive Canonicalization* just copies the namespaces you are actually using (the ones that are a part of the XML syntax). It does not look into attribute values or element content, so the namespace declarations required to process these are not copied. This is useful if you have a signed XML document that you want to insert into other XML documents (or you need self-signed structures that support placement within various XML contexts), as it will ensure the signature is verified correctly each time.

- **Exclusive with comments** - If selected, the exclusive with comments canonicalization method is used.

- **Inclusive** - If selected, the inclusive (uncommented) canonicalization method is used.

  **Note:** *Inclusive Canonicalization* copies all the declarations, even if they are defined outside of the scope of the signature, and all the declarations you might use will be unambiguously specified. Inclusive Canonicalization is useful when it is less likely that the signed data will be inserted
in other XML document and it is the safer method from the security standpoint because it requires no knowledge of the data that are to be secured to safely sign them. A problem may occur if the signed document is moved into another XML document that has other declarations because the Inclusive Canonicalization will copy them and the signature will be invalid.

- **Inclusive with comments** - If selected, the inclusive with comments canonicalization (on page 3217) method is used.
- **XPath** - The XPath expression provides the fragments of the XML document to be signed.
- **Output** - Available if the Canonicalize action was selected from the Tools menu. It allows you to specify the output file path where the signed XML document will be saved.
- **Open in editor** - If selected, the output file will be opened in the editor.

Related Information:
Digital Signatures Overview (on page 877)

**Signing Files**

You can select the type of signature to be used for documents from a signature settings dialog box. To open this dialog box, select the Sign action from the Source submenu when invoking the contextual menu in Text mode or from the Tools menu.
Figure 655. Signature Settings Dialog Box

The following options are available:

Note:
If Oxygen XML Editor could not find a valid certificate, a link is provided at the top of the dialog box that opens the XML Signing Certificates preferences page (on page 268) where you can configure a valid certificate.

- **Input** - Available if the **Sign** action was selected from the **Tools** menu. Specifies the location of the input URL.
- **Transformation Options** - See the Digital Signature Overview (on page 877) section for more information about these options.
None - If selected, no canonicalization (on page 3217) algorithm is used.

Exclusive - If selected, the exclusive (uncommented) canonicalization (on page 3217) method is used.

Note:

Exclusive Canonicalization just copies the namespaces you are actually using (the ones that are a part of the XML syntax). It does not look into attribute values or element content, so the namespace declarations required to process these are not copied. This is useful if you have a signed XML document that you want to insert into other XML documents (or you need self-signed structures that support placement within various XML contexts), as it will ensure the signature is verified correctly each time.

Exclusive with comments - If selected, the exclusive with comments canonicalization (on page 3217) method is used.

Inclusive - If selected, the inclusive (uncommented) canonicalization (on page 3217) method is used.

Note:

Inclusive Canonicalization copies all the declarations, even if they are defined outside of the scope of the signature, and all the declarations you might use will be unambiguously specified. Inclusive Canonicalization is useful when it is less likely that the signed data will be inserted in other XML document and it is the safer method from the security standpoint because it requires no knowledge of the data that are to be secured to safely sign them. A problem may occur if the signed document is moved into another XML document that has other declarations because the Inclusive Canonicalization will copy them and the signature will be invalid.

Inclusive with comments - If selected, the inclusive with comments canonicalization (on page 3217) method is used.

• XPath - The XPath expression provides the fragments of the XML document to be signed.
• ID - Provides ID of the XML element to be signed.
• Envelope - If selected, the enveloped signature is used. See the Digital Signature Overview (on page 877) for more information.
• Detached - If selected, the detached signature is used. See the Digital Signature Overview (on page 877) for more information.
• Append KeyInfo - If this option is selected, the &lt;ds:KeyInfo&gt; element will be added in the signed document.
• Signature algorithm - The algorithm used for signing the document. The following options are available: RSA with SHA1, RSA with SHA256, RSA with SHA384, and RSA with SHA512.
• Output - Available if the Sign action was selected from the Tools menu. Specifies the path of the output file where the signed XML document will be saved.
• Open in editor - If selected, the output file will be opened in Oxygen XML Editor.
Related Information:
Digital Signatures Overview (on page 877)
Verifying Signature (on page 885)
Example of How to Digitally Sign XML Files or Content (on page 885)

Verifying Signature

You can verify the signature of a file by selecting the **Verify Signature** action from the **Source** submenu when invoking the contextual menu in Text mode or from the **Tools** menu. The **Verify Signature** dialog box then allows you to specify the location of the file whose signature is verified.

If the signature is valid, a dialog box displays the name of the signer. Otherwise, an error shows details about the problem.

Related Information:
Digital Signatures Overview (on page 877)
Signing Files (on page 882)
Example of How to Digitally Sign XML Files or Content (on page 885)

WSDL SOAP Analyzer Tool

**WSDL SOAP Analyzer** is a tool that helps you test if the messages defined in a Web Service Descriptor (WSDL) are accepted by a Web Services server.

After you edit and validate your Web service descriptor against a mix of the XML Schemas for WSDL and SOAP, it is easy to check if the defined SOAP messages are accepted by the remote Web Services server by using the integrated **WSDL SOAP Analyzer** tool (available from the toolbar or **Tools** menu).

Oxygen XML Editor provides two ways of testing, one for the currently edited WSDL document and another for the remote WSDL documents that are published on a web server. To open the **WSDL SOAP Analyzer** tool for the currently edited WSDL document do one of the following:

- Click the WSDL SOAP Analyzer toolbar button.
- Use the WSDL SOAP Analyzer action from the Tools menu.
- Go to Open with > WSDL SOAP Analyzer in the contextual menu of the Project (on page 402) view.
This tool contains a SOAP analyzer and sender for Web Services Description Language file types. The analyzer fields are as follows:

- **Services** - The list of services defined by the WSDL file.
- **Ports** - The ports for the selected service.
- **Operations** - The list of available operations for the selected service.
- **Action URL** - The script that serves the operation.
- **SOAP Action** - Identifies the action performed by the script.
- **Version** - Choose between 1.1 and 1.2. The SOAP version is selected automatically depending on the selected port.
• **Request Editor** - It allows you to compose the web service request. When an action is selected, Oxygen XML Editor tries to generate as much content as possible for the SOAP request. The envelope of the SOAP request has the correct namespace for the selected SOAP version, that is `http://schemas.xmlsoap.org/soap/envelope/` for SOAP 1.1 or `http://www.w3.org/2003/05/soap-envelope` for SOAP 1.2. Usually you just have to change a few values for the request to be valid. The **Content Completion Assistant (on page 3217)** is available for this editor and is driven by the schema that defines the type of the current message. While selecting various operations, Oxygen XML Editor remembers the modified request for each one. You can click the **Regenerate** button to overwrite your modifications for the current request with the initial generated content.

• **Attachments List** - You can define a list of file URLs to be attached to the request.

• **Response Area** - Initially it displays an auto generated server sample response so you can have an idea about how the response looks like. After pressing the **Send** button, it presents the message received from the server in response to the Web Service request. It may show also error messages. If the response message contains attachments, Oxygen XML Editor prompts you to save them, then tries to open them with the associated system application.

• **Errors List** - There may be situations where the WSDL file is respecting the WSDL XML Schema, but it fails to be valid (for example, in the case of a message that is defined by means of an element that is not found in the types section of the WSDL). In such a case, the errors are listed here. This list is presented only when there are errors.

• **Send Button** - Executes the request. A status dialog box is displayed when Oxygen XML Editor is connecting to the server.

The testing of a WSDL file is straight-forward. Click the WSDL analysis button, then select the service, the port, and the operation. The editor generates the skeleton for the SOAP request. You can edit the request, eventually attach files to it and send it to the server. Watch the server response in the response area. You can find more details in the **Testing Remote WSDL Files (on page 1075)** section.

**Note:**

SOAP requests and responses are automatically validated in the **WSDL SOAP Analyzer** using the XML Schemas specified in the WSDL file.

Once defined, a request derived from a Web Service descriptor can be saved with the **Save** button to a Web Service SOAP Call (WSSC) file for later reuse. In this way, you save time in configuring the URLs and parameters.

You can open the result of a Web Service call in an editor panel using the **Open** button.

**Testing Remote WSDL Files**

To open and test a remote WSDL file the steps are the following:

1. Go to **Tools > WSDL SOAP Analyzer**.
2. On the **WSDL File** tab enter the URL of the remote WSDL file.
3. Click the OK button.
This will open the WSDL SOAP Analyzer tool (on page 1072). In the Saved SOAP Request tab you can open directly a previously saved Web Service SOAP Call (WSSC) file, thus skipping the analysis phase.

**XML Schema Regular Expressions Builder Tool**

The XML Schema regular expressions builder allows you to test regular expressions on a fragment of text as they are applied to an XML instance document. Start the tool by selecting XML Schema Regular Expressions Builder from the Tools menu.

**Figure 657. XML Schema Regular Expressions Builder Dialog Box**

The dialog box contains the following:

- **Regular expressions editor**
  Allows you to edit the regular expression to be tested and used. Content completion is available and presents a list with all the predefined expressions. It is triggered by pressing Ctrl + Space (Command + Space on macOS).

- **Error display area**
If the edited regular expression is incorrect, an error message will be displayed here. The message contains the description and the exact location of the error. Also, clicking the quick navigation button (↔) highlights the error inside the regular expression.

**Category**

You can choose from several categories of predefined expressions. The selected category influences the displayed expressions in the Available expressions table.

**Available expressions**

This table includes the available regular expressions and a short description for each of them. The set of expressions depends on the category selected in the previous Category combo box. You can add an expression in the Regular expressions editor by double-clicking the expression row in the table. You will notice that in the case of Character categories and Block names, the expressions are also listed in complementary format.

**Evaluate expression on**

You can choose between two options:

- **Evaluate expression on each line** - The edited expression will be applied on each line in the Test area.
- **Evaluate expression on all text** - The edited expression will be applied on the whole text.

**Test**

A text editor that allows you to enter a text sample that will have the regular expression applied. All matches of the edited regular expression will be highlighted.

After editing and testing your regular expression you can insert it in the current editor. The Insert button will become active when an editor is opened in the background and there is an expression in the Regular expressions editor.

The regular expression builder cannot be used to insert regular expressions in the Grid mode (on page 354) or schema Design mode (on page 355). Accordingly, the Insert button will be not available if the current document is edited in these modes.

**Note:**

Some regular expressions may indefinitely block the Java Regular Expressions engine. If the execution of the regular expression does not end in about five seconds, the application displays a dialog box that allows you to interrupt the operation.

**Large File Viewer**

XML files tend to become larger and larger mostly because they are frequently used as a format for database export or for porting between multiple database formats. Traditional XML text editors simply cannot handle
opening these huge export files, some having sizes exceeding one gigabyte, because all of the file content must be loaded in memory before the user can actually view it.

The best performance of the viewer is obtained for encodings that use a fixed number of bytes per character (such as UTF-16 or ASCII). The performance for UTF-8 is very good for documents that use mostly characters of the European languages. For the same encoding, the rendering performance is higher for files consisting of long lines (up to few thousands characters) and may degrade for short lines. In fact, the maximum size of a file that can be rendered in the Large File Viewer decreases when the total number of the text lines of the file increases. Trying to open a very large file (for example, a file of 4 GB) with a very high number of short lines (100 or 200 characters per line) may produce an out of memory error (OutOfMemoryError) that would require either increasing the Java heap memory with the -Xmx startup parameter or decreasing the total number of lines in the file.

The powerful Large File Viewer is available from the Tools menu or as a standalone application. You can also right-click a file in your project and choose to open it with the viewer. It uses an efficient structure for indexing the open document. No information from the file is stored in the main memory, just a list of indexes in the file. In this way the viewer can open very large files, up to 10 gigabytes. If the open file is XML, the encoding used to display the text is detected from the XML prolog of the file. For other file types, the encoding is taken from the Oxygen XML Editor options. See Encoding for non-XML files (on page 168).

Figure 658. Large File Viewer

Large File Viewer components:
The menu bar provides menu driven access to all the features and functions that are available in Large File Viewer:

File > Open

Opens files in the viewer (also available in the contextual menu).

File > Exit

Closes the viewer.

Edit > Copy

Copies the selected text to clipboard (also available in the contextual menu).

Find > Find

Opens a reduced Find dialog box that provides some basic search options, such as:

- **Case sensitive** - When selected, operations are case-sensitive.
- **Regular Expression** - When selected, allows you to use any regular expression in Perl-like syntax *(on page 446)*.
- **Wrap around** - Continues the find operation from the start/end of the document after reaching the end/, depending on whether the search is in forward or backward direction.

Help > Help

Provides access to the User Manual.

The status bar provides information about the current file path, the Unicode representation of the character at the cursor position and the line and column in the open document where the cursor is located.

**Attention:**

For faster computation the Large File Viewer uses a fixed font (plain, monospace font of size 12) to display characters. The font is *not* configurable from the Preferences page *(on page 124)*.

**Tip:**

The best performance of the viewer is accomplished for encodings that use a fixed number of bytes per character (such as UTF-16 or ASCII). The performance for UTF-8 is very good for documents that use mostly characters of the European languages. For the same encoding the rendering performance is high for files consisting of short lines (up to a few thousand characters) and may degrade for long lines.

**Hex Viewer**

When the Unicode characters that are visible in a text viewer or editor are not enough and you need to see the byte values of each character of a document, you can start the Hex Viewer that is available on the Tools menu. It has two panels: the characters are rendered in the right panel and the bytes of each
character are displayed in the left panel. There is a 1:1 correspondence between the characters and their byte representation: the byte representation of a character is displayed in the same matrix position of the left panel as the character in the matrix of the right panel.

**Figure 659. Hex Viewer**

To open a file in **Hex Viewer** use the **File > Open** action. Alternatively, you can drag a file and drop it in the **Hex Viewer** panel.

**Standalone SVG Viewer**

Oxygen XML Editor includes a simple **SVG Viewer** that allows you to work with SVG images.

To open the viewer, select **SVG Viewer** from the **Tools** menu.
You can browse for and open any SVG file that has the `.svg` or `.svgz` extension.

If the file is included in the current project, you can open it in the viewer by right-clicking the image file in the **Project view (on page 402)** and selecting **Open with > SVG Viewer**.

**Actions Available in the SVG Viewer**

The following actions are available in the **SVG Viewer**:

**Zoom in**

To zoom in on an image, use any of the following methods:

- Scroll **forward** with the mouse wheel.
- Select **Zoom in** from the contextual menu.
- Use the **Ctrl + I (Command + I on macOS)** keyboard shortcut.

**Zoom out**

To zoom out on an image, use any of the following methods:

- Scroll **backward** with the mouse wheel.
- Use the **Ctrl + O (Command + O on macOS)** keyboard shortcut.
- Select **Zoom out** from the contextual menu.

**Rotate**

To rotate an image, use either of the following methods:

- Use the **Ctrl + Right-Click + Drag (Command + Right-Click + Drag on macOS)** shortcut.
- Select **Rotate** from the contextual menu. This rotates the image exactly 90 degrees clockwise.
Refresh
To refresh (or reset) an image, use either of the following methods:

• Use the Ctrl + T (Command + T on macOS) keyboard shortcut.
• Select Refresh from the contextual menu.

Move
To move an image, use either of the following methods:

• Use the Arrow Keys on your keyboard.
• Use the Shift + Left-Click + Drag shortcut.

Pan
To pan an image, click and drag the image with your mouse.

Related Information:
Editing SVG Files (on page 1251)

Tree Editor
The Tree Editor (Tools > Tree Editor) is used for editing the content of a document displayed as an XML tree. The workspace offers the following functional areas:

• Main menu - Provides access to all the features and functions available in Oxygen XML Editor Tree Editor.
• Toolbar - Provides easy access to common and frequently used functions. Each icon is a button that acts as a shortcut to a related function.
• Editor panel - Easy editing of structured mark-up documents. Each token has an associated icon for easier visual identification.
• Message panel - Displays messages returned from user operations.
• Model view - Shows the detailed information about the attribute or element that you are working on.
• All Elements panel - Presents a list of all defined elements that can be inserted within your document.

The tree editor does not offer entity support. Entities are not presented with a special type of node in the tree and new entity nodes cannot be inserted in the document.

Compare Files Tool
The built-in Compare Files tool can be used to compare files or XML file fragments. The tool provides a mechanism for comparing two files or fragments, as well as the mechanism for a three-way comparison. The utility is available from the Tools menu or can be opened as a stand-alone application from the Oxygen XML Editor installation folder (diffFiles.exe).
Two-Way Comparisons

The Compare Files tool can be used to compare the differences between two files or XML fragments.

Compare Files

To perform a two-way comparison, follow these steps:

1. Open a file in the left panel and the file you want to compare it to in the right panel. You can specify the path by using the text field, the history drop-down, or the browsing actions in the Browse drop-down menu.

   **Step Result:** The selected files are opened in the two side-by-side editors. A text editing mode is used to offer a better view of the differences.

2. To highlight the differences between the two files, click the Perform File Differencing button from the toolbar.

3. You can use the drop-down menu on the left side of the toolbar to change the algorithm (on page 2682) for the operation.
4. You can also use the Diff Options button to access the Files Comparison preferences page where you can choose to ignore certain types of markup and configure various options.

5. If you are comparing XML documents using the XML Fast or XML Accurate algorithms, you can enter an XPath 2.0 expression in the Ignore nodes by XPath text field to ignore certain nodes from the comparison.

The resulting comparison will show you differences between the two files. The line numbers on each side and colored marks on the right-side vertical stripe help you to quickly identify the locations of the differences. Adjacent changes are grouped into blocks of changes. This layout allows you to easily identify and focus on a group of related changes.

Figure 662. Two-Way Differences

Highlighting Colors

The differences are also highlighted in several colors, depending on the type of change, and dynamic lines connect the compared fragments in the middle section between the two panes. The highlighting colors can be customized in the Files Comparison / Appearance preferences page (on page 288), but the default colors and their shades mean the following:

- **Pink** - Identifies modifications on either side.
- **Gray** - Identifies an addition of a node in the left side (your outgoing changes).
- **Blue** - Identifies an addition of a node in the right side (incoming changes).
- **Lighter Shade** - Identifies blocks of changes that can be merged in their entirety.
- **Darker Shade** - Identifies specific changes within the blocks that can be merged more precisely.

Comparing Fragments (Copy/Paste)

To compare XML file fragments, you need to copy and paste the fragments you want to compare into each side, without selecting a file. If a file is already selected, you need to close it using the Close (Ctrl + W (Command + W on macOS)) button, before pasting the fragments. Other notes for pasting fragments:

- As long as the fragment is more than 10 characters, the application will attempt to automatically detect the content type. It can detect the following types: XML, DTD, CSS, JSON, and Markdown (if it starts with #). If one of those content types is detected, the fragments will be displayed with syntax highlights.
- If you save modified fragments, a dialog box opens that allows you to save the changes as a new document.

Navigate Differences

To navigate through differences, do one of the following:
• Use the navigation buttons on the toolbar (or in the Compare menu).
• Select a block of differences by clicking its small colored marker in the overview ruler located in the right-most part of the window. At the top of the overview ruler there is a success indicator that turns green where there are no differences, or red if differences are found.
• Click a colored area in between the two text editors.

Editing Actions

You can edit the files directly in either editing pane. The two editors are constantly synchronized and the differences are refreshed when you save the modified document or when you click the Perform File Differencing button.

A variety of actions are available on the toolbar (on page 486) and in the various menus (on page 490) (these same actions are also available in the contextual menu in both editing panes). The tool also includes some inline actions to help you merge, copy, or remove changes. When you select a change, the following inline action widgets are available, depending on the type of change:

- **Append left change to right and Append right change to left**
  
  Copies the content of the selected change from one side and appends it on the other, according to the content of the corresponding change. As a result, the side where the arrow points to will contain the changes from both sides.

- **Copy change from left to right and Copy change from right to left**
  
  Replaces the content of a change from one side with the content of the corresponding change from the other side.

- **Remove change**
  
  Rejects the change on the particular side and preserves the particular content on the other side.

Two-Way Diff Algorithms

Oxygen XML Editor offers the following two-way diff algorithms to compare files or fragments:

- **Auto** - Selects the most appropriate algorithm, based on the compared content and its size (selected by default).
- **Characters** - Computes the differences at character level, meaning that it compares two files or fragments looking for identical characters. This algorithm is not available when the file comparison is in Author comparison mode.
- **Words** - Computes the differences at word level, meaning that it compares two files or fragments looking for identical words. This algorithm is not available when the file comparison is in Author comparison mode.
- **Lines** - Computes the differences at line level, meaning that it compares two files or fragments looking for identical lines of text. This algorithm is not available when the file comparison is in Author comparison mode.
• **Syntax Aware** - Computes differences for known file types or fragments. This algorithm splits the files or fragments into sequences of *tokens* and computes the differences between them. The meaning of a *token* depends on the type of compared files or fragments.

Known file types include those listed in the **New** dialog box, such as XML file types (XSLT files, XSL-FO files, XSD files, RNG files, NVDL files, etc.), XQuery file types (.xquery, .xq, .xqy, .xqm extensions), DTD file types (.dtd, .ent, .mod extensions), TEXT file type (.txt extension), or PHP file type (.php extension).

For example:

- When comparing XML files or fragments, a token can be one of the following:
  - The name of an XML tag
  - The `<` character
  - The `/>` sequence of characters
  - The name of an attribute inside an XML tag
  - The `=` sign
  - The `"` character
  - An attribute value
  - The text string between the start tag and the end tag (a text node that is a child of the XML element corresponding to the XML tag that encloses the text string)

- When comparing plain text, a token can be any continuous sequence of characters or any continuous sequence of whitespaces, including a new line character.

• **XML Fast** - Comparison that works well on large files or fragments, but it is less precise than **XML Accurate**.

• **XML Accurate** - Comparison that is more precise than **XML Fast**, at the expense of speed. It compares two XML files or fragments looking for identical XML nodes.

### Three-Way Comparisons

Oxygen XML Editor also includes a three-way comparison feature to help you solve conflicts and merge changes between multiple modifications. It is especially helpful for teams who have multiple authors editing and committing the same documents. It provides a comparison between a local change, another change, and the original base revision. Some additional advantages include:

- Visualize and merge content that was modified by you and another member of your team.
- Marks differences correctly even when the document structure is rearranged.
- Allows you to merge XML-relevant modifications.
Figure 663. Three-Way Comparison

Compare Files

To perform a three-way comparison, follow these steps:

1. Open a file in the left panel and the file you want to compare it to in the right panel. You can specify the path by using the text field, the history drop-down, or the browsing actions in the Browse drop-down menu.

   **Step Result:** The selected files are opened in the two side-by-side editors. A text editing mode is used to offer a better view of the differences.

2. Click the Three-Way Comparison button on the toolbar and select the base (original) file in the Base field. You can specify the path by using the text field, the history drop-down, or the browsing actions in the Browse drop-down menu.

3. To highlight the differences, click the Perform File Differencing button on the toolbar.

4. You can use the drop-down menu on the left side of the toolbar to change the algorithm (on page 2682) for the operation.

5. You can also use the Diff Options button to access the Files Comparison preferences page where you can choose to ignore certain types of markup and configure various options.

The resulting comparison will show you differences between the two files, as well as differences between either of them and the base (original) file. The line numbers on each side and colored marks on the right-side vertical stripe help you to quickly identify the locations of the differences. Adjacent changes are grouped into blocks of changes.

Figure 664. Three-Way Differences

**Highlighting Colors**
The differences are also highlighted in several colors, depending on the type of change, and dynamic lines connect the compared fragments in the middle section between the two panes. The highlighting colors can be customized in the Files Comparison / Appearance preferences page (on page 288), but the default colors and their shades mean the following:

- **Pink** - Identifies blocks of changes that include conflicts.
- **Gray** - Identifies your outgoing changes that do not include conflicts.
- **Blue** - Identifies incoming changes that do not include conflicts.
- **Lighter Shade** - Identifies blocks of changes that can be merged in their entirety.
- **Darker Shade** - Identifies specific changes within the blocks that can be merged more precisely.

### Navigate Differences

To navigate through differences, do one of the following:

- Use the navigation buttons on the toolbar (or in the Compare menu).
- Select a block of differences by clicking its small colored marker in the overview ruler located in the right-most part of the window. At the top of the overview ruler there is a success indicator that turns green where there are no differences, or red if differences are found.
- Click a colored area in between the two text editors.

### Editing Actions

You can edit the files directly in either editing pane. The two editors are constantly synchronized and the differences are refreshed when you save the modified document or when you click the Perform File Differencing button.

A variety of actions are available on the toolbar (on page 486) and in the various menus (on page 490) (these same actions are also available in the contextual menu in both editing panes). The tool also includes some inline actions to help you merge, copy, or remove changes. When you select a change, the following inline action widgets are available, depending on the type of change:

- **Append left change to right** and **Append right change to left**
  
  Copies the content of the selected change from one side and appends it on the other, according to the content of the corresponding change. As a result, the side where the arrow points to will contain the changes from both sides.

- **Copy change from left to right** and **Copy change from right to left**
  
  Replaces the content of a change from one side with the content of the corresponding change from the other side.

- **Remove change**
  
  Rejects the change on the particular side and preserves the particular content on the other side.

### Three-Way Diff Algorithms

Oxygen XML Editor offers the following three-way diff algorithms to compare files:
• **Auto** - Selects the most appropriate algorithm, based on the compared content and its size (selected by default).

• **Lines** - Computes the differences at line level, meaning that it compares two files or fragments looking for identical lines of text. This algorithm is not available when the file comparison is in **Author** comparison mode.

• **XML Fast** - Comparison that works well on large files or fragments, but it is less precise than **XML Accurate**.

• **XML Accurate** - Comparison that is more precise than **XML Fast**, at the expense of speed. It compares two XML files or fragments looking for identical XML nodes.

### Second-Level Comparisons

For both two-way and three-way comparisons, Oxygen XML Editor automatically performs a second-level comparison for the **Lines**, **XML Fast**, and **XML Accurate** algorithms. After the first comparison is finished, the second-level comparison for the **Lines** algorithm is processed on text nodes using a word level comparison, meaning that it looks for identical words. For the **XML Fast** and **XML Accurate** algorithms, the second-level comparison is processed using a syntax-aware comparison (on page 2683), meaning that it looks for identical tokens. This second-level comparison makes it easier to spot precise differences and you can merge or reject the precise modifications.

**Figure 665. Second-Level Diff Comparison**

![Second-Level Diff Comparison](image)

**Note:**

If a modified text fragment contains XML markup (such as processing instructions, XML comments, CDATA, or elements), the second-level comparison will not automatically be performed. In this case you can manually select a second-level comparison by doing a word level or character level comparison.

To do a word level comparison, select **Show word level details** from the contextual menu or **Compare** menu.

**Figure 666. Word Level Comparison**

![Word Level Comparison](image)
To do a character level comparison, select **Show Character Level details** from the contextual menu or **Compare** menu.

**Figure 667. Character Level Comparison**

Author Visual Mode

The **Compare Files** tool includes an **Author** comparison mode that displays the files in a visual mode similar to the **Author** editing mode in **Oxygen XML Editor/Author**. This makes it easier to see how the compared changes will look in the final output. This visual mode is available when the compared files are detected as being XML. To determine whether the files are initially opened in the merge tool's **Text** or **Author** comparison mode, it detects the **Initial Edit Mode** in the **Document Type Association** configuration (on page 141) and the mode the files were last opened in **Oxygen XML Editor/Author**.

**Note:**

This mode is not available if the **Enable file comparison in Author mode** option (on page 286) is not selected in the **Diff > Files Comparison** preferences page.

This visual mode includes unique features such as a **Tags Display Mode** drop-down button (on page 488) on the toolbar that allows you to select the amount of tags to display in this visual mode. This mode also presents differences that were made using the **Track Changes** feature (although the **Track Changes** feature is not available in the comparison tool).
Author Mode Algorithms

The visual Author comparison mode offers the following diff algorithms to compare files:

- **Auto** - Selects the most appropriate algorithm, based on the compared content and its size (selected by default).
- **XML Fast** - Comparison that works well on large files or fragments, but it is less precise than **XML Accurate**.
- **XML Accurate** - Comparison that is more precise than **XML Fast**, at the expense of speed. It compares two XML files or fragments looking for identical XML nodes.

Author Mode Second-Level Comparisons

The visual Author comparison mode automatically performs a second-level comparison for the **XML Fast** and **XML Accurate** algorithms. After the first comparison is finished, the second-level comparisons is processed on text nodes using a word-level comparison, meaning that it looks for identical words. This second-level comparison makes it easier to spot precise differences and you can merge or reject the precise modifications.

Related information

- Files Comparison Preferences Page (on page 286)
- Compare Directories Tool (on page 494)
Starting File Comparison Tool from a Command Line

The file comparison tool can be started by using command-line arguments. In the installation folder there is an executable shell (diffFiles.bat on Windows, diffFiles.sh on macOS and Linux). To specify the files to compare, you can pass command-line arguments using the following construct:

\`\`diffFiles.bat/diffFiles.sh [path to left file] [path to right file] [path to 3-way base file].\`

If three files are specified, the tool will start in the 3-way comparison mode (on page 477). If only two files are specified, the tool will start in the 2-way comparison mode (on page 474). The first specified file will be added to the left panel in the comparison tool, the second file to the right panel, and the optional third file will be the base (ancestor) file used for a 3-way comparison. If you pass only one argument, you are prompted to manually choose another file.

If you want to launch the file comparison tool from an external application with specified files and you want the file browsing buttons at the top of both panels to be hidden, you should use the `-ext` argument as the first command. There are some additional arguments that are allowed and to see all the details for the command-line construct, type `diffFiles.bat --help` in the command line.

**Example:**

To do a 3-way comparison, the command line might look like this:

**Windows**

```
 diffFiles.bat "c:\docs\file 1" "c:\docs\file 2" c:\docs\basefile
```

**Tip:**

If there are spaces in the path names, surround the paths with quotes.

**Linux**

```
 diffFiles.sh home/file1 home/file2 home/basefile
```

**macOS**

```
 diffFiles.sh documents/file1 documents/file2 documents/basefile
```

How to Integrate the File Comparison Tool with Git

The file comparison tool can be integrated with Git clients. It requires that you configure your `.gitconfig` file and then you can simply start the tool from the command line.

To integrate the **Compare Files** tool with your Git client, follow this procedure:
1. Use one of the following methods to instruct your Git client to use the Oxygen Compare Files tool:

   ◦ **Manual Configuration** - Locate your Git user-specific configuration file (.gitconfig) and edit it with a text editor (for example, in Windows, the .gitconfig file is most likely located in your user home directory). Add (or replace) the following lines:

   ```
   [diff]
   tool = oxygendiff
   [merge]
   tool = oxygendiff
   [difftool "oxygendiff"]
   cmd = '[pathToOxygenInstallDir]/diffFiles.exe' -ext $REMOTE $LOCAL $LOCAL
   [mergetool "oxygendiff"]
   cmd = '[pathToOxygenInstallDir]/diffFiles.exe' -ext $LOCAL $REMOTE $BASE $MERGED
   trustExitCode = true
   [difftool]
   prompt = false
   ```

   **Note:**
   For macOS, the `cmd` lines would start with something like: `sh "/Applications/Oxygen XML Editor/diffFiles.sh"`. For Linux, the `cmd` lines would start with something like: `sh "/Oxygen XML Editor/diffFiles.sh"`.

   **Tip:**
   On Redhat 7, the following command would work, where the whole command is quoted and then inside that, the path to `diffFiles.sh` is quoted:

   ```
   [difftool "oxygendiff"]
   cmd = "'/home/user/Oxygen XML Editor 21/diffFiles.sh'" -ext $REMOTE $LOCAL
   $LOCAL
   [mergetool "oxygendiff"]
   cmd = "'/home/user/Oxygen XML Editor 21/diffFiles.sh'" -ext $LOCAL $REMOTE $BASE
   $MERGED trustExitCode = true
   ```

   ◦ **Command Line Configuration** - To automatically configure the .gitconfig file, you can run the following commands from a command line:

   ```
   git config --global diff.tool oxygendiff
   git config --global difftool.oxygendiff.cmd '[Oxygen install dir]/diffFiles.exe -ext $REMOTE $LOCAL $LOCAL'
   git config --global merge.tool oxygendiff
   git config --global mergetool.oxygendiff.cmd '[Oxygen install dir]/diffFiles.exe'
   ```
For macOS, the Oxygen file comparison tool would be specified in the second and fourth commands with something like: `sh "/Applications/Oxygen XML Editor/diffFiles.sh"`. For Linux, it would be something like: `sh "/Oxygen XML Editor/diffFiles.sh"`.

2. To start the Compare Files tool and see a comparison of changes for a particular file, run the following command from a command line:

```
git difftool [PathToFile]
```

Tip:

If the file you want to compare has conflicts, you can start the Compare Files tool as a merge conflict resolution tool by running the following command:

```
git mergetool [PathToFile]
```

For more information about the Git difftool syntax, see https://git-scm.com/docs/git-difftool.

For more information about the Git mergetool syntax, see https://git-scm.com/docs/git-mergetool.

How to Integrate the File Comparison Tool with Sourcetree

The file comparison tool can be integrated with Sourcetree so that you can use it to compare changes. The advantages of doing this include:

- The Oxygen Compare Files tool presents the files side-by-side and makes it much easier to determine real changes.
- The Oxygen Compare Files tool includes XML comparison algorithms.
- The Oxygen Compare Files tool includes various options for configuring the comparison.
- The Oxygen Compare Files tool allows you to navigate through changes.

To integrate the Compare Files tool with Sourcetree, follow this procedure, depending on your operating system:

**Windows**

1. In Sourcetree, go to Tools > Options.
2. Go to the Diff tab.
3. In the External Diff/Merge section, configure the settings as follows:
   - External Diff Tool - Select Custom.
   - Diff Command - Enter the path of the Oxygen `diffFile.exe` file (for example: `c:\Programs\Oxygen XML Editor 19\diffFiles.exe`).
• **Arguments** - Enter `-ext $REMOTE $LOCAL $LOCAL`.
• **Merge Tool** - Select **Custom**.
• **Diff Command** - Enter the path of the *Oxygen diffFile.exe* file (for example:`c:\Programs\Oxygen XML Editor 19\diffFiles.exe`).
• **Arguments** - Enter `-ext $LOCAL $REMOTE $BASE $MERGED`.

4. Click **OK**.

**Result:** In Sourcetree, you can now compare file changes with the *Oxygen Compare Files* tool by simply selecting **External Diff** from the contextual menu, **Actions** menu, or **Ctrl+D**.

**macOS**

1. In Sourcetree, go to **Sourcetree > Preferences**.
2. Go to the **Diff** tab.
3. In the **External Diff/Merge** section, configure the settings as follows:
   • **External Diff Tool** - Select **Custom**.
   • **Diff Command** - Enter a command-line argument to launch the *Oxygen diffFiles.sh* file (for example: `sh"/Applications/Oxygen XML Editor/diffFiles.sh"`).
   • **Arguments** - Enter `-ext $REMOTE $LOCAL $LOCAL`.
   • **Merge Tool** - Select **Custom**.
   • **Diff Command** - Enter a command-line argument to launch the *Oxygen diffFiles.sh* file (for example: `sh"/Applications/Oxygen XML Editor/diffFiles.sh"`).
   • **Arguments** - Enter `-ext $LOCAL $REMOTE $BASE $MERGED`.
4. Close the preferences dialog box.

**Result:** In Sourcetree, you can now compare file changes with the *Oxygen Compare Files* tool by simply selecting **External Diff** from the contextual menu or **Actions** menu.

**Toolbar and Contextual Menu Actions of the Compare Files Tool**

The toolbar of the **Compare Files** tool contains operations that can be performed on the source and target files or XML fragments. Many of the actions are also available in the contextual menu.

![Compare Toolbar](Image)

The following actions are available:

**Algorithm**

This drop-down menu allows you to select one of the following diff algorithms (depending on whether it is a two-way or three-way comparison):
• **Auto** - Selects the most appropriate algorithm, based on the compared content and its size (selected by default).

• **Characters** - Computes the differences at character level, meaning that it compares two files or fragments looking for identical characters. This algorithm is not available when the file comparison is in **Author** comparison mode.

• **Words** - Computes the differences at word level, meaning that it compares two files or fragments looking for identical words. This algorithm is not available when the file comparison is in **Author** comparison mode.

• **Lines** - Computes the differences at line level, meaning that it compares two files or fragments looking for identical lines of text. This algorithm is not available when the file comparison is in **Author** comparison mode.

• **Syntax Aware** - Computes differences for the file types or fragments known by Oxygen XML Editor, taking the syntax (the specific types of tokens) into consideration. This algorithm is not available when the file comparison is in **Author** comparison mode.

• **XML Fast** - Comparison that works well on large files or fragments, but it is less precise than **XML Accurate**.

• **XML Accurate** - Comparison that is more precise than **XML Fast**, at the expense of speed. It compares two XML files or fragments looking for identical XML nodes.

---

**Diff Options**

Opens the **Files Comparison** preferences page (on page 286) where you can configure various options.

**Three-Way Comparison**

Toggle action that allows you to perform a three-way comparison between the two files displayed in the two editing panes and a base (ancestor) file.

**Perform Files Differencing**

Looks for differences between the two files displayed in the left and right side panels.

**Synchronized scrolling**

Toggles synchronized scrolling on or off so that a selected difference can be seen on both sides of the application window. This option is on by default.

**Ignore Whitespaces**

Enables or disables the whitespace ignoring feature. Ignoring whitespace means that before performing the comparison, the application normalizes the content and trims its leading and trailing whitespaces. This option is not available when in the **Author** comparison mode.

**Format and Indent Both Files** (Ctrl + Shift + P (Command + Shift + P on macOS))
Formats and indents both files before comparing them. Use this option for comparisons that contain long lines that make it difficult to spot differences. This option is not available when in the **Author** comparison mode.

**Note:**
When comparing two JSON files, the **Format and Indent Both Files** action will automatically sort the keys in both files the same to make it easier to compare.

**Tags Display Mode**

Allows you to select the amount of markup to be displayed in the **Author** visual comparison mode *(on page 481)*. You can choose between: **Full Tags with Attributes**, **Full Tags**, **Block Tags**, **Block Tags without Element Names**, **Inline Tags**, **Partial Tags**, or **No Tags**.

**Copy Change from Right to Left**

Copies the selected difference from the file in the right panel to the file in the left panel.

**Copy All Changes from Right to Left**

Copies all changes from the file in the right panel to the file in the left panel.

**Next Block of Changes** *(Ctrl + Period (Command + Period on macOS))*

Jumps to the next block of changes. This action is not available when the cursor is positioned on the last change block or when there are no changes.

**Previous Block of Changes** *(Ctrl + Comma (Command + Comma on macOS))*

Jumps to the previous block of changes. This action is not available when the cursor is positioned on the first change block or when there are no changes.

**Next Change** *(Ctrl + Shift + Period (Command + Shift + Period on macOS))*

Jumps to the next change from the current block of changes. When the last change from the current block of changes is reached, it highlights the next block of changes. This action is not available when the cursor is positioned on the last change or when there are no changes.

**Previous Change** *(Ctrl + Shift + Comma (Command + Shift + M on macOS))*

Jumps to the previous change from the current block of changes. When the first change from the current block of changes is reached, it highlights the previous block of changes. This action is not available when the cursor is positioned on the first change or when there are no changes.

**Copy All Changes from Left to Right**
Copies all changes from the file in the left panel to the file in the right panel.

**Copy Change from Left to Right**

Copies the selected difference from the file in the left panel to the file in the right panel.

**Ignore Nodes by XPath**

You can use this text field to enter an XPath expression (on page 1999) to ignore certain nodes from the comparison. It will be processed as XPath version 2.0. You can also enter the name of the node to ignore all nodes with the specified name (for example, if you want to ignore all ID attributes from the document, you could simply enter `@id`). This field is only available when comparing XML documents using the XML Fast or XML Accurate algorithms.

**Note:**

If an XPath expression is specified in the Ignore nodes by XPath option (on page 288) in the Diff / File Comparison preferences page, that one is used as a default when the application is started. If you then enter an expression in this field on the toolbar, this one will be used instead of the default. If you delete the expression from this field, neither will be used.

**First Change (Ctrl + B (Command + B on macOS))**

Jumps to the first change.

**Base**

Available for three-way comparisons (on page 477). It is the base file that will be compared with the files opened in the left and right editors. You can specify the path to the file by using the text field, its history drop-down, or the browsing actions in the **Browse** drop-down menu.

**Left-Side (Source) File**

You can specify the path to the file to be compared on the left side (source) by using the text field, its history drop-down, or the browsing actions in the **Browse** drop-down menu.

**Save**

Saves the changes made in the source (left-side) file.

**Reload**

Reloads the source (left-side) file.

**Close**

Closes the source (left-side) file.

**Right-Side (Target) File**

You can specify the path to the file to be compared on the right side (target) by using the text field, its history drop-down, or the browsing actions in the **Browse** drop-down menu.
Save
Saves the target (right-side) file.

Reload
Reloads the target (right-side) file.

Close
Closes the target (right-side) file.

Compare Files Tool Menus
The menus in the Compare Files tool contain some of the same actions that are on the toolbar, as well as some common actions that are identical to the same actions in the Oxygen XML Editor menus. The menu actions include:

File Menu

Source > Open
Browses for a file that will be displayed in the left panel.

Source > Open URL
Browses for a remote file that will be displayed in the left panel.

Source > Open File from Archive
Browses an archive for a file that will be displayed in the left panel.

Source > Reload
Reloads the file in the left panel.

Source > Save
Saves the changes made to the file in the left panel.

Source > Save As
Allows you to choose a destination to save the file in the left panel.

Source > Close
Closes the file in the left panel.

Target > Open
Browses for a file that will be displayed in the right panel.

Target > Open URL
Browses for a remote file that will be displayed in the right panel.

Target > Open File from Archive
Browses an archive for a file that will be displayed in the right panel.
Options Menu

Target > Reload

Reloads the file in the right panel.

Target > Save

Saves the changes made to the file in the right panel.

Target > Save As

Allows you to choose a destination to save the file in the right panel.

Target > Close

Closes the file in the right panel.

Base > Open

Browses for a file that will be compared with both files in a three-way comparison (on page 477).

Base > Open URL

Browses for a remote file that will be compared with both files in a three-way comparison (on page 477).

Base > Open File from Archive

Browses an archive for a file that will be compared with both files in a three-way comparison (on page 477).

Save Results as HTML (Available in Text mode only)

Generates an HTML file that contains detailed information about the comparison result. See an example of what the generated report look like in the Generate HTML Report for Directory Comparison topic (on page 511).

Close (Ctrl + W (Command + W on macOS))

Closes the application.

Edit Menu

Cut

Cut the selection from the currently focused editor panel to the clipboard.

Copy

Copy the selection from the currently focused editor panel to the clipboard.

Paste

Paste content from the clipboard into the currently focused editor panel.

Select all

Selects all content in the currently focused editor panel.

Undo
Undo changes in the currently focused editor panel.

Redo

Redo changes in the currently focused editor panel.

Find Menu

Find/Replace

Perform find/replace operations in the currently focused editor panel.

Find Next

Go to the next match using the same options as the last find operation. This action runs in both editor panels.

Find Previous

Go to the previous match using the same options as the last find operation. This action runs in both editor panels.

Compare Menu

Three-Way Comparison

Toggle action that allows you to perform a three-way comparison between the two files displayed in the two editing panes and a base (ancestor) file.

Perform Files Differencing

Looks for differences between the two files displayed in the left and right side panels.

Next Block of Changes (Ctrl + Period (Command + Period on macOS))

Jumps to the next block of changes. This action is not available when the cursor is positioned on the last change block or when there are no changes.

Note:

A change block groups one or more consecutive lines that contain at least one change.

Previous Block of Changes (Ctrl + Comma (Command + Comma on macOS))

Jumps to the previous block of changes. This action is not available when the cursor is positioned on the first change block or when there are no changes.

Next Change (Ctrl + Shift + Period (Command + Shift + Period on macOS))

Jumps to the next change from the current block of changes. When the last change from the current block of changes is reached, it highlights the next block of changes. This action is not available when the cursor is positioned on the last change or when there are no changes.

Previous Change (Ctrl + Shift + Comma (Command + Shift + M on macOS))
Jumps to the previous change from the current block of changes. When the first change from the current block of changes is reached, it highlights the previous block of changes. This action is not available when the cursor is positioned on the first change or when there are no changes.

**Last Change (Ctrl + E (Command + E on macOS))**
Jumps to the last change.

**First Change (Ctrl + B (Command + B on macOS))**
Jumps to the first change.

**Copy All Changes from Right to Left**
Copies all changes from the file in the right panel to the file in the left panel.

**Copy All Changes from Left to Right**
Copies all changes from the file in the left panel to the file in the right panel.

**Copy Change from Right to Left**
Copies the selected difference from the file in the right panel to the file in the left panel.

**Copy Change from Left to Right**
Copies the selected difference from the file in the left panel to the file in the right panel.

**Show Word Level Details**
Provides a word-level comparison of the selected change.

**Show Character Level Details**
Provides a character-level comparison of the selected change.

**Format and Indent Both Files (Ctrl + Shift + P (Command + Shift + P on macOS))**
Formats and indents both files before comparing them. Use this option for comparisons that contain long lines that make it difficult to spot differences.

**Note:**
When comparing two JSON files, the **Format and Indent Both Files** action will automatically sort the keys in both files the same to make it easier to compare.

**Options Menu**

**Preferences**
Opens the preferences dialog box that includes numerous pages of options that can be configured.

**Menu Shortcut Keys**
Opens the **Menu Shortcut Keys** option page where you can configure keyboard shortcuts available for menu items.
Reset Global Options

Resets options to their default values. Note that this option appears only when the tool is executed as a stand-alone application.

Import Global Options

Allows you to import an options set that you have previously exported.

Export Global Options

Allows you to export the current options set to a file.

Help Menu

Help (F1)

Opens a Help dialog box that displays the User Manual at a section that is appropriate for the context of the current cursor position.

Use Online Help

If this option is selected, when you select Help or press F1 while hovering over any part of the interface, Oxygen XML Editor attempts to open the help documentation in online mode. If this option is not selected or an internet connection fails, the help documentation is opened in offline mode.

Report problem

Opens a dialog box that allows the user to write the description of a problem that was encountered while using the application. You can change the URL where the reported problem is sent by using the com.oxygenxml.report.problems.url system property. The report is sent in XML format through the report parameter of the POST HTTP method.

Support Center

Opens the Oxygen XML Editor Support Center web page in a browser.

Compare Directories Tool

The Compare Directories tool can be used to compare and manage changes to files and folders within the structure of your directories. The utility is available from the Tools menu or can be opened as a stand-alone application from the Oxygen XML Editor installation folder (diffDirs.exe).
Starting the Tool from a Command Line

The directory comparison tool can also be started by using command-line arguments. In the installation folder there is an executable shell (`diffDirs.bat` on Windows, `diffDirs.sh` on macOS and Linux). To specify the directories to compare, you can pass command-line arguments using the following construct: `diffDirs.bat/diffDirs.sh [directory path 1] [directory path 2].` If you pass only one argument, you are prompted to manually choose the second directory or archive.

**Example:**

To do a comparison between two directories, the command line would look like this:

**Windows**

```
diffDirs.bat "c:\documents new" "c:\documents old"
```

**Tip:**

If there are spaces in the path names, surround the paths with quotes.

**Linux**

```
diffDirs.sh home/documents1 home/documents2
```

**macOS**

```
diffDirs.sh documents1 documents2
```
Directory Comparisons

To perform a directory comparison, follow these steps:

1. Select a folder in the left panel and the folder you want to compare it to in the right panel. You can specify the path by using the text field, the history drop-down, or the Browse for local directory action in the Browse drop-down menu.

   **Step Result:** The selected directory structures are opened in the two side-by-side panels.

2. To highlight the differences between the two folders, click the Perform Directories Differencing button from the toolbar.

3. You can also use the Diff Options button to access the Directories Comparison preferences page (on page 289) where you can configure various options.

To compare the content of two archives, follow these steps:

1. Use the Browse for archive file action in the Browse drop-down menu to select the archives in the left and right panels.

2. By default, the supported archives are not treated as directories and the comparison is not performed on the files inside them. To make Oxygen XML Editor treat supported archives as directories, select the Look in archives option (on page 290) in the Directories Comparison preferences page.

3. To highlight the differences, click the Perform Directories Differencing button from the toolbar.

The directory comparison results are presented using two tree-like structures showing the files and folders, including their name, size, and modification date. A column that contains graphic symbols separates the two tree-like structures. The graphic symbols can be one of the following:

- An X symbol, when a file or a folder exists in only one of the compared directories.
- A ≠ symbol, when a file exists in both directories but the content differs. The same sign appears when a collapsed folder contains differing files.

The color used for the symbol and the directory or file name can be customized in the Directories Comparison / Appearance preferences page (on page 290). You can double-click lines marked with the ≠ symbol to open a Compare Files window, which shows the differences between the two files.

The directories that contain files that differ are expanded automatically so that you can focus directly on the differences. You can merge the contents of the directories by using the copy actions. If you double-click (or press Enter) on a line with a pair of files, Oxygen XML Editor starts a file comparison (on page 474) between the two files, using the Compare Files tool.

**Related information**

- Compare Files Tool (on page 474)
- Compare Directories Script (on page 3206)
Toolbar and Contextual Menu Actions of the Compare Directories Tool

The toolbar of the Compare Directories tool contains operations that can be performed on the compared directory structure. Some of the toolbar actions are also available in the contextual menu.

**Figure 671. Compare toolbar**

![Compare toolbar](image)

### Toolbar Actions

**Perform Directories Differencing**

Looks for differences between the two directories displayed in the left and right side of the application window.

**Perform Files Differencing**

Opens the Compare Files tool *(on page 474)* that allows you to compare the currently selected files.

**Copy Change from Right to Left**

Copies the selected change from the right side to the left side (if there is no file/folder in the right side, the left file/folder is deleted).

**Copy Change from Left to Right**

Copies the selected change from the left side to the right side (if there is no file/folder in the left side, the right file/folder is deleted).

**Binary Compare**

Performs a byte-level comparison on the selected files.

**Diff Options**

Opens the Directory Comparison preferences page *(on page 289)* where you can configure various options.

**Show Only Modifications**

Displays a more uncluttered file structure by hiding all identical files.

**Save Results as HTML**

Generates an HTML file that contains detailed information about the comparison result.

### File and folder filters

Differences can be filtered using three combo boxes: *Include files*, *Exclude files*, and *Exclude folders*. They come with predefined values and are editable to allow custom values. All of them accept multiple comma-separated values and the * and ? wildcards. For example, to filter out
all JPEG and GIF image files, edit the Exclude files filter box to read *.jpeg, *.png. Each filter includes a drop-down menu with the latest 15 filters applied.

**Contextual Menu Actions**

- **Perform Files Differencing**
  Opens the Compare Files tool (on page 474) that allows you to compare the currently selected files.

- **Binary Compare**
  Performs a byte-level comparison on the selected files.

- **Copy Change from Right to Left**
  Copies the selected difference from the file in the right panel to the file in the left panel.

- **Copy Change from Left to Right**
  Copies the selected difference from the file in the left panel to the file in the right panel.

- **Open**
  If the action is invoked on a file, the selected file is opened in Oxygen XML Editor. If the action is invoked on a directory, the selected directory is opened in the default file browser for your particular operating system.

- **Open in System Application**
  Opens the selected file in the system application that is associated with that type of file. The action is available when launching the Compare Directories tool from the Tools menu in Oxygen XML Editor.

- **Show in Explorer**
  Opens the default file browser for your particular operating system with the selected file highlighted.

**Compare Directories Tool Menus**

The menus in the Compare Directories tool contain some of the same actions that are on the toolbar, as well as some common actions that are identical to the same actions in the Oxygen XML Editor menus. The menu actions include:

**File Menu**

- **Save Results as HTML**
  Generates an HTML file that contains detailed information about the comparison result. See an example of what the generated report look like in the Generate HTML Report for Directory Comparison topic (on page 511).

- **Close (Ctrl + W (Command + W on macOS))**
Closes the application.

**Compare Menu**

**Perform Directories Differencing**

Looks for differences between the two directories displayed in the left and right side of the application window.

**Perform Files Differencing**

Opens the Compare Files tool (on page 474) that allows you to compare the currently selected files.

**Copy Change from Right to Left**

Copies the selected change from the right side to the left side (if there is no file/folder in the right side, the left file/folder is deleted).

**Copy Change from Left to Right**

Copies the selected change from the left side to the right side (if there is no file/folder in the left side, the right file/folder is deleted).

**Options Menu**

**Preferences**

Opens the preferences dialog box that includes numerous pages of options that can be configured.

**Menu Shortcut Keys**

Opens the Menu Shortcut Keys option page where you can configure keyboard shortcuts available for menu items.

**Reset Global Options**

Resets options to their default values. Note that this option appears only when the tool is executed as a stand-alone application.

**Import Global Options**

Allows you to import an options set that you have previously exported.

**Export Global Options**

Allows you to export the current options set to a file.

**Help Menu**

**Help (F1)**

Opens a Help dialog box that displays the User Manual at a section that is appropriate for the context of the current cursor position.

**Use Online Help**
If this option is selected, when you select Help or press F1 while hovering over any part of the interface, Oxygen XML Editor attempts to open the help documentation in online mode. If this option is not selected or an internet connection fails, the help documentation is opened in offline mode.

**Report problem**

Opens a dialog box that allows the user to write the description of a problem that was encountered while using the application. You can change the URL where the reported problem is sent by using the `com.oxygenxml.report.problems.url` system property. The report is sent in XML format through the `report` parameter of the POST HTTP method.

**Support Center**

Opens the Oxygen XML Editor Support Center web page in a browser.

**Compare Images**

You can use the **Compare Directories** tool to compare images. If you double-click a line that contains two different images, the **Compare images** window is displayed. This dialog box presents the images in the left and right sides, scaled to fit the available view area. You can use the contextual menu actions to scale the images to their original size or scale them down to fit in the view area.

The supported image types are: GIF, JPG, JPEG, PNG, and BMP.

**Compare Directories Against a Base (3-Way) Tool**

The **Compare Directories Against a Base (3-way)** tool allows you to perform three-way comparisons on directories to help you identify and merge changes between multiple modifications of the same directory structure. It is especially helpful for teams that have multiple authors contributing documents to the same directory system. It offers information about conflicts and changes, and includes actions to easily merge, accept, overwrite, or ignore changes to the directory system.

**How to Perform 3-Way Directory Comparisons**

To perform a 3-way directories comparison, follow these steps:

1. Select **Compare Directories Against a Base (3-way)** from the **Tools** menu.

   **Step Result:** This opens a dialog box that allows you to select the 3 file sets that will be used for the comparison.
Figure 672. Compare Directories Against a Base File Set Chooser

2. Select the file sets to be compared:
   - **Base directory** - This is the original (base) file set before any modifications were made by you or others.
   - **Directory with your changes** - This is the file set with changes that you have made. This file set will be displayed in the left panel in the comparison tool.
   - **Directory with changes made by others** - This is the file set with changes made by others that you want to merge with your changes. This file set will be displayed in the right panel in the comparison tool.

3. Click the **Compare** button to compare the file sets and open the comparison and merge tool.

4. Use the features and actions described in the next section to identify and merge the changes.
3-Way Directory Comparison and Merge Tool

Figure 673. Comparison and Merge Tool

The 3-way directory comparison and merge tool includes the following information, features, and actions:

**Number of Changes and Conflicts**

The first thing you see in the top-left corner of the tool is the grand total of all the changes made by others, changes made by you, and the number of conflicts.

**Filter Buttons**

In the top-right corner you can use the toggle buttons to filter the list of modifications:

- **Show all files**
  Use this button to show all modified and unmodified files, as well as conflicts.

- **Show only files modified by you and others**
  Filters the list to show all files that have been modified, including conflicts.

- **Show only files modified by others**
  Filters the list to only show the files that were modified by others.

- **Show only files modified by you**
  Filters the list to only show the files that were modified by you.
Show only conflicting files

Filters the list to only show files that contain conflicts.

List of Files Panel

This panel shows the list of files in the compared file sets based upon the filter button that is selected. This panel includes the following sortable columns:

- **Name** - The file names.
- **Status** - An icon that represents the file status. Red icons indicate some sort of conflict. Gray icons indicate modifications made by you. Blue icons indicate modifications made by others.
- **Description** - A description of the file status.
- **Merge Action** - This column provides a drop-down menu for each file that allows you to choose some merge actions depending upon its status. A default action is always set to **Automatically merge** the changes made by others with your changes. If there is a conflict, the default is **<Select action>** and you are required to make a selection. Click this column to access the drop-down menu where you can make a selection. The same actions are available in the contextual menu.

Tip:

If the solution proposed in the Merge Action column for any particular file is not satisfactory, you can change it directly in that column (even if that file is not selected) without automatically re-triggering the comparison (except for in certain cases where re-triggering the comparison is necessary).

You can click a file to open it in the file comparison panel (the file from your file set is shown in the left panel while the file from the file set with changes made by others is shown in the right panel). For image files, the comparison panel shows a preview of the image. For other binary files, a preview is not available and you will just see its status.

File Comparison Panels

If you click a file in the top panel, the file is opened in this file comparison section. The file from your file set is shown in the left panel and the file from the other file set is shown in the right panel.

Note:

If Oxygen XML Editor does not recognize the file type, a dialog box will be displayed that allows you to select the type of editor you want it to be associated with for this comparison (if you want Oxygen XML Editor to remember this association, you can select the Associate file type with editor option at the bottom of the dialog box).
File Path

The first thing you see in this panel is the file path where merge actions will be applied if you make changes.

× Close

Closes the file comparison panel.

Algorithm Drop-down Menu

This drop-down menu allows you to select one of the following diff algorithms to be used for file comparisons:

• Auto - Selects the most appropriate algorithm, based on the compared content and its size (selected by default).
• Lines - Computes the differences at line level, meaning that it compares two files or fragments looking for identical lines of text. This algorithm is not available when the file comparison is in Author comparison mode.
• XML Fast - Comparison that works well on large files or fragments, but it is less precise than XML Accurate.
• XML Accurate - Comparison that is more precise than XML Fast, at the expense of speed. It compares two XML files or fragments looking for identical XML nodes.

Diff Options

Opens the Files Comparison preferences page (on page 286) where you can configure various options.

Perform Files Differencing

Looks for differences between the two files displayed in the left and right side panels.

Synchronized scrolling

Toggles synchronized scrolling. When toggled on, a selected difference can be seen in both panels.

Ignore Whitespaces

Enables or disables the whitespace ignoring feature. Ignoring whitespace means that before performing the comparison, the application normalizes the content and trims its leading and trailing whitespaces. This option is not available when the file comparison is in Author mode.

Tags Display Mode

Allows you to select the amount of markup to be displayed in the Author visual mode. You can choose between: Full Tags with Attributes, Full Tags,
Block Tags, Block Tags without Element Names, Inline Tags, Partial Tags, or No Tags.

Copy Change from Right to Left
Copies the selected difference from the file in the right panel to the file in the left panel.

Copy All Changes from Right to Left
Copies all changes from the file in the right panel to the file in the left panel.

Next Block of Changes (Ctrl + Period (Command + Period on macOS))
Jumps to the next block of changes. This action is not available when the cursor is positioned on the last change block or when there are no changes.

Note:
A change block groups one or more consecutive lines that contain at least one change.

Previous Block of Changes (Ctrl + Comma (Command + Comma on macOS))
Jumps to the previous block of changes. This action is not available when the cursor is positioned on the first change block or when there are no changes.

Next Change (Ctrl + Shift + Period (Command + Shift + Period on macOS))
Jumps to the next change from the current block of changes. When the last change from the current block of changes is reached, it highlights the next block of changes. This action is not available when the cursor is positioned on the last change or when there are no changes.

Previous Change (Ctrl + Shift + Comma (Command + Shift + M on macOS))
Jumps to the previous change from the current block of changes. When the first change from the current block of changes is reached, it highlights the previous block of changes. This action is not available when the cursor is positioned on the first change or when there are no changes.

First Change (Ctrl + B (Command + B on macOS))
Jumps to the first change.

Left-Side File (Your changes)
Above the panel you can see the file path and the following two buttons:

Save
Saves changes made to the file.

Reload
Reloads the file.

**Right-Side File (Changes made by others)**

Above the panel you can see the file path and the following two buttons:

![Reload](reload_icon.png)

Reloads the file.

### Displaying Changes in the File Comparison Panels

The line numbers on each side and colored marks on the right-side vertical stripe help you to quickly identify the locations of the differences. Adjacent changes are grouped into blocks of changes.

**Figure 674. File Comparison Panels**

The differences are also highlighted in several colors, depending on the type of change, and dynamic lines connect the compared fragments in the middle section between the two panes. The highlighting colors can be customized in the Files Comparison / Appearance preferences page (on page 288), but the default colors and their shades mean the following:

- **Pink** - Identifies modifications on either side.
- **Gray** - Identifies an addition of a node in the left side (your outgoing changes).
- **Blue** - Identifies an addition of a node in the right side (incoming changes).
- **Lighter Shade** - Identifies blocks of changes that can be merged in their entirety.
- **Darker Shade** - Identifies specific changes within the blocks that can be merged more precisely.

### Direct Editing Actions in the File Comparison Panels

In addition to selecting merge actions from the drop-down menus in the Merge Action column in the top panel, you can also edit the files directly in the left pane (your local changes). The two editors are constantly synchronized and the differences are refreshed when you save the modified document ([Save](#) button or [Ctrl]+[S]) or when you click the [Perform File Differencing](#) button.

A variety of actions are available in the contextual menu in both editing panes. The tool also includes some inline actions to help you merge, copy, or remove changes. When you select a change, the following inline action widgets are available, depending on the type of change:

![Append right change to left](append_right_change_to_left_icon.png)

Append right change to left
Copies the content of the selected change from the right side and appends it on the left side.

**Copy change from right to left**

Replaces the content of a change in the left side with the content of the change in the right side.

**Remove change**

Removes the change from the left side.

Anytime you save manual changes (Save button or Ctrl+S), the selection in the Merge Action column in the top panel automatically changes to Use merged and a copy of the original file is kept so that you can revert to the original file if necessary. To discard your manual changes and revert to your original changes, select a different action in the Merge Action drop-down menu.

**Open Merged Files**

If you select this option, all the files that will be modified by the merge operation will be opened in the editor after the operation is finished.

**Applying Changes**

When you click the Apply button, all the merge actions you have selected and the changes you have made will be processed.

If there are unresolved conflicts (conflicts where no merge action is selected in the Merge Action drop-down menu), a dialog box will be displayed that allows you to choose how to solve the conflicts. You can choose between the following:

- **Keep your changes** - If you select this option and then click Apply, your local changes will be preserved for the unresolved conflicts.
- **Overwrite your changes** - If you select this option and then click Apply, your local changes will be overwritten with the changes made by others, for the unresolved conflicts.
- **Cancel** - You can click the Cancel button to go back to the merge tool to resolve the conflicts individually.

**Canceling Changes**

If you click the Cancel button at the bottom of the merge tool, all the changes you made in the tool will be lost.

**Author Visual Mode**

The Comparison and Merge tool includes an Author mode that displays the files in a visual mode similar to the Author editing mode in Oxygen XML Editor/Author. This makes it easier to see how the compared changes will look in the final output. This visual mode is available when the compared files are detected as being XML. To determine whether the files are initially opened in the merge tool's Text or Author mode, it
detects the **Initial Edit Mode** in the *Document Type Association* configuration (on page 141) and the mode the files were last opened in *Oxygen XML Editor/Author*.

**Note:**
This mode is not available if the **Enable file comparison in Author mode** option (on page 286) is not selected in the **Diff > Files Comparison** preferences page.

This visual mode includes unique features such as a **Tags Display Mode** drop-down button (on page 2710) on the toolbar that allows you to select the amount of tags to display in this visual mode. This mode also presents differences that were made using the **Track Changes** feature (although the **Track Changes** feature is not available in the comparison tool).

![Figure 675. File Comparison Tool - Author Mode](image)

**Author Mode Algorithms**

The visual **Author** mode offers the following diff algorithms to compare files:

- **Auto** - Selects the most appropriate algorithm, based on the compared content and its size (selected by default).
- **XML Fast** - Comparison that works well on large files or fragments, but it is less precise than **XML Accurate**.
- **XML Accurate** - Comparison that is more precise than **XML Fast**, at the expense of speed. It compares two XML files or fragments looking for identical XML nodes.
**Author Mode Second-Level Comparisons**

The visual **Author** mode automatically performs a second-level comparison for the **XML Fast** and **XML Accurate** algorithms. After the first comparison is finished, the second-level comparison is processed on text nodes using a word level comparison, meaning that it looks for identical words. This second-level comparison makes it easier to spot precise differences and you can merge or reject the precise modifications.

**Related information**
- Compare Directories Tool *(on page 494)*
- Compare Files Tool *(on page 474)*

**Generate HTML Report for Directory Comparison**

The **Generate HTML report for directory comparison** tool can be used to generate a report in the form of an HTML file that contains the results of a directory comparison (for either 2-way or 3-way comparisons). The **Generate HTML report for directory comparison** action for invoking the tool can be found in the **Tools** menu. It opens a dialog box where you can specify the directories to compare as well as some other options.

![Generate HTML Report for Directory Comparison Dialog Box](image)

The **Generate HTML report for directory comparison** dialog box contains the following options:

**Base directory**

Specifies the path of the base directory that the other two directories will be compared against in a 3-way comparison. This field should be left empty for 2-way comparisons.
First directory

Specifies the path of the first directory to be included in the comparison.

Second directory

Specifies the path of the second directory to be included in the comparison.

Diff options

Specifies which option set to use for generating the comparison report. If you choose Use the current settings from Preferences, the options set in the Directories Comparison preferences page (on page 289) and the include/exclude filter options in the Compare Directories tool (on page 497) are taken into account when generating the comparison result. You can also click the Diff options button to open the Directories Comparison preferences page where you can see or modify the current settings. If you choose Use the default settings, the default values for all settings are used.

Generate additional file comparison reports

Generates further comparison reports for all non-binary modified file pairs and provides links to them in the main report (in the middle cells of the results table). See the example below (on page 2717). These additional file comparison reports are saved to a directory that will have the same parent directory and the same name as the output file provided, suffixed by “-OXY-FC-REPORTS”. The links created in the main report are relative to this directory. If the main HTML report is later copied or moved to another location, to retain full functionality in the browser, the directory with the additional file comparison reports must also be copied/moved to the same location.

Note:

Generating additional file comparison reports could significantly increase the execution time. A progress tracker for the whole operation is available.

Output file

Specifies the path for an output file to save the comparison results file.

Open in Browser/System Application

Opens the comparison results file in the browser or system application that is associated with HTML files.

After clicking the Generate report button, a report in the form of an HTML file is generated with details about the comparison results.
Figure 677. HTML Report for Directory Comparison

<table>
<thead>
<tr>
<th>File name</th>
<th>Size</th>
<th>Modified</th>
<th>File name</th>
<th>Size</th>
<th>Modified</th>
</tr>
</thead>
<tbody>
<tr>
<td>concepts/autumnFlowers.dita</td>
<td>1161</td>
<td>2021-07-06 01:19:12</td>
<td>concepts/autumnFlowers.dita</td>
<td>1143</td>
<td>2021-07-16 08:52:21</td>
</tr>
<tr>
<td>concepts/glossaryGenus.dita</td>
<td>571</td>
<td>2021-07-16 06:56:17</td>
<td>concepts/glossaryGenus.dita</td>
<td>577</td>
<td>2021-07-06 01:49:12</td>
</tr>
<tr>
<td>concepts/glossaryPamela.dita</td>
<td>481</td>
<td>2021-07-16 08:53:54</td>
<td>concepts/glossaryPamela.dita</td>
<td>495</td>
<td>2021-07-06 01:49:12</td>
</tr>
<tr>
<td>images/Gerbera.png</td>
<td>10131</td>
<td>2021-07-06 01:19:12</td>
<td>images/Gerbera.png</td>
<td>22776</td>
<td>2021-07-06 05:56:25</td>
</tr>
<tr>
<td>publishing/flowers README.txt</td>
<td>127</td>
<td>2021-07-06 01:19:12</td>
<td>publishing/flowers README.txt</td>
<td>127</td>
<td>2021-07-06 01:49:12</td>
</tr>
<tr>
<td>tasks/gardenPreparation.dita</td>
<td>2275</td>
<td>2021-07-06 01:49:12</td>
<td>tasks/gardenPreparation.dita</td>
<td>2261</td>
<td>2021-07-15 12:21:02</td>
</tr>
<tr>
<td>topics/flowers/gerbera.dita</td>
<td>2432</td>
<td>2021-07-16 08:38:28</td>
<td>topics/flowers/gerbera.dita</td>
<td>2468</td>
<td>2021-07-16 08:26:13</td>
</tr>
<tr>
<td>topics/mary/</td>
<td></td>
<td></td>
<td>topics/mary/</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 678. Example of an Additional File Comparison Report

Resources

For more information about how to generate HTML comparison reports, watch our video demonstration:

https://www.youtube.com/embed/6jPccHKUNNk
Syncro SVN Client (Deprecated)

The Syncro SVN Client is a client application for the Apache Subversion™ version control system, compatible with Subversion 1.6, 1.7, and 1.8 servers. It manages files and directories that change over time and are stored in a central repository. The version control repository is much like an ordinary file server, except that it remembers every change ever made to your files and directories. This allows you to access older versions of your files and examine the history of how and when your data changed.

To start Syncro SVN Client, go to Tools > SVN Client.

Attention:
The Syncro SVN Client that comes bundled with Oxygen XML Editor is considered deprecated as of version 21.0.

Main Window

This section explains the main window of Syncro SVN Client.

Views

The main window consists of the following views:

- **Repositories view (on page 2808)** - Allows you to define and manage Apache Subversion™ repository locations.
- **Working Copy view (on page 2813)** - Allows you to manage with ease the content of the working copy.
- **History view (on page 2828)** - Displays information (author name, revision number, commit message) about the changes made to a resource during a specified period of time.
- **Editor view (on page 2834)** - Allows you to edit various types of text files, with full syntax-highlight.
- **Annotations view (on page 2835)** - Displays a list with information regarding the structure of a document (author and revision for each line of text).
- **Compare view (on page 2837)** - Displays the differences between two revisions of a text file from the working copy.
- **Image Preview panel (on page 2841)** - Allows you to preview standard image files supported by Syncro SVN Client: JPG, GIF and PNG.
- **Compare Images view (on page 2841)** - Displays two images side by side.
- **Properties view (on page 2841)** - Displays the SVN properties of a resource under version control.
• **Console view (on page 2843)** - Displays information about the currently running operation, similar with the output of the Subversion command-line client.

• **Dynamic Help view (on page 2843)** - Shows information about the currently selected view.

The main window's status bar presents in the left side the operation in progress or the final result of the last performed action. In the right side there is a progress bar for the running operation and a stop button to cancel the operation.

### SVN Main Menu

The main menu of the Syncro SVN Client is composed of the following menus:

**File Menu**

New submenu:

**New File**

This operation creates a new file as a child of the selected folder from the **Repositories view (on page 2808)** tree or the **Working Copy view (on page 2813)** tree, depending on the view that was last used. Note that for the **Working Copy view (on page 2813)**, the file is added to **version control only** if the selected folder is under **version control**.

**New Folder (Ctrl + Shift + F (Command + Shift + F on macOS))**

This operation creates a new folder as a child of the selected folder from the **Repositories view (on page 2808)** tree or the **Working Copy view (on page 2813)** tree, depending on the view that was last used. Note that for the **Working Copy view (on page 2813)**, the file is added to **version control only** if the selected folder is under **version control**.

**New External Folder (Ctrl + Shift + W (Command + Shift + W on macOS))**

This operation allows you to add a new external definition on the selected folder. An external definition is a mapping of a local directory to a **URL of a versioned directory (on page 2848)**, and ideally a particular revision, stored in the **svn:externals** property of the selected folder.

Tip:

You can specify a particular revision of the external item by using a **peg revision (on page 2850)** at the end of the URL (for example, URL@rev1234). You can also use peg revisions to access external items that were deleted, moved, or replaced.
The URL used in the external definition format can be relative. You can specify the repository URL that the external folder points to by using one of the following relative formats:

- `../` - Relative to the URL of the directory that the `svn:externals` property is set.
- `^/` - Relative to the root of the repository where the `svn:externals` property is versioned.
- `//` - Relative to the scheme of the URL of the directory that the `svn:externals` property is set.
- `/` - Relative to the root URL of the server that has the `svn:externals` property versioned.

**Important:**
To change the target URL of an external definition, or to delete an external item, do the following:

1. Modify or delete the item definition found in the `svn:externals` property that is set on the parent folder.
2. For the change to take effect, use the **Update** operation on the parent folder of the external item.

**Note:**
Syncro SVN Client does not support definitions of local relative external items.

**Open (Ctrl + O (Command + O on macOS))**

This action opens the selected file in an editor where you can modify it. The action is active only when a single item is selected. The action opens a file with the internal editor or the external application associated with that file type. This action works on any file selection from the **Repositories view (on page 2808)**, **Working Copy view (on page 2813)**, **History view (on page 2828)**, or **Directory Change Set view (on page 2833)**, depending on the view that was last used to invoke it. In the case of a folder, the action opens the selected folder with the system application for folders (for example, Windows Explorer on Windows or Finder on macOS). Note that opening folders is available only for folders selected in the **Working Copy view (on page 2813)**.

**Open with (Ctrl + Shift + O (Command + Shift + O on macOS))**

Displays the **Open with** dialog box for specifying the editor where the selected file is opened. If multiple files are selected only external applications can be used to
open the files. This action works on any file selection from Repositories view (on page 2808), Working Copy view (on page 2813), History view (on page 2828), or Directory Change Set view (on page 2833), depending on the view that was last used to invoke it.

Show in Explorer/Show in Finder

Opens the parent directory of the selected working copy file and selects the file.

Save (Ctrl + S (Command + S on macOS))

Saves the local file currently opened in the editor or the Compare view.

Save as

Saves any file selected in the Repositories, History, or Directory Change Set view.

Copy URL Location (Ctrl + Alt + U (Command + Alt + U on macOS))

Copies the URL location of the resource currently selected in the Repositories view to clipboard.

Copy to

Copies the currently selected resource, either in Repositories or Working copy view, to a specified location.

Note:

This action can also be used from History and Directory Change Set views to recover older versions of a repository item.

Move to (Ctrl + M (Command + M on macOS))

Moves the currently selected resource, either in Repositories or Working copy view, to a specified location.

Rename (F2)

Renames the resource currently selected, either in Repositories or Working copy view.

Delete (Delete)

Deletes the resource currently selected either, in Repositories or Working copy view.

Locking:

• Scan for locks (Ctrl + L (Command + L on macOS)) - Contacts the repository and recursively obtains the list of locks for the selected resources. A dialog box containing the locked files and the lock description will be displayed. This is only active for resources under version control. For more details see Scanning for locks (on page 2749).
- **Lock (Ctrl + K (Command + K on macOS))** - Allows you to lock certain files that need exclusive access. You can write a comment describing the reason for the lock and you can also force (steal) the lock. This action is active only on files under version control. For more details on the use of this action see Locking a file (on page 2750).

- **Unlock (Ctrl + Alt + K (Command + Alt + K on macOS))** - Releases the exclusive access to a file from the repository. You can also choose to unlock it by force (break the lock).

**Show SVN Properties (Ctrl + P (Command + P on macOS))**

Opens the Properties view (on page 2841) and displays the SVN properties for a selected resource from Repositories view (on page 2808) or Working Copy view (on page 2813), depending on the view that was last used to invoke it.

**Show SVN Information (Ctrl + I (Command + I on macOS))**

Provides additional information for a selected resource. For more details, go to Obtain information for a resource (on page 2766).

**Exit (Ctrl + Q (Command + Q on macOS))**

Closes the application.

**Edit Menu**

- **Undo (Ctrl + Z (Command + Z on macOS))**
  
  Undo edit changes in the local file that is currently opened in the editor or the Compare view.

- **Redo (Ctrl + Y (Command + Y on macOS))**
  
  Redo edit changes in the local file that is currently opened in the editor or the Compare view.

- **Cut (Ctrl + X (Command + X on macOS))**
  
  Cut selection from the local file that is currently opened in the editor view or the Compare view to clipboard.

- **Copy (Ctrl + C (Command + C on macOS))**
  
  Copy selection from the local file that is currently opened in the editor or the Compare view to clipboard.

- **Paste (Ctrl + V (Command + V on macOS))**
  
  Paste selection from clipboard into the local file that is currently opened in editor or the Compare view.

- **Find/Replace (Ctrl + F (Command + F on macOS))**
Perform find and replace operations in the local file that is currently opened in the editor or the Compare view.

**Find Next (F3)**

Go to the next match using the same find options of the last find operation. This action runs in the editor panel and in any non-editable text area (for example, the Console view).

**Find Previous (Shift + F3)**

Go to the previous match using the same find options of the last find operation. This action runs in the editor panel and in any non-editable text area (for example, the Console view).

**Repository Menu**

**New Repository Location (Ctrl + Alt + N (Command + Alt + N on macOS))**

Displays the Add SVN Repository dialog box. This dialog box allows you to define a new repository location.

![Add SVN Repository Dialog Box](image)

If the Validate repository connection option is selected, the URL connection is validated before being added to the Repositories view.

**Edit Repository Location (Ctrl + Alt + E (Command + Alt + E on macOS))**

Context-dependent action that allows you to edit the selected repository location using the Edit SVN Repository dialog box. It is active only when a repository location root is selected.

**Change the Revision to Browse (Ctrl + Alt + B (Command + Alt + B on macOS))**

Context-dependent action that allows you to change the selected repository revision using the Change the Revision to Browse dialog box. It is active only when a repository location root is selected.

**Remove Repository Location (Ctrl + Alt + R (Command + Alt + R on macOS))**

Allows you to remove the selected repository location from the view. It shows you a confirmation dialog box before removal. It is active only when a repository location root is selected.

**Refresh (F5)**
Refreshes the resource selected in the **Repositories** view.

![Check out](image)

**Check out (Ctrl + Alt + O (Command + Alt + O on macOS))**

Allows you to create a working copy from a repository directory, on your local file system. To read more about this operation, see Check out a working copy (on page 2739).

**Export**

Opens the **Export** dialog box (on page 2803) that allows you to configure options for exporting a folder from the repository to the local file system.

**Import:**

**Import folder (Ctrl + Shift + L (Command + Shift + L on macOS))**

Allows you to import the contents of a specified folder from the file system into the selected folder in a repository. To read more about this operation, see the section Importing resources into a repository (on page 2802).

**Note:**

The difference between the **Import folder** and **Share project** actions is that the latter also converts the selected directory into a working copy.

**Import Files (Ctrl + Shift + I (Command + Shift + I on macOS))**

Imports the files selected from the file system into the selected folder in the repository.

**Working Copy Menu**

**Working Copies Manager (on macOS)**

Opens a dialog box with a list of working copies that the Apache Subversion™ client is aware of. In this dialog box you can add existing working copies or remove those that are no longer needed.

**Switch to**

Selects one of the following view modes: ✰ All Files, ✅ Modified, ✉ Incoming,➡ Outgoing, or ⚪ Conflicts.

**Refresh (F5)**

Refreshes the state of the selected resources or of the entire working copy (if there is no selection).

**Synchronize (Ctrl + Shift + S (Command + Shift + S on macOS))**
Connects to the repository and determines the working copy and repository changes made to the selected resources. The application switches to Modified view mode if the Always switch to 'Modified' mode option (on page 283) is selected.

**Update (Ctrl + U (Command + U on macOS))**

Updates all the selected resources that have incoming changes to the HEAD revision. If one of the selected resources is a directory then the update for that resource will be recursive.

**Update to revision/depth**

Allows you to update the selected resources from the working copy to an earlier revision from the repository. You can also select the update depth for the current folder. You can find out more about the depth term in the sparse checkouts (on page 2807) section.

**Commit**

Collects the outgoing changes from the selected resources in the working copy and allows you to choose exactly what resources to commit. A directory will always be committed recursively. Unversioned resources will be deselected by default. In the Commit dialog box you can also enter a comment before sending your changes to the repository.

**Update all (Ctrl + Shift + U (Command + Shift + U on macOS))**

Updates all resources from the working copy that have incoming changes. It performs a recursive update on the synchronized resources.

**Commit all**

Commits all the resources with outgoing changes. It is disabled when Incoming mode is selected or the synchronization result does not contain resources with outgoing changes. It performs a recursive commit on the synchronized resources.

**Revert (Ctrl + Shift + V (Command + Shift + V on macOS))**

Undoes all local changes for the selected resources. It does not contact the repository and the files are obtained from Apache Subversion™ pristine copy. It is available only for modified resources. See Revert your changes (on page 2757) for more information.

**Edit conflict (Ctrl + E (Command + E on macOS))**

Opens the Compare editor, allowing you to modify the content of the currently conflicting resources. For more information about editing conflicts, see Edit conflicts (on page 2755).

**Mark Resolved (Ctrl + Shift + R (Command + Shift + R on macOS))**
Instructs the Subversion system that you resolved a conflicting resource. For more information, see Merge conflicts (on page 2758).

**Mark as Merged (Ctrl + Shift + M (Command + Shift + M on macOS))**

Instructs the Subversion system that you resolved the pseudo-conflict by merging the changes and you want to commit the resource. Read the Merge conflicts (on page 2758) section for more information about how you can solve the pseudo-conflicts.

**Override and Update**

Drops any outgoing change and replaces the local resource with the HEAD revision. This action is available on resources with outgoing changes, including conflicting ones. See the Revert your changes (on page 2757) section.

**Override and Commit**

Drops any incoming changes and sends your local version of the resource to the repository. This action is available on conflicting resources. For more information see Drop incoming modifications (on page 2759).

**Mark as copied**

You can use this action to mark an item from the working copy as a copy of another item under version control, when the copy operation was performed outside of an SVN client. The Mark as copied action is available when you select two items (both the new item and source item), and it depends on the state of the source item.

**Mark as moved**

You can use this action to mark an item from the working copy as being moved from another location of the working copy, when the move operation was performed outside of an SVN client. The Mark as moved action is available when you select two items from different locations (both the new item and the source item that is usually reported as missing), and it depends on the state of the source item.

**Mark as renamed**

You can use this action to mark an item from the working copy as being renamed outside of an SVN client. The Mark as renamed action is available when you select two items from the same directory (both the new item and the source item that is usually reported as missing), and it depends on the state of the source item.

**Add to "svn:ignore" (Ctrl + Alt + I (Command + Alt + I on macOS))**

Allows you to add files that should not participate in the version control operations inside your working copy. This action can only be performed on resources not under version control. It actually modifies the value of the svn:ignore property in
the parent directory of the resource. Read more about this in the Ignore Resources Not Under Version Control (on page 2745) section.

**Add to version control (Ctrl + Alt + V (Command + Alt + V on macOS))**

Allows you to add resources that are not under version control. For further details, see Add Resources to Version Control (on page 2743) section.

**Remove from version control**

Schedules the selected items for deletion from repository upon the next commit. The items are not removed from the file system after committing.

**Clean up (Ctrl + Shift + C (Command + Shift + C on macOS))**

Performs a maintenance cleanup operation on the selected resources from the working copy. This operation removes the Subversion maintenance locks that were left behind. This is useful when you already know where the problem originated and want to fix it as quickly as possible. It is only active for resources under version control.

**Expand All (Ctrl + Alt + X (Command + Alt + X on macOS))**

Displays all descendants of the selected folder. The same behavior is obtained by double-clicking a collapsed folder.

**Collapse all (Ctrl + Alt + Z (Command + Alt + Z on macOS))**

Collapses all descendants of the selected folder. The same behavior is obtained by double-clicking an expanded folder.

**Compare Menu**

**Perform Files Differencing**

Looks for differences between the two files displayed in the left and right side panels.

**Next Block of Changes (Ctrl + Period (Command + Period on macOS))**

Jumps to the next block of changes. This action is not available when the cursor is positioned on the last change block or when there are no changes.

**Previous Block of Changes (Ctrl + Comma (Command + Comma on macOS))**

Jumps to the previous block of changes. This action is not available when the cursor is positioned on the first change block or when there are no changes.

**Next Change (Ctrl + Shift + Period (Command + Shift + Period on macOS))**

- **Note:**
  A change block groups one or more consecutive lines that contain at least one change.
Jumps to the next change from the current block of changes. When the last change from the current block of changes is reached, it highlights the next block of changes. This action is not available when the cursor is positioned on the last change or when there are no changes.

 Ramirez Change (Ctrl + Shift + Comma (Command + Shift + M on macOS))

 Jumps to the previous change from the current block of changes. When the first change from the current block of changes is reached, it highlights the previous block of changes. This action is not available when the cursor is positioned on the first change or when there are no changes.

 Last Change (Ctrl + E (Command + E on macOS))

 Jumps to the last change.

 First Change (Ctrl + B (Command + B on macOS))

 Jumps to the first change.

 Copy All Changes from Right to Left

 Copies all changes from the file in the right panel to the file in the left panel.

 Copy Change from Right to Left

 Copies the selected difference from the file in the right panel to the file in the left panel.

 Show Word Level Details

 Provides a word-level comparison of the selected change.

 Show Character Level Details

 Provides a character-level comparison of the selected change.

 Format and Indent Both Files (Ctrl + Shift + P (Command + Shift + P on macOS))

 Formats and indents both files before comparing them. Use this option for comparisons that contain long lines that make it difficult to spot differences.

 Note:

 When comparing two JSON files, the Format and Indent Both Files action will automatically sort the keys in both files the same to make it easier to compare.

 Ignore Whitespaces

 Enables or disables the whitespace ignoring feature. Ignoring whitespace means that before performing the comparison, the application normalizes the content and trims its leading and trailing whitespaces.

 History Menu
Show History (Ctrl + H (Command + H on macOS))

Displays the history for an SVN resource at a given revision. The resource can be one selected from the Repositories view, Working Copy view, or from the Affected Paths table from the History view, depending on which view was last focused when this action was invoked.

Show Annotation (Ctrl + Shift + A (Command + Shift + A on macOS))

Opens the Show Annotation dialog box that computes the annotations for a file and displays them in the Annotations view (on page 2835), along with the history of the file in the History view.

Repositories

This operation is available for any resource selected from view, Working Copy view, History view or Directory Change Sets view, depending on which view was last focused when this action was invoked.

Revision Graph (Ctrl + G (Command + G on macOS))

This action allows you to see the graphical representation of a resource's history. For more details about a resource's revision graph see the section Revision Graph (on page 2843). This operation is available for any resource selected in the Repositories view or Working Copy view.

Tools Menu

Share project

Allows you to share a new project (on page 2737) using an SVN repository. The local project is automatically converted into an SVN working copy.

Branch / Tag

Allows you to copy the selected resource from the Repositories view or Working Copy view to a branch or tag into the repository. To read more about this operation, see the section Creating a Branch / Tag (on page 2768).

Merge (Ctrl + J (Command + J on macOS))

Allows you to merge the changes made on one branch back into the trunk, or vice versa, using the selected resource from the working copy. To read more about this operation, see the section Merging (on page 2770).

Switch (Ctrl + Alt + W (Command + Alt + W on macOS))

Allows you to change the repository location of a working copy, or only of a versioned item of the working copy, within the same repository. It is available when the selected item of the working copy is a versioned resource, except for external items. To read more about this action, see the Switching the Repository Location (on page 2787) section.

Relocate
Allows you to change the base URL of the root folder of the working copy to a new URL when the base URL of the repository changed. For example, if the repository itself was moved to a different server. This operation is only available for the root item of the working copy. To read more about this operation, see the Relocate a Working Copy (on page 2789) section.

**Create patch (Ctrl + Alt + P (Command + Alt + P on macOS))**

Allows you to create a file containing all the differences between two resources, based on the `svn diff` command. To read more about creating patches, see the section about patches (on page 2792).

### Working copy format

This submenu contains the following two operations:

- **Upgrade**
  
  Upgrades the format of the currently loaded working copy to the newest one known by Syncro SVN Client. This allows you to benefit of all the new features of the client.

- **Downgrade**
  
  Downgrades the format of the currently loaded working copy to SVN 1.7 format. This is useful if you want to use older SVN clients with the current working copy, or, by mistake, you have upgraded the format of an older working copy to SVN 1.8.

  **Note:**
  
  SVN 1.7 working copies cannot be downgraded to older formats.

See the section Working Copy Format (on page 2820) to read more about this subject.

### Options Menu

- **Preferences**
  
  Opens the Preferences dialog box.

- **Menu Shortcut Keys**
  
  Opens the Menu Shortcut Keys preferences page (on page 294), where users can configure in one place the keyboard shortcuts available for menu items available in Syncro SVN Client.

- **Global Run-Time Configuration**
  
  Allows you to configure SVN general options, that should be used by all the SVN clients you may use:
• **Edit 'config' file** - In this file you can configure various SVN client-side behaviors.

• **Edit 'servers' file** - In this file you can configure various server-specific protocol parameters, including HTTP proxy information and HTTP timeout settings.

**Export Options**

Allows you to export the current options to an XML file.

**Import Options**

Allows you to import options you have previously exported.

**Reset Options**

Resets all your options to the default ones.

**Reset Authentication**

Resets the Subversion authentication information.

**Window Menu**

**Show View**

Allows you to select the view you want to bring to front.

**Show Toolbar**

Allows you to select the toolbar you want to be visible.

**Enable flexible layout**

Toggles between a fixed and a flexible layout. When the flexible layout is enabled, you can move and dock the internal views to adapt the application to various viewing conditions and personal requirements.

**Reset Layout**

Resets all the views to their default position.

**Help Menu**

**Help (F1)**

Opens the Help dialog box.

**Use online help (selected by default)**

If this option is selected, when you select Help or press F1 while hovering over any part of the interface, Oxygen XML Editor attempts to open the help documentation in online mode. If this option is not selected or an internet connection fails, the help documentation is opened in offline mode.

**Show Dynamic Help view**

Displays the Dynamic Help view.
Report Problem

Opens a dialog box that allows you to write the description of a problem that was encountered while using the application.

Support Center

Opens the Support Center web page in a browser.

About

Opens the About dialog box.

SVN Main Toolbar

The toolbar of the Syncro SVN Client SVN Repositories window contains the following actions:

- **Check out**
  Checks out a working copy from a repository. The repository URL and the working copy format must be specified.

- **Synchronize**
  Synchronizes the current working copy with the repository.

- **Update All**
  Updates all resources of the working copy that have an older revision that repository.

- **Commit All**
  Commits all resources of working copy that have a newer version compared to that of the repository.

- **Refresh**
  Refreshes the whole content of the current working copy from disk starting from the root folder. At the end of the operation, the modified files and folders that were not committed to repository yet, are displayed in the Working Copy view.

- **Compare**
  The selected resource is compared with:

  - The BASE revision, when the selected resource is:
    - Locally modified and the All Files view mode is currently selected (no matter if there are incoming changes).
    - Locally modified and there are no incoming changes when any other view mode is selected.
• The remote version of the same resource, when remote information is available after a 
  Synchronize operation (only when one of Modified, Incoming, Outgoing and Conflicts 
  view modes is selected).
• The working copy revision, when the selected resource is from the History view.

Show History
Displays the history of the selected resource (from the Working Copy or Repository views) in the 
History view.

Show Annotation
Displays the annotations of the selected resource. The selected resource can be in the Working 
Copy or the History views.

Revision Graph
Displays the revision graph of the selected resource. The selected resource can be in the 
Working Copy or the Repositories views.

Enable/Disable flexible layout
Toggles between a fixed and a flexible layout. When the flexible layout is enabled, you can move 
and dock the internal views to adapt the application to various viewing conditions and personal 
requirements.

Status Bar
The status bar of the Syncro SVN Client window displays important details of the current status of the 
application. This information is available only in the Working Copy view.

Figure 680. Status bar

The status bar is composed of the following areas:

• The path of the currently processed file from the current working copy (during an operation such as 
  Check out or Synchronize) or the result of the last operation.
• The current status of the following working copy options:
  ◦ Show ignored files ( ).
  ◦ Show deleted files ( ).
  ◦ Process svn:externals definitions ( ).
  The options for ignored and deleted files are switched on and off from the Settings menu (on page 
  2819) of the Working Copy panel:
• The format of the currently loaded working copy.
• The current numbers of incoming changes (➡️), outgoing changes (➡️) and conflicting changes (➡️).
• A progress bar for the currently running SVN operation and a button (⏹) that allows you to stop it.

Getting Started

This section explains the basic operations that can be done in Syncro SVN Client.

SVN Repository Location

This section explains how to add and edit the repository locations in Syncro SVN Client.

Add / Edit / Remove Repository Locations

Usually, team members do all of their work separately, in their own working copy, and then must share their work by committing their changes. This is done using an Apache Subversion™ repository. Oxygen XML Editor supports versions 1.4, 1.5, 1.6, 1.7, and 1.8 of the SVN repository format.

Before you can begin working with a Subversion repository, you must define a repository location in the Repositories view (on page 2808).

To create a repository location, use the New Repository Location action that is available in the Repository menu, the Repositories view toolbar, and in the contextual menu. This action opens the New Repository Location dialog box, which prompts you for the URL of the repository (on page 2848) you want to connect to. You can also use peg revisions at the end of the URLs (on page 2850) (for example, URL@rev1234) to browse only that specific revision. No authentication information is requested at the time the location is defined. It is left to the Subversion client to request the user and password information when it is needed. The main benefit of allowing Subversion to manage your password is that it prompts you for a new password only when your password changes.

Once you enter the repository URL, Oxygen XML Editor tries to contact the server to get the content of the repository for displaying it in the Repositories view (on page 2808). If the server does not respond in the timeout interval set in the preferences, an error is displayed. If you do not want to wait until the timeout expires, you can use the Stop button from the toolbar of the view.

To edit a repository location, use the Edit Repository Location action that is available in the Repository menu and in the contextual menu. This action opens the Edit Repository Location dialog box, which prompts you for the URL of the repository (on page 2848) you want to connect to. You can also use peg revisions at the end of the URLs (on page 2850) (for example, URL@rev1234) to browse only that specific revision.

To remove a repository location, use the Remove Repository Location action that is available in the Repository menu and in the contextual menu. A confirmation dialog box is displayed to make sure that you do not accidentally remove the wrong locations.
The order of the repositories can be changed in the **Repositories** view at any time with the **Up** arrow and **Down** arrow buttons on the toolbar of the view. For example, pressing the up arrow once moves the selected repository in the list up one position.

To set the reference revision number of an SVN repository use the **Change the Revision to Browse** action that is available in the **Repository** menu and in the contextual menu. The revision number of the repository is used for displaying the contents of the repository when it is viewed in the **Repositories view** (on page 2808). Only the files and folders that were present in the repository at the moment when this revision number was generated in the repository are displayed as contents of the repository tree. Also, this revision number is used for all the operations executed directly from the **Repositories view** (on page 2808).

### Authentication

Five protocols are supported: **HTTP**, **HTTPS**, **SVN**, **SVN + SSH** and **FILE**. If the repository that you are trying to access is password protected, the **Enter authentication data** dialog box requests a user name and a password. If the **Store authentication data** checkbox is selected, the credentials are stored in the Apache Subversion™ default directory:

- **Windows** - `%HOME%\Application Data\Subversion\auth`. Example: C:\Documents and Settings\John\Application Data\Subversion\auth
- **Linux and macOS** - `$HOME/.subversion/auth`. Example: /home/John/.subversion/auth

There is one file for each server that you access. If you want to make Subversion forget your credentials, you can use the **Reset authentication** command from the **Options** menu. This causes Subversion to forget all your credentials. When you reset the authentication data, restart Oxygen XML Editor for the change to take effect.

**Tip:**

The **FILE** protocol is recommended if the SVN repository and Oxygen XML Editor are located on the same computer as it ensures faster access to the SVN repository compared with other protocols.

For HTTPS connections where client authentication is required by your SSL server, you must choose the certificate file and enter the corresponding certificate password that is used to protect your certificate.

When using a secure HTTP (HTTPS) protocol for accessing a repository, a **Certificate Information** dialog box is displayed and asks you whether you want to accept the certificate permanently, temporarily, or simply deny it.

If the repository has SVN+SSH protocol, the SSH authentication can also be made with a private key and a pass phrase.
After the SSH authentication dialog box, another dialog box appears for entering the SVN user name that accesses the SVN repository. The SVN user name is recorded as the *committer* in SVN operations.

When connecting for the first time to a Subversion repository through SVN+SSH protocol, you will be asked to confirm if you trust the SSH host. The same dialog box is also displayed when the server changed the SSH key or when the key was deleted from the local Subversion cache folder.
Share a Project

Even if you start developing a new project, or you want to migrate an existing one to Subversion, the Syncro SVN Client allows you to easily share it with the rest of your team. The shared project directory is automatically converted to a working copy and added under Syncro SVN Client management. The Share project action is available in the Tools menu and the contextual menu of the Repositories view.

Figure 683. Share Project Dialog Box

The following options can be configured in the Share project dialog box:

**Project**

The location of the project folder (on page 2848) on the local disk by using the text box or the Browse button. This folder should not be empty or already under version control.

⚠️ Important:

By default, the SVN system only imports the content of the specified folder, and not the root folder itself. Therefore, it is recommended to use the Browse button to select the project folder so that the client will automatically append the name of it to the specified URL.

**URL**

The new location of the project (on page 2848) (inside the repository) that will be used to access it.
Note:

Peg revisions have no effect for this operation since it is used to send information to the repository.

Attention:

If the new location already exists, make sure that it is an empty directory to avoid mixing your project content with other files (if items exist with the same name, an error will occur and the operation will not proceed). Otherwise, if the address does not exist, it is created automatically.

Format

The SVN format of the working copy. You can choose between SVN 1.8 or SVN 1.7.

Share files matching global ignore patterns

When selected, the file names that match the patterns defined in either of the following locations are also imported into the repository:

• The global-ignores property in the SVN configuration file (on page 2848).
• The File name ignore patterns option (on page 284) in the SVN > Working Copy preferences page (on page 283).

Enable automatic properties/Disable automatic properties

Enables or disables automatic property assignment (per runtime configuration rules), overriding the enable-auto-props runtime configuration directive, defined in the SVN configuration file (on page 2848).

Note:

This option is available only when there are defined properties to be applied automatically for newly added items under version control. You can define these properties in the SVN config file (in the auto-props section). Based on the value of the enable-auto-props runtime configuration directive, the presented option is either Enable automatic properties, or Disable automatic properties.

Defining a Working Copy

An Apache Subversion™ working copy is an ordinary directory tree on your local system, containing a collection of files. You can edit these files however you want, your working copy being your private work area. To make your own changes available to others or incorporate changes made by others, you must explicitly tell Subversion to do so. You can even have multiple working copies of the same project.
A Subversion working copy also contains some extra files, created and maintained by Subversion, to help it keep track of your files. In particular, each directory in your working copy contains a subdirectory named `.svn`, also known as the working copy administrative directory. This administrative directory contains an unaltered copy of the last updated files from the repository. This copy is usually referred to as the pristine copy or the BASE revision of the working copy. These files help Subversion recognize which files contain unpublished changes, and which files are out-of-date with respect to others’ work.

A typical Subversion repository often holds the files (or source code) for several projects. Usually each project is a subdirectory in the repository’s file system tree. In this arrangement, a user’s working copy usually corresponds to a particular sub-tree of the repository.

### Check Out a Working Copy

*Check out* means to make a copy of a project from a repository to your local file system. This copy is called a working copy. An Apache Subversion™ working copy is a specially formatted directory structure that contains additional `.svn` directories that store Subversion information, as well as a pristine copy of each item that is checked out.

To check out a working copy, locate and select the desired directory in the Repositories view and select the *Check out* action from the contextual menu, the toolbar, or the Repository menu.
The following options can be configured in the **Check out** dialog box:

**URL**

The location of the repository directory (on page 2848) to be checked out.

*Note:* To check out an item that was deleted, moved, or replaced, you need to specify the original URL (before the item was removed) and use a peg revision (on page 2850) at the end (for example, URL@rev1234).

**Revision**

You can choose between the **HEAD** or **Other** revision. If you need to check out a specific revision, specify it in the **Other** text box or use the **History** button and choose a revision from the **History** dialog box (on page 2741).

**Check out to**

Specify the location where you want to check out (on page 2848) the new working copy by typing the local path in the text box or by using the **Browse** button. If the specified local path does not point to an existing directory, it will automatically be created.

*Important:* By default, the SVN system only checks out the content of the directory specified by the URL, and not the directory itself. Therefore, it is recommended to use the **Browse** button to select the check out location so that the client will automatically append the name of the remote directory to the path of the selected directory.
Warning:
The destination directory should be empty. If files exist, they are skipped (left unchanged) by the check out operation and displayed as modified (on page 2815) after the operation has finished. Also, the destination directory must not already be under version control.

Format
The SVN format of the working copy. You can choose between SVN 1.8 or SVN 1.7.

Depth
The depth is useful if you want to check out only a part of the selected repository directory and bring the rest of the files and subdirectories in a future update. You can find out more about the checkout depth in the sparse checkouts (on page 2807) section. You can choose between the following depths:

- Recursive (infinity) - Checks out all the files and folders contained in the selected folder.
- Immediate children (immediates) - Checks out only the child files and folders without recursing subfolders.
- File children only (files) - Checks out only the child files.
- This folder only (empty) - Checks out only the selected folder (no child file or folder is included).

Ignore "svn:externals" definitions
When selected, external items are ignored in the check out operation. This option is only available if you choose the Recursive (infinity) depth.

After a check out, the new working copy is added to the list in the Working Copy view (on page 2813) and loaded automatically.

History Dialog Box
The History dialog box presents a list of revisions for a resource. It is opened from the dialog boxes that require setting an SVN revision number, such as the Check Out dialog box (on page 2739) or the Branch / Tag dialog box (on page 2768). It presents information about revision, commit date, author, and commit comment.
The initial number of entries in the list is 50. Additional revisions can be added to the list using the **Get next 50** and **Get all** buttons. The list of revisions can be refreshed at any time with the **Refresh** button. You can group revisions in predefined time frames (today, yesterday, this week, this month), by pressing the **Group by date** button from the toolbar.

The **Affected Paths** area displays all paths affected by the commit of the revision selected in history. You can see the changes between the selected revision and the file's previous state using the **Compare with previous version** action, available in the contextual menu.

### Use an Existing Working Copy

Using an existing working copy is the process of taking a working copy that exists on your file system and connecting it to the Apache Subversion™ repository. If you have a brand new project that you want to import into your repository, then see the section **Import resources into the repository** *(on page 2802)*. The following procedure assumes that you have an existing valid working copy on your file system.

1. Click the **Working Copies Manager** toolbar button (.folder icon) in the **Working Copy** view *(on page 2813)*.

   **Step Result:** This action opens the **Working copies list** dialog box.

2. Click the **Add** button.
3. Select the working folder copy from the file system. The name is useful to differentiate between working copies located in folders with the same name. The default name is the name of the root folder of the working copy.

**Note:**
For SVN 1.7 and newer working copies, all the internal information is kept only in the root directory. Thus, Syncro SVN Client needs to load the whole working copy.

4. Click the OK button.

The selected working copy is loaded and presented in the **Working Copy view** (on page 2813).

**Notice:**
You can add working copies older than SVN 1.7. However, to load any of them, Syncro SVN Client will require to upgrade the working copy to SVN 1.8 format.

**Manage Working Copy Resources**

This section explains how to work with the resources that are displayed in the **Working Copy** view.

**Edit Files**

You can edit files from the **Working Copy view** (on page 2813) by double clicking them or by right clicking them and choosing **Open** from the contextual menu.

Note that only one file can be edited at a time. If you try to open another file, it is opened in the same editor window. The editor has syntax highlighting for known file types, meaning that a different color is used for each type of recognized token in the file. If the selected file is an image, then it is previewed in the editor, with no access to modifying it.

After modifying and saving a file from a working copy, a modified marker - an asterisk (*) - will be added to the file's icon in the **Working Copy view** (on page 2813). The asterisk marks the files that have local modifications that were not committed to the repository.

**Add Resources to Version Control**

To share new files and folders (created in your working copy), add them to version control using the **Add to version control** option from the **Working Copy view** (on page 2813).

You can easily spot resources not under version control by the **unversioned** (فاقس) icon displayed in the **Local file status** column. Resources scheduled for addition are displayed with this **added** (قيد الربط) icon in the **Working Copy view** and are added in the repository after you commit them.
**Note:**

Do not make a confusion between 🍀 and 🍁 icons. The former icon stands for resources that are actually copies of resources already committed in the repository, meaning they are *scheduled for addition with history*.

When you use the **Add to version control** option on a directory, its entire structure is scanned and all the resources that can be added under version control are presented.

Although it is not mandatory to add resources under version control explicitly, it is recommended. If you forgot to add a resource, when you commit your changes (on page 2761), the resource is presented in the commit dialog box, but not selected. When you commit and *unversioned* resource, it is automatically added under version control before starting the commit operation.

![Add to Version Control Dialog Box](image)

**Figure 687. Add to Version Control Dialog Box**

**Note:**

*Ignored* (⚫) items can also be added under version control.

The **Depth** column is displayed only when directories are also presented in the dialog box. For any directory, you can use one of the available values to instruct Subversion to limit the scope of the operation to a particular tree depth.

**Note:**

The initial value of the **Depth** field can have the following values, depending on the listing mode of the items in the working copy view (on page 2819):
When you add unversioned or ignored directories, the initial value of the Depth field also depends on the state of the Show unversioned directories content and Show ignored directories content options. If these options are selected, the value is based on the listing mode of the items in the working copy view. When they are not selected, the value is empty.

The following options are available in this dialog box:

- **Enable automatic properties** or **Disable automatic properties** - enables or disables automatic property assignment (per runtime configuration rules), overriding the enable-auto-props runtime configuration directive, defined in the config file of the Subversion configuration directory.

  **Note:**
  This option is available only when there are defined properties to be applied automatically for resources newly added under version control. You can define these properties in the config file of the Subversion configuration directory in the auto-props section. Based on the value of the enable-auto-props runtime configuration directive, the presented option is either Enable automatic properties, or Disable automatic properties.

- **No ignore** - when you select this option, file-name patterns defined to ignore unversioned resources do not apply. Resources that are located inside an unversioned directory selected for addition, and match these patterns, are also scheduled for addition in the repository.

  **Note:**
  This option is available only when directories are also presented in the dialog box.

  You can define file-name patterns to ignore unversioned resources in one of the following locations:
  - In the config file of the Subversion configuration directory (the global-ignores option from the miscellany section).
  - In the Oxygen XML Editor options (open the Preferences dialog box (Options > Preferences) (on page 124) and go to SVN > Working copy > Application global ignores).

Each of the above two options is activated only when you select an item that can have the option applied.

**Ignore Resources Not Under Version Control**

Some resources inside your working copy do not need to be subject to version control. These resources can be files created by the compiler, *.obj, *.class, *.lst, or output folders used to store temporary
files. Whenever you commit changes (on page 2761), Apache Subversion™ shows your modified files in the commit dialog box, but the unversioned files are also listed. Since the unversioned files are committed unless otherwise specified, it is difficult to see exactly what you are committing.

The best way to avoid these problems is to add the derived files to the Subversion ignore list. That way they are never displayed in the commit dialog box and only genuine unversioned files that must be committed are displayed.

You can choose to ignore a resource by using the Add to svn:ignore action in the contextual menu of the Working Copy view (on page 2813).

In the Add to svn:ignore dialog box, you can specify the resource to be ignored by name or by a custom pattern. The custom pattern can contain the following wildcard characters:

- * - Matches any string of characters of any size, including the empty string.
- ? - Matches any single character.

For example, you can choose to ignore all text documents by using the pattern: *.txt.

The action Add to svn:ignore adds a predefined Subversion property called svn:ignore to the parent directory of the specified resource. In this property, there are specified all the child resources of that directory that must be ignored. The result is visible in the Working Copy view. The ignored resources are represented with gray icons.

Delete Resources

The Delete action is available in the contextual menu of the Working Copy view (on page 2813). When you delete an item from the working copy, it is marked as deleted (scheduled for deletion from repository upon the next commit) and removed from the file system. Depending on the state of each item, you are prompted to confirm the operation.

If a resource is deleted from the file system without Subversion’s knowledge, the resource is marked as missing (2) in your working copy. You can decide what you want to do with a missing item:

- In the case of a commit, any missing item is first automatically deleted and then committed.

  **Note:**

  Not any missing item can be committed as deleted, and removed from the repository. For example, you cannot commit an item that no longer exists on the disk and that was scheduled for addition (4) previously, since this item does not exist in the repository, but you can use the Delete action instead.

- If you want to recover missing items, either update (on page 2760) the items themselves or one of their parent directories. This fetches their latest version from the repository.
You can also delete conflicting items (file content conflicts, property conflicts, tree-conflicts) and Syncro SVN Client automatically marks them as resolved.

**Note:**
It is recommended that you resolve conflicts manually to avoid losing any important remote modifications.

Finally, you can change your mind and revert (on page 2757) the deleted items to their initial, pristine, state.

## Copy Resources

You can copy resources from various locations of the working copy. You select them in the Working Copy view (on page 2813) and then use Copy to from the contextual menu. This is not a simple file system copy, but an Apache Subversion™ command. It will copy the resource and the copy will also have the original history. This is one of the important features of Subversion, as you can keep track of where the copied resources originated.

Based on the selected items, the Copy to action is available only if it can be performed. Even if the operation would not normally be possible in SVN (due to some invalid local file states against copy), Oxygen XML Editor performs the copy operation as a simple file system operation. This means no SVN versioning meta-data is affected.

**Note:**
- If you copy an item to a directory that is not under version control (on page 2815) (unversioned or ignored), the history of the item is not preserved. For example, when copying directories, all items inside them will also be copied without history.
- If you copy a directory that contains external (on page 2815) items, these are not copied. This is specific for SVN 1.7 working copies only. To fetch the external items, use the Update operation on the copied directory.

In the Copy to dialog box, you can navigate through the working copy directories to choose a target directory, to copy inside it. If you try to copy a single resource you are also able to change that resource's name. For versioned items, you can select Ignore resource history to copy them without their history (similar to a simple file system copy).

**Note:**
The Copy to dialog box only presents all the local directories that are a valid destination against the copy operation, based on their local file status. Also, the working copy settings (on page 2819) are taken into account.

In the Commit dialog box, only the directory in question will appear without its children.
Move Resources

As in the case of the copy command, you can move several resources at once. Select the resources in the Working Copy view (on page 2813) and choose the Move to action from the contextual menu. The move command actually behaves as if a copy followed by a delete command were issued. You will find the moved resources at the desired destination and also at their original location, but marked as deleted.

Note:

External items cannot be moved using the Move to action, because they cannot be deleted. Instead, you should edit the svn:externals property defining the external item and use the Update operation on the item's parent folder for the change to take effect.

Attention:

For SVN 1.8 working copies: when committing items that were moved and/or renamed, make sure you select both the source and the destination. Otherwise, the commit operation will fail.

Rename Resources

The Rename action is available in the contextual menu of the Working Copy view (on page 2813) and can be performed on a single resource. This action acts as a move command with the destination directory being the same as the original location of the resource. A copy of the original item is created with the new name, also keeping its history. The original item is marked as deleted.

Note:

External items cannot be renamed using the Rename action because they cannot be deleted. Instead, you should edit the svn:externals property defining the external item, then use the Update operation on the item's parent folder for the change to take effect.

Attention:

For SVN 1.8 working copies: when committing items that were moved and/or renamed, make sure you select both the source and the destination. Otherwise, the commit operation will fail.

Lock / Unlock Resources

The idea of version control is based on the copy-modify-merge model of file sharing. This model states that each user contacts the repository and creates a local working copy (checkout). Users can then work independently and modify their working copies according to their needs. When their goal has been accomplished, it is time for the users to share their work with the others, to send them to the repository (commit). When a user has modified a file that has been also modified on the repository, the two files will have to be merged. The version control system assists the user with the merging as much as it can, but in the end the user is the one that must make sure it is done correctly.
The copy-modify-merge model only works when files are contextually mergeable: this is usually the case of line-based text files (such as source code). However this is not always possible with binary formats, such as images or sounds. In these situations, the users must each have exclusive access to the file, ending up with a *lock-modify-unlock* model. Without this, one or more users could end up wasting time on changes that cannot be merged.

An SVN lock is a piece of metadata that grants exclusive access to a user. This user is called the lock owner. A lock is uniquely identified by a lock token (a string of characters). If someone else attempts to commit the file (or delete a parent of the file), the repository demands two pieces of information:

- User authentication - the user performing the commit must be the lock owner
- Software authorization - the user's working copy must have the same lock token as the one from the repository, proving that it is the same working copy where the lock originated from.

**Scanning for Locks**

When starting to work on a file that is not contextually mergeable (usually a binary file), it is better to verify if someone else is not already working on that file. You can do this in the Working Copy view (on page 2813) by selecting one or more resources, then right-clicking them and choosing the Scan for Locks action from the contextual menu.

![Locked Items Dialog Box](image)

The **Locked items** dialog box contains a table with all the resources that were found locked on the repository. For each resource there are specified: resource path, state of the lock, owner of the lock, lock comment, creation and expiration date for the lock (if any).

The state of the lock can be one of the following:

- 🗝️ - Appears when one of the following conditions apply:
  - Another user has locked the file in the repository.
  - The file was locked by the same user from another working copy.
  - The file was locked from the Repositories view.
- 🗝️ - Displayed after you have locked a file from the current working copy.
- A file already locked from your working copy is no longer locked in the repository (it was unlocked by another user).
- A file already locked from your working copy is being locked by another user. Now the owner of the file lock is the user who stole the lock from you.

You can unlock a resource by selecting it and pressing the **Unlock** button.

**Related Information:**

- Working Copy Locks *(on page 2817)*
- Repository Locks *(on page 2808)*

**Locking a File**

By locking a file, you have exclusive write access to it in the repository.

You can lock a file from your working copy or directly from the **Repositories** view.

**Note:**

You can only lock files (not directories). This is a restriction imposed by Apache Subversion™.

The **Lock** dialog box allows you to write a comment when you set a lock or when you steal an existing one. Note that you should steal a lock only after you made sure that the previous owner no longer needs it. Otherwise, you may cause an unsolvable conflict, which could be the reason the lock was put there in the first place. The Subversion server can have a policy concerning lock stealing, as it may not allow you to do this if certain conditions are not met.

The lock stays in place until you unlock the file or until someone breaks it. There is also the possibility that the lock expires after a period of time specified in the Subversion server policy.

**Unlocking a File**

A file can be unlocked from the contextual menu of the **Working Copy view** *(on page 2813)*. A dialog box will prompt you to confirm the unlocking and it will also allow you to break the lock (unlock it by force).

**Synchronize with Repository**

In the work cycle you will need to incorporate other people's changes (update) and to make your own work available to others (commit). This is what the **Incoming** and **Outgoing** modes of the **Working Copy view** *(on page 2813)* was designed for, to help you send and receive modifications from the repository.

The **Incoming** and **Outgoing** modes of this view focus on incoming and outgoing changes. The incoming changes are the changes that other users have committed in the repository since you last updated your working copy. The outgoing changes are the modifications you made to your working copy as a result of editing, removing or adding resources.
The view presents the status of the working copy resources against the BASE revision after a Refresh operation. You can view the state of the resources versus a repository HEAD revision by using the Synchronize action from the Working Copy view (on page 2813).

**View Differences**

One of the most common requirements in project development is to see what changes have been made to the files from your Working Copy or to the files from the repository. You can examine these changes after a synchronize operation with the repository, by using the Open in compare editor action from the contextual menu.

The text files are compared using a built-in Compare view (on page 2837) that uses a line differencing algorithm or a specified external diff application (if such an application is set in the SVN Diff preferences page (on page 284)). When a file with outgoing status is involved, the compare is performed between the file from the working copy and the BASE revision of the file. When a file with incoming or conflict status is involved, the differences are computed using a three-way algorithm that means that the local file and the repository file are each compared with the BASE revision of the file. The results are displayed in the same view. The differences obtained from the local file comparison are considered outgoing changes and the ones obtained from the repository file comparison are considered incoming changes. If any of the incoming changes overlap outgoing changes then they are in conflict.

A special case of difference is a diff pseudo-conflict. This is the case when the left and the right sections are identical but the BASE revision does not contain the changes in that section. By default, this type of changes are ignored. If you want to change this, you can go to the SVN preferences page and select the Allow unversioned obstructions option (on page 282).

The right editor of the internal compare view presents either the BASE revision or a revision from the repository of the file so its content cannot be modified. By default, when opening a synchronized file in the Compare view, a compare is automatically performed. After modifying and saving the content of the local file presented in the left editor, another compare is performed. You will also see the new refreshed status in the Working Copy view (on page 2813).
At the top of each of the two editors, there are presented the name of the open file, the corresponding SVN revision number (for remote resources) and the author who committed the associated revision.

There are three types of differences:

- **Incoming changes** - Changes committed by other users and not present yet in your working copy file. They are marked with a blue highlight and on the middle divider the arrows point from right to left.
- **Outgoing changes** - Changes you have done in the content of the working copy file. They are marked with a gray highlight and the arrows on the divider are pointing from left to right.
- **Conflicting changes** - This is the case when the same section of text that you already modified in the local file has been modified and committed by some other person. They are marked with a red highlight and red diamonds on the divider.

There are numerous actions and options available in the Compare View toolbar (on page 2838) or in the Compare menu from the main menu. You can decide that some changes need adjusting or that new ones must be made. After you perform the adjustments, you may want to perform a new compare between the files. For this case there is an action called **Perform files differencing**. After each files differencing operation the first found change will be selected. You can navigate from one change to another by using the actions **Go to first**, **Go to previous**, **Go to next** and **Go to last modification**. If you decide that some incoming change needs to be present in your working file you can use the action **Copy change from right to left**. This is useful also when you want to override the outgoing modifications contained in a conflicting section. The **Copy all**
non-conflicting changes from right to left action copies all incoming changes that are not contained inside a conflicting section in your local file.

Suppose that only a few words or letters are changed. Considering that the differences are performed taking whole lines of text into account, the change will contain all the lines involved. To find exactly what words or letters have changed, the Word Details and Character Details dialog boxes are available. They present a more detailed comparison result when you double-click the middle divider of a difference.

When you want to examine only the changes in the real text content of the files, while disregarding the changes in the number of white spaces between words or lines, there is an option available in the SVN Preferences (on page 281) that allows you to enable or disable the white space ignoring feature of the compare algorithm.

Conflicts

A file conflict occurs when two or more developers have changed the same few lines of a file or the properties of the same file. As Subversion knows nothing of your project, it leaves resolving the conflicts to the developers. Whenever a conflict is reported, you should open the file in question, and try to analyze and resolve the conflicting situation.

Real Conflicts vs Mergeable Conflicts

There are two types of conflicts:

• real conflict ( icon in Name column) - Syncro SVN Client considers the following resource states to be real conflicts:
  ◦ conflicted state - A file reported by SVN as being in this state is obtained after it was updated/merged while having incoming and outgoing content or property changes at the same time, changes that could not be merged. A content conflict ( icon in Local file status column) is reported when the modified file has binary content or it is a text file and both local and remote changes were found on the same line. A properties conflict ( icon in Local properties status column) is reported when a property’s value was modified both locally and remotely.
  ◦ tree conflicted state ( icon in Local file status column) - Obtained after an update or merge operation, while having changes at the directory structure level (for example, file is locally modified and remotely deleted or locally scheduled for deletion and remotely modified).
  ◦ obstructed state ( icon in Local file status column) - Obtained after a resource was versioned as one kind of object (file, directory, symbolic link), but has been replaced outside Syncro SVN Client by a different kind of object.

• pseudo-conflict ( icon in Name column) - A file is considered to be in pseudo-conflict when it contains both incoming and outgoing changes. When incoming and outgoing changes do not intersect, an update operation may automatically merge the incoming file content into the existing locally one. In this case, the pseudo-conflict marker is removed. This marker is used only as a warning that should prevent you to run into a real conflict.
Note:

- A conflicting resource cannot be committed to repository. You have to resolve it first, by using **Mark Resolved** action (after manually editing/merging file contents) or by using **Mark as Merged** action (for pseudo-conflicts).
- and icons are presented only when one of the following view modes is selected: Modified, Incoming, Outgoing, Conflicts.
- The icon is used also for folders to signal that they contain a file in real conflict or pseudo-conflict state.

Content Conflicts vs Property Conflicts

A **Content conflict** appears in the content of a file. A merge occurs for every inbound change to a file that is also modified in the working copy. In some cases, if the local change and the incoming change intersect each other, Apache Subversion™ cannot merge these changes without intervention. So if the conflict is real when updating the file in question the conflicting area is marked like this:

```
<<<<<<< filename
your changes
======
code merged from repository
>>>>>>> revision
```

Also, for every conflicted file Subversion places three additional temporary files in your directory:

- **filename.ext.mine** - This is your file as it existed in your working copy before you updated your working copy, that is without conflict markers. This file has your latest changes in it and nothing else.
- **filename.ext.rOLDREV** - This is the file that was the BASE revision before you updated your working copy, that is the file revision that you updated before you made your latest edits.
- **filename.ext.rNEWREV** - This is the file that Subversion client just received from the server when you updated your working copy. This file corresponds to the HEAD revision of the repository.

OLDREV and NEWREV are revision numbers. If you have conflicts with binary files, Subversion does not attempt to merge the files by itself. The local file remains unchanged (exactly as you last changed it) and you will get **filename.ext.r* files also**.

A **Property conflict** is obtained when two people modify the same property of the same file or folder. When updating such a resource a file named **filename.ext.prej** is created in your working copy containing the nature of the conflict. Your local file property that is in conflict will not be changed. After resolving the conflict, you should use the **Mark resolved** action to commit the file. Note that the **Mark resolved** action does not really resolve the conflict. It just removes the conflicted flag of the file and deletes the temporary files.
Edit Real Content Conflicts

The conflicts of a file in the conflicted state (a file with the red double arrow icon) can be edited visually with the Compare view (the built-in file comparison tool) or with an external diff application (on page 281). Resolving the conflict means deciding for each conflict if the local version of the change will remain or the remote one instead of the special conflict markers inserted in the file by the SVN server.

The Compare view (or the external diff application set in Preferences (on page 281)) is opened with the Edit Conflict action, which is available on the contextual menus of the Working Copy view (on page 2813) for files in the conflicted state (an update operation was executed but the differences could not be merged without conflicts). The external diff application is called with 3 parameters because it is a 3-way diff operation between the local version of the file from the working copy and the HEAD version from the SVN repository with the BASE version from the working copy as common ancestor.

If the Show warning dialog when edit conflicts option (on page 285) is selected, you will be warned at the beginning of the operation that the operation will overwrite the conflict version of the file received from the SVN server (the version that contains the conflict markers <<<<<<<<, ==========, >>>>>>>>) with the original local version of the file that preceded the update operation. If you click the OK button the visual conflict editing will proceed and a backup file of the conflict version received from the SVN server is created in the same working copy folder as the file with the edited conflicts. The name of the backup file is obtained by appending the extension .sync.bak to the file as stored on the SVN server. If you click the Cancel button the visual editing will be aborted.

The usual actions on the differences between two versions of a file are available on the toolbar of this view:

- **Save**
  Saves the modifications of the local version of the file displayed in the left side of the view.

- **Perform Files Differencing**
  Looks for differences between the two files displayed in the left and right side panels.

- **Ignore Whitespaces**
  Enables or disables the whitespace ignoring feature. Ignoring whitespace means that before performing the comparison, the application normalizes the content and trims its leading and trailing whitespaces.

- **Synchronized scrolling**
  Toggles synchronized scrolling. When toggled on, a selected difference can be seen in both panels.

- **Format and Indent Both Files** (Ctrl + Shift + P (Command + Shift + P on macOS))
  Formats and indents both files before comparing them. Use this option for comparisons that contain long lines that make it difficult to spot differences.
Note:
When comparing two JSON files, the **Format and Indent Both Files** action will automatically sort the keys in both files the same to make it easier to compare.

**Copy Change from Right to Left**
Copies the selected difference from the file in the right panel to the file in the left panel.

**Copy All Changes from Right to Left**
Copies all changes from the file in the right panel to the file in the left panel.

**Next Block of Changes (Ctrl + Period (Command + Period on macOS))**
Jumps to the next block of changes. This action is not available when the cursor is positioned on the last change block or when there are no changes.

Note:
A change block groups one or more consecutive lines that contain at least one change.

**Previous Block of Changes (Ctrl + Comma (Command + Comma on macOS))**
Jumps to the previous block of changes. This action is not available when the cursor is positioned on the first change block or when there are no changes.

**Next Change (Ctrl + Shift + Period (Command + Shift + Period on macOS))**
Jumps to the next change from the current block of changes. When the last change from the current block of changes is reached, it highlights the next block of changes. This action is not available when the cursor is positioned on the last change or when there are no changes.

**Previous Change (Ctrl + Shift + Comma (Command + Shift + M on macOS))**
Jumps to the previous change from the current block of changes. When the first change from the current block of changes is reached, it highlights the previous block of changes. This action is not available when the cursor is positioned on the first change or when there are no changes.

**First Change (Ctrl + B (Command + B on macOS))**
Jumps to the first change.

The operation begins by overwriting the conflict version of the file received from the SVN server (the version that contains the conflict markers <<<<<<<<, =======, >>>>>>>) with the original local version of the file before running the update action that created the conflict. After that the differences between this original local version and the repository version are displayed in the **Compare** view.

If you want to edit the conflict version of the file directly in a text editor instead of the visual editing offered by the **Compare** view you should work on the local working copy file after the update operation without running the action **Edit Conflict**. If you decide that you want to edit the conflict version directly after running the action **Edit Conflict** you have to work on the .sync.bak file.
If you did not finish editing the conflicts in a file at the first run of the action Edit Conflict you can run the action again and you will be prompted to choose between resuming the editing where the previous run left it and starting again from the conflict file received from the SVN server.

After the conflicts are edited and saved in the local version of the file you should run one of the following:

- The **Mark Resolved** action on the file so that the result of the conflict editing process can be committed to the SVN repository.
- The **Revert** action so that the repository version overwrites all the local modifications.

Both actions remove the backup file and other temporary files created with the conflict version of the local file.

### Revert Your Changes

If you want to undo changes made in your working copy, since the last update, select the items you are interested in, right-click to display the contextual menu and select **Revert**. A dialog box will open that shows you the files and folders that you have changed and can be reverted. Select those you want to revert and click the **OK** button. Revert will undo only your local changes. It does not undo any changes that have already been committed. If you choose to revert a conflicting item to its pristine copy, then the eventual conflict is solved by losing your outgoing modifications. If you try to revert a resource not under version control, the resource will be deleted from the file system.

**Note:**

By default, a directory will be recursively reverted (including any other modified item it contains). However, if the directory has only property changes, you need to explicitly choose if the operation will include any modified items found inside it.

If you want some of your outgoing changes to be overridden you must first open the file in **Compare view** (on page 2837) and choose the sections to be replaced with ones from the repository file. This can be achieved either by editing directly the file or by using the action **Copy change from right to left** from the **Compare view toolbar** (on page 2838). After editing the conflicting file you have to run the action **Mark as merged** before committing it.

If you want to drop all local changes and bring all incoming changes into your working copy resource, you can use the **Override and update** action. It discards the changes in the local file and updates it from the repository. A dialog box will display the files that will be affected.
In the first table of the dialog box you will be able to see the resources that will be overridden. In the second table you will find the list of resources that will be updated. Only resources that have an incoming status are updated.

Tip:
If you want to roll-back out of your working copy changes that have already been committed to the repository, see Merge Revisions (on page 2775).

Merge Conflicted Resources

Before you can safely commit your changes to the repository you must first resolve all conflicts. In the case of pseudo-conflicts they can be resolved in most cases with an update operation that will merge the incoming modifications into your working copy resource. In the case of real conflicts, conflicts that persist after an update operation, it is necessary to resolve the conflict using the built-in compare view and editor or, in the case of properties conflict, the Properties view (on page 2841). Before you can commit you must mark as resolved the affected files.

Both pseudo and real conflicts can be resolved without an update. You should open the file in the compare editor and decide which incoming changes need to be copied locally and which outgoing changes must
be overridden or modified. After saving your local file you have to use the Mark as merged action from the contextual menu before committing.

**Drop Incoming Modifications**

In the situation when your file is in conflict but you decide that your working copy file and its content is the correct one, you can decide to drop some or all of the incoming changes and commit afterwards. The action Mark as merged proves to be useful in this case too. After opening the conflicting files with Compare view (on page 2837), Editor (on page 2834) or editing their properties in the Properties view and deciding that your file can be committed in the repository replacing the existing one, you should use the Mark as merged action. When you want to override completely the remote file with the local file you should run the Override and commit action, which drops any remote changes and commits your file.

In general it is much safer to analyze all incoming and outgoing changes using the Compare view and only after to update and commit.

**Tree Conflicts**

A tree conflict is a conflict at the directory tree structure level and occurs when the user runs an update action on a resource that has the following conditions:

- It is locally modified and the same resource was deleted from the repository (or deleted as a result of being renamed or moved).
- It was locally deleted (or deleted as a result of being renamed or moved) and the same resource is incoming as modified from the repository.

The same conflict situation can occur after a merge or a switch action. The action ends with an error and the folder containing the file that is now in the tree conflict state is also marked with a conflict icon.

Such a conflict can be resolved in one of the following ways that are available when the user double clicks on the conflicting resource or when running the Edit conflict action:
Figure 691. Resolve a tree conflict

• **Keep local change (delete resource)** - Keeps the incoming change that comes from the repository.
• **Keep incoming modified resource** - If there is a renamed version of the file committed by other user that will be added to the working copy too.

**Update the Working Copy**

While you are working on a project, other members of your team may be committing changes to the project repository. To get these changes, you have to **update** your working copy. Updating may be done on single files, a set of selected files, or recursively on entire directory hierarchies. The update operation can be performed from **Working Copy view (on page 2813)**. It updates the selected resources to the last synchronized revision (if remote information is available) or to the **HEAD** revision of the repository.

There are three different kinds of incoming changes:

• **Non-conflicting** - A non-conflicting change occurs when a file has been changed remotely but has not been modified locally.
• **Conflicting, but auto-mergeable** - An auto-mergeable conflicting change occurs when a text file has been changed both remotely and locally (for example, has non-committed local changes) but the changes are on different lines of text. Not applicable to binary resources (for example, multimedia files, PDFs, executable program files)
• **Conflicting** - A conflicting change occurs when one or more of the same lines of a text file have been changed both remotely and locally.
If the resource contains only incoming changes or the outgoing changes do not intersect with incoming ones then the update will end normally and the Subversion system will merge incoming changes into the local file. In the case of a conflicting situation the update will have as result a file with conflict status.

The Oxygen XML Editor allows you to update your working copy files to a specific revision, not only the most recent one. This can be done by using the Update to revision/depth action from the Working Copy view (All Files view mode) or the Update to revision action from the History view (on page 2828) contextual menu.

If you select multiple files and folders and then you perform an Update operation, all of those files and folders are updated one by one. The Subversion client makes sure that all files and folders belonging to the same repository are updated to the exact same revision, even if between those updates another commit occurred.

When the update fails with a message saying that there is already a local file with the same name Subversion tried to check out a newly versioned file, and found that an unversioned file with the same name already exists in your working folder. Subversion will never overwrite an unversioned file unless you specifically do this with an Override and update action. If you get this error message, the solution is simply to rename the local unversioned file. After completing the update, you can check to see if the renamed file is still needed.

**Send Your Changes to the Repository**

Sending the changes you made to your working copy is known as committing the changes. If your working copy is up-to-date and there are no conflicts, you are ready to commit your changes.

The Commit action sends the changes from your local working copy to the repository. The Commit dialog box presents all the items that you can commit.
Figure 692. Commit dialog box

Enter a message to associate with the commit, or choose a previous message from the Previous messages list (the last 10 commit messages will be remembered even after restarting the SVN client application).

An item that can be committed has one of the following states: added, modified (content or properties), replaced, and deleted. All items that have one of these states are selected in the dialog box by default. If you do not want to commit one of the items, deselect it.

Attention:
For SVN 1.8 working copies: when committing items that were moved and/or renamed, make sure you select both the source and the destination. Otherwise, the commit operation will fail.

Besides the items that have one of the mentioned states, Syncro SVN Client also includes the files being unversioned or missing and these items are handled automatically:

- Unversioned items are added under version control.
- Missing items are deleted.
Note:
If the **Show unversioned directories content** option is not selected, the **Commit** dialog box does not display the items inside an *unversioned* directory.

*Unversioned or missing* items are not selected by default in the **Commit** dialog box, unless you have selected them explicitly when issuing the commit command.

Note:
In some cases, items that have one of the above states are not presented in the **Commit** dialog box.

For example:

- Items that have been *added* or *replaced* previously, but now are presented as *missing* after being removed from the file system, outside of an SVN client. Such items do not exist in the repository and you should use the **Delete** action to remove them from your working copy.
- Items that have incoming changes from the repository, after a synchronization. You need to have your working copy up-to-date before committing your changes.
- Files that, after a synchronization, appear as locked by other users or from other locations than the current working copy.

Note:
Due to dependencies between items, when you select or clear an *unversioned* (*miş*) or *added* (*eklenmiştir*) item in the **Commit** dialog box, other items with one of these states can be selected or cleared automatically.

The modifications that will be committed for each file can be reviewed in the compare editor window by double-clicking a file in the **Commit** dialog box, or by right-clicking and selecting the **Show Modifications** action from the contextual menu. This option is available to review only file content changes, not property changes.

The **Local file status** column indicates the actual state of the items and the **Local properties status** column indicates whether or not the properties of an item are modified.

The **Lock information** column is displayed if at least one of the files in the **Commit** dialog box has lock information associated with it, valid against the commit operation.

The following options are available in this dialog box:
- **Enable automatic properties** or **Disable automatic properties** - enables or disables automatic property assignment (per runtime configuration rules), overriding the `enable-auto-props` runtime configuration directive, defined in the `config` file of the Subversion configuration directory.

**Note:**
This option is available only when there are defined properties to be applied automatically for resources newly added under version control. You can define these properties in the `config` file of the Subversion configuration directory, in the `auto-props` section. Based on the value of the `enable-auto-props` runtime configuration directive, the presented option is either **Enable automatic properties**, or **Disable automatic properties**.

- **Keep locks** - selecting the **Keep locks** option preserves any locks you set on various files.

**Note:**
This option is available only when files that you locked are presented in the dialog box.

Each of the above options is activated only when you select an item that can have the option applied.

Your working copy must be up-to-date with respect to the resources you commit. This is ensured by using the **Update** action prior to committing, resolving conflicts and re-testing as needed. If your working copy resources you are trying to commit are out of date you will get an appropriate error message.

### Committing to Multiple Locations

Although Subversion does not support committing to multiple locations at once, Syncro SVN Client offers this functionality regarding external items.

If items to be committed belong to different external definitions than those found in the working copy, they are grouped under the corresponding item that indicates their repository origin. Each parent item is rendered bold and its corresponding repository location is presented when hovering it. Parent items are decorated with a small arrow (▲) if they are external definitions. The working copy root directory is never decorated and is not presented if there are no external items listed (all items belong to the main working copy). Each child item is presented relative to the parent item.

**Note:**
When an external directory has modifications of its own, it is presented both as a parent item and as an item that you can select and commit. This is always the case for external files.

The sets of items belonging to external definitions from the same repository are committed together, resulting a single revision. So, the number of revisions can be smaller than the number of externals. External definitions are considered from the same repository if they have the same protocol, server address, port, and repository address within the server.
Integration with Bug Tracking Tools

Users of bug tracking systems can associate the changes they make in the repository resources with a specific ID in their bug tracking system. The only requirement is that the user includes the bug ID in the commit message that they enter in the Commit dialog box. The format and the location of the ID in the commit message are configured with SVN properties.

To make the integration possible Syncro SVN Client needs some data about the bug tracking tool used in the project. You can configure this using the following SVN properties (on page 2767) that must be set on the folder that contains resources associated with the bug tracking system (usually they are set recursively on the root folder of the working copy):

- **bugtraq:message** - A string property. If it is set the Commit dialog box (on page 2761) will display a text field for entering the bug ID. It must contain the string %BUGID%, which is replaced with the bug number on commit.
- **bugtraq:label** - A string property that sets the label for the text field configured with the bugtraq:message property.
- **bugtraq:url** - A string property that is the URL pointing to the bug tracking tool. The URL string should contain the substring %BUGID% which Syncro SVN Client replaces with the issue number. That way the resulting URL will point directly to the correct issue.
- **bugtraq:warnifnoissue** - A boolean property with the values true/yes or false/no. If set to true, the Syncro SVN Client will warn you if the bug ID text field is left empty. The warning will not block the commit, only give you a chance to enter an issue number.
- **bugtraq:number** - A boolean property with the value true or false. If this property is set to false, then any character can be entered in the bug ID text field. If the property is set to true or is missing then only numbers are allowed as the bug ID.
- **bugtraq:append** - A boolean property. If set to false, then the bug ID is inserted at the beginning of the commit message. If yes or not set, then it is appended to the commit message.
- **bugtraq:logregex** - This property contains one or two regular expressions, separated by a newline. If only one expression is set, then the bug ID's must be matched in the groups of the regular expression string (for example, [Ii]ssue #?\(\d+\)). If two expressions are set, then the first expression is used to find a string which relates to a bug ID but may contain more than just the bug ID (for example, Issue #123 or resolves issue 123). The second expression is then used to extract the bug ID from the string extracted with the first expression. An example: if you want to catch every pattern issue #XXX and issue #890, #789 inside a log message you could use the following strings:
  ◦ [Ii]ssue #?\(\d+)\(, . ?#?\(\d+\)\)\+
  ◦ \(\d+\)
The data configured with these SVN properties is stored on the repository when a revision is committed. A bug tracking system or a statistics tool can retrieve the revisions that affected a bug from the SVN server and present the commits related to that bug to the user of the bug tracking system.

If the `bugtraq:url` property was filled in with the URL of the bug tracking system and this URL includes the `%BUGID%` substring as specified above in the description of the `bugtraq:url` property then the History view (on page 2828) presents the bug ID as a hyperlink in the commit message. Clicking such a hyperlink in the commit message of a revision opens a Web browser at the page corresponding to the bug affected by that commit.

Obtain Information for a Resource

This section explains how to obtain information for a SVN resource:

Request Status Information for a Resource

While you are working with the SVN Client you often need to know which files you have changed, added, removed, or renamed, or even which files got changed and committed by others. This is where the Synchronize action from the Working Copy view (on page 2813) comes in handy. The Working Copy view shows you every file that has changed your working copy, as well as any unversioned files you may have.

If you want more detailed information about a given resource, you can use the Show SVN Information action. This action is available from the File menu or the contextual menu of the Working Copy, Repositories, History, or Directory Change Set views, or from the Revision Graph dialog box. The SVN Information dialog box will be displayed, showing information about the selected resource. The information displayed depends on the location of the item (local or remote) and may include the following:

- Local path and repository location
- Revision number
- Last change author, revision and date
- Information about locks
- Local file status
- Local properties status
- Local directory depth
- Repository location and revision number for copied files or directories
- Path information about locally moved items
- Path information about conflict generated files
- Remote file status
- Remote properties status
- File size and other information

The value of a property of the resource displayed in the dialog box can be copied by right-clicking the property and selecting the Copy action.
Request History for a Resource

In Apache Subversion™, both files and directories are versioned and have a history. If you want to examine the history for a selected resource and find out what happened at a certain revision you can use the History view that can be accessed from Repositories view (on page 2808), Working Copy view (on page 2813), Revision Graph (on page 2843), or Directory Change Set view (on page 2833). From the Working copy view you can display the history of local versioned resources. If the view is not displayed, it can be opened by selecting it from the Window > Show View menu.

Related Information:
- History View (on page 2828)

Management of SVN Properties

In the Properties view (on page 2841) you can read and set the Apache Subversion™ properties of a file or folder. There is a set of predefined properties with special meaning to Subversion. For more information about properties in Subversion see the SVN Subversion specification. Subversion properties are revision-dependent. After you change, add or delete a property for a resource, you have to commit your changes to the repository.

If you want to change the properties of a given resource you need to select that resource from the Working Copy view (on page 2813) and run the Show properties action from the contextual menu. The Properties view (on page 2841) will show the local properties for the resource in the working copy. Once the Properties view is visible, it will always present the properties of the currently selected resource. There are actions available in the Properties view toolbar (on page 2842) that allows you to add, change, and delete the properties.

If you choose the Add a new property action, a new dialog box will appear that contains the following:

- Name - Combo box that allows you to enter the name of the property. The drop-down menu of the combo box presents the predefined Subversion properties (such as svn:ignore, svn:externals, svn:needs-lock, etc.)
- Current value - Text area that allows you to enter the value of the new property.

If the selected item is a directory, you can also set the property recursively on its children by selecting the Set property recursively checkbox.

If you want to change the value for a previously set property, you can use the Edit property action, which will display a dialog box with the following information:

- Name - Property name (cannot be changed).
- Current value - The current value (can be changed).
- Base value - The value of the property, if any, from the resource in the pristine copy (cannot be changed).
If you want to completely remove a property previously set you can choose the **Remove property** action. It will display a confirmation dialog box where you can also choose if the property will be removed recursively.

There is a **Refresh** action in the **Properties view** that can be used when the properties have been changed from outside the view. This can happen, for example, when the view was already presenting the properties of a resource and they have been changed after an **Update** operation.

### Branches and Tags

One of the fundamental features of version control systems is the ability to create a new line of development from the main one. This new line of development will always share a common history with the main line if you look far enough back in time. This line is known as a **branch**. Branches are mostly used to try out features or fixes. When the feature or fix is finished, the branch can be merged back into the main branch (**trunk**).

Another feature of version control systems is the ability to take a snapshot of a particular revision, so you can at any time recreate a certain build or environment. This is known as **tagging**. Tagging is especially useful when making release versions.

In **Apache Subversion™**, there is no difference between a **tag** and a **branch**. On the repository, both are ordinary directories that are created by copying. The trick is that they are cheap copies instead of physical copies. Cheap copies are similar to hard links in Unix, which means that they merely link to a specific tree and revision without making a physical copy. As a result, branches and tags occupy little space on the repository and are created very quickly.

Provided that nobody ever commits to the directory in question, it remains a tag. If people start committing to it, it becomes a branch.

### Create a Branch / Tag

To create a branch or tag by copying a directory, use the **Branch/Tag** action that is available in the **Tools** menu when an item is selected in the **Working Copy view** or **Repositories view**, or from the contextual menu of the **Repositories view**.
You can configure the following options in this dialog box:

You can specify the source revision of the copy in the **Copy from** section. You can choose between the following options:

- **HEAD revision in the repository** - The new branch or tag will be copied in the repository from the HEAD revision. The branch will be created very quickly, as the repository will make a cheap copy.

- **Specific revision in the repository** - The new branch will be copied into the repository, but you can specify the exact desired revision. For example, this is useful if you forgot to make a branch or tag when you released your application. If you click the **History** button you can select the revision number from the **History** dialog box (on page 2741). This type of branch will also be created very quickly.

- **Working copy** - (Available only if the item is selected from the **Working copy** view). The new branch will be a copy of your local working copy. If you have updated some files to an older revision in your working copy, or if you have made local changes, that is exactly what goes into the copy. This involves transferring some data from your working copy back to the repository, or more specifically, the locally modified files.

You can specify the location of the new branch or tag in the **Destination** section:
• **Into repository’s directory** - The URL of the parent directory (on page 2848) of the new branch or tag.

> **Note:**

*Peg revisions* have no effect for this operation since it is used to send information to the repository.

• **Under the name** - You can specify another branch or tag name other than the name of the resource selected in the **Repositories** or **Working copy** view.

The new branch or tag will be created as a child of the specified URL of the repository directory and will have the new name.

### Merging

At some stage during the development process, you will want to merge the changes made on a *branch* back into the *trunk*, or vice-versa. The *merge* is accomplished by comparing two points (branches or revisions) in the repository and applying the obtained differences to your working copy. This process is closely related to the *diff* concept.

> **Note:**

A *branch* is a line of development that exists independently of another line, yet still shares a common history if you look far enough back in time. A *branch* always begins life as a *copy of something* (such as a trunk, another branch, or tag), and moves on from there, generating its own history.

The ➔ **Merge** action is available in the **Tools** menu. The working copy item selected when you issued the command will be the one receiving the generated changes. If there is no item selected, the **merge** operation will be performed on the entire working copy.
The four types of merging are as follows:

- **Merge revisions** (on page 2774) - Port changes from one branch to another. Note that the trunk can also be considered a branch, in this context.
- **Synchronize branch** (on page 2776) - Fetch all the changes made on a parent branch (or the trunk) to a child branch.
- **Reintegrate a branch** (on page 2778) - Merge a branch back to its parent branch (can also be the trunk).
- **Merge two different trees** (on page 2780) - Integrate the changes done on a branch to a different branch.

It is recommended that you enable the following pre-merge check:

**Perform pre-merge best practices checks of the working copy target** (on page 2772) - When selected, the SVN Client checks if the working copy target item is ready for the merge operation and displays the **pre-merge checks** wizard page.
Remember:
It is a good idea to perform a merge into an unmodified working copy. If you have made changes to your working copy, commit them first. If the *merge* does not go as you expect, you may want to revert the changes and revert cannot recover your uncommitted modifications.

Important:
The above recommendation becomes mandatory when *reintegrating a branch (on page 2770)*.

Pre-Merge Checks

Before performing a merge, it is recommended to make sure that the working copy target item is ready for the merge operation. The SVN Client includes a best practices step that checks various conditions of the working copy target item to ensure that the merge operation will succeed. By selecting the **Perform pre-merge best practices checks of the working copy target** option in the first page of the Merge wizard, the **Pre-merge checks** wizard page is displayed to give you a summary of the verified conditions.

![Pre-Merge Checks Wizard Page](image)

The following conditions are checked in this operation:

**No local modifications**
The working copy item (or any of its children) receiving the merge should not contain uncommitted changes, to make it easier to revert merge-generated changes if you encounter unexpected results.

**Tip:**
If this condition fails, you should commit or revert the local modifications before merging.

### No switched children

None of the children of the working copy item receiving the merge should be switched, to avoid incomplete merges and subtree mergeinfo.

**Tip:**
If this condition fails, you should switch back all the children before merging.

### Complete working copy tree

The working copy item receiving the merge should be a complete directory tree structure with an infinite depth, to avoid incomplete merges and subtree mergeinfo.

**Tip:**
If this condition fails, you should change the sticky depth of the working copy item receiving the merge to infinity value.

### No mixed revisions

To avoid unexpected merge conflicts, the working copy item that is receiving the merge should not contain items that were updated to other revisions.

**Tip:**
If this condition fails, you should update the working copy before merging.

Each condition is marked with an icon that represents the state of the condition. The possible states are as follows:

- **✓ (Successful)** - The condition is fulfilled successfully.
- **⚠️ (Warning)** - The condition is not fulfilled, but it is not mandatory.
- **⚠️ (Error)** - The condition is not fulfilled and is mandatory (therefore, the operation cannot proceed until you solve the error).
Tip:
For each condition state, a message is displayed that gives you additional information about the results and, for warning or errors, a hint that explains how you can solve them.

Important:
After solving any of the warnings or errors, it is recommended that you perform the pre-merge checks again to make sure your new changes are valid.

Merge Revisions

This case is when you have made one or more changes to a branch and you want to duplicate them in another branch. For example, suppose you know that a problem has been fixed by committing revisions 17, 20, and 25 on branch B1. These changes are also needed in branch B2. Thus, to merge them, you need a working copy of the B2 branch.

To merge revisions from a different branch, follow these steps:

1. Go to menu Tools > Merge.
The Merge wizard is opened.
2. Select the Merge revisions option.
3. It is recommended that you select the Perform pre-merge best practices checks of the working copy target option to make sure that the working copy target item is ready for the merge operation.
   a. Click the Next button.
      If the Perform pre-merge best practices checks of the working copy target option is selected, the Pre-Merge Checks wizard page (on page 2772) is displayed.

   Note:
   If errors are found you need to solve them before proceeding.

4. Click the Next button.
The Merge revisions wizard page is displayed.
5. In the Merge from (URL) text box, enter the URL of the branch or tag (on page 2848) that contain the changes that you want to duplicate in your working copy.
   You may also click the Browse button to browse the repository and find the desired branch. If you have previously merged from this branch, then you can simply use the drop-down menu, which displays a history of previously used URLs.

   Note:
   If the URL belongs to a different repository than the working copy, the Ignore ancestry / Disable merge tracking option (in the Merge Options wizard page (on page 2783)) will be selected.
automatically (and you cannot change this). This is because the Subversion client cannot track changes between different repositories (on page 2786).

Tip:
You can also specify a peg revision (on page 2850) at the end of the URL (for example, URL@rev1234). The peg revision does not affect the merge range you select. By default, the HEAD revision is assumed.

6. In the Revisions to merge section, choose between the all revisions and specific revision(s) options.
   - all revisions - The operation will include all eligible revisions that were not yet merged.
   - specific revision(s) - You can specify one or more individual revisions and/or revision ranges. Also, you can mix forward ranges (for example, 1-5), backward ranges (for example, 20-15), and subtract specific revisions from a range (for example, 1-5, -3).

Note:
If using the Subversion command-line client, a revision range of the form 1-5 means all changes starting from revision 2 up to revision 5 (the changes necessary to reach revision 5, committed after revision 1). Unlike the Subversion command-line client, in Syncro SVN Client the revision ranges are inclusive, meaning that it will process all revisions, starting with revision 1, up to and including revision 5.

Attention:
The HEAD revision is the only non-numerical revision allowed, and it can only be used when specifying revision ranges as one of the ends of the range (for example, 10-HEAD). Be careful when using it, as it might not refer to the desired revision, if it has recently been committed by another user.

Tip:
If you want to perform a reverse merge and roll-back your working copy changes that have already been committed to the repository, use the negative revisions notation (for example, -7) or backward revision ranges (for example, 20-10).

   a. If you click the History button, the History dialog box (on page 2741) is displayed, which allows you to select one or more revisions to be merged.

7. Optionally, if you want to configure the options (on page 2783) for your merge, click the Next button. The Merge Options wizard page (on page 2783) is displayed that allows you to configure options for the operation.
Warning:
If the Ignore ancestry / Disable merge tracking option is selected and you chose all revisions in the Revisions to merge section, revisions that were previously merged will also be included, which may result in conflicts.

8. Click the Merge button.
   The merge operation is performed.

If the merge is completed successfully, all the changes corresponding to the selected revisions should be merged in your working copy.

It is recommended to look at the results of the merge, in the working copy, to review the changes and see if it meets your expectations. Since merging can sometimes be complicated, you may need to resolve conflicts (on page 2785) after making major changes.

Note:
The merge result is only in your local working copy and needs to be committed to the repository for it to be available to others.

Synchronize a Branch

While working on your own branch, other people on your team might continue to make important changes in the parent branch (which can be the trunk itself or any other branch). It is recommended to periodically duplicate those changes in your branch to make sure your changes are compatible with them. This is done by performing a synchronize merge, which will bring your branch up-to-date with any changes made to its ancestral parent branch since your branch was last created or synchronized. Subversion is aware of the history of your branch and can detect when it split away from the parent branch.

Frequently keeping your branch in sync with the parent branch helps you to prevent unexpected conflicts when the time comes for you to duplicate your changes back into the parent branch. The synchronization uses merge tracking to skip all those revisions that have already been merged, thus a sync merge can be repeated periodically to fetch all the latest changes of the parent branch to keep up-to-date with it.

Important:
It is recommended to synchronize the whole working copy that was created from the child branch (the root of the working copy), rather than just a part of it.

After running the synchronize merge, your working copy from the child branch now contains new local modifications, and these edits are duplications of all of the changes that have happened on the trunk since you first created your branch. At this point, your private branch is now synchronized with the trunk.

To synchronize your branch with its parent branch, follow these steps:
1. Go to Tools > Merge.
   The Merge wizard is opened.
2. Select the Synchronize branch option.
3. It is recommended that you select the Perform pre-merge best practices checks of the working copy target option to make sure that the working copy target item is ready for the merge operation.

   a. Click the Next button.
      If the Perform pre-merge best practices checks of the working copy target option is selected, the Pre-Merge Checks wizard page (on page 2772) is displayed.

      Note:
      If errors are found you need to solve them before proceeding.

4. Click the Next button.
   The Synchronize branch wizard page is displayed.
5. In the Parent branch (URL) text box, enter the URL of the branch where you created your branch (on page 2848). This means that the URL must belong to the same repository as your working copy that was created from the child branch.
   You may also click the Browse button to browse the repository and find the desired branch. If you have previously merged from this branch, then you can simply use the drop-down menu, which displays a history of previously used URLs.

   Tip:
   You can also specify a peg revision (on page 2850) at the end of the URL (for example, URL@rev1234). The peg revision specifies both the peg revision of the URL and the latest revision that will be considered for merging. By default, the HEAD revision is assumed.

6. Optionally, if you want to configure the options (on page 2783) for your merge, click the Next button.
   The Merge Options wizard page (on page 2783) is displayed that allows you to configure options for the operation.

   Note:
   The Ignore ancestry / Disable merge tracking option is not available for this merge type, since a synchronization merge should always be recorded in the destination branch.

7. Click the Merge button.
   The merge operation is performed.

If the merge is completed successfully, all the changes corresponding to the selected revisions should be merged in your working copy.
It is recommended to look at the results of the merge, in the working copy, to review the changes and see if it meets your expectations. Since merging can sometimes be complicated, you may need to resolve conflicts (on page 2785) after making major changes.

**Note:**
The merge result is only in your local working copy and needs to be committed to the repository for it to be available to others.

### Reintegrate a Branch

**Prerequisites:**
There are some conditions that apply to reintegrate a branch:

- The server must support merge tracking.
- The source branch (to be reintegrated) must be coherently synchronized with its parent branch. This means that all revisions between the branching point and the last revision merged from the parent branch to the child branch must be merged to the latter one (there must be no missing revisions in-between).
- The working copy **must not** contain the following:
  - Local modifications.
  - A mixture of revisions (all items must point to the same revision).
  - Sparse directories (all directories must be of *infinity* depth).
  - Switched items.
- The revision of the working copy must be greater than or equal to the last revision of the parent branch with which the child branch was synchronized.

**Tip:**
You can use the pre-merge checks option (on page 2772) to make sure these conditions are fulfilled.

This method is useful when you have a feature branch on which the development has concluded and it should be merged back into its parent branch. Since you have kept the feature branch synchronized with its parent, the latest versions of them will be absolutely identical except for your feature branch changes. These changes can be reintegrated into the parent branch by using a working copy of it and the Reintegrate a branch option.

This method uses the merge-tracking features of Apache Subversion™ to automatically calculate the correct revision ranges and to perform additional checks that will ensure that the branch to be reintegrated has been fully updated with its parent changes. This ensures that you do not accidentally undo work that others have committed to the parent branch since the last time you synchronized the child branch with it. After the merge, all branch development will be completely merged back into the parent branch, and the child branch will be redundant and can be deleted from the repository.
Tip:
Before reintegrating the child branch it is recommended to synchronize it with its parent branch one more time, to help avoid any possible conflicts.

To reintegrate a child branch into its parent branch, follow these steps:

1. Go to menu Tools > Merge.
   The Merge wizard is opened.
2. Select the Reintegrate a branch option.

Note:
This option is not available if the selected working copy item (or if it is a directory, any of the items inside of it) has any type of modification. This is because it is mandatory for the target item to have no modifications.

3. It is recommended that you select the Perform pre-merge best practices checks of the working copy target option to make sure that the working copy target item is ready for the merge operation.
   a. Click the Next button.
      If the Perform pre-merge best practices checks of the working copy target option is selected, the Pre-Merge Checks wizard page (on page 2772) is displayed.

Note:
If errors are found you need to solve them before proceeding.

4. Click the Next button.
   The Reintegrate a branch wizard page is displayed.
5. In the Child branch (URL) text box, enter the URL of the child branch to be reintegrated (on page 2848).
   This means that the URL must belong to the same repository as your working copy that was created from the parent branch.
   You may also click the Browse button to browse the repository and find the desired branch. If you have previously merged from this branch, then you can simply use the drop-down menu, which displays a history of previously used URLs.

Tip:
You can also specify a peg revision (on page 2850) at the end of the URL (for example, URL@rev1234). The peg revision specifies both the peg revision of the URL and the latest revision that will be considered for merging. By default, the HEAD revision is assumed.

The Merge Options wizard page (on page 2783) is displayed that allows you to configure options for the operation.
6. Click the **Merge** button.
   The merge operation is performed.

If the merge is completed successfully, all the changes corresponding to the selected revisions should be merged in your working copy.

It is recommended to look at the results of the merge, in the working copy, to review the changes and see if it meets your expectations. Since merging can sometimes be complicated, **you may need to resolve conflicts (on page 2785)** after making major changes.

**Note:**
The merge result is only in your local working copy and needs to be committed to the repository for it to be available to others.

### Merge Two Different Trees

This merge type is useful when you need to duplicate changes from one child branch (for example, cb1) to another child branch (cb2) from the same parent branch. The SVN client will calculate the changes necessary to get from the **HEAD** revision of the parent branch (or the **trunk**) to the **HEAD** revision of one of its child branches (cb1), and apply those changes to your working copy of the other branch (cb2). The result is that the latter child branch (cb2) will also include the changes made on the original child branch (cb1), although that branch was not reintegrated into the parent branch.

This merge type could also be used to reintegrate a child branch back into its parent when the repository does not support **merge tracking**.

**Note:**
If the server does not support **merge-tracking**, then this is the only way to merge a branch back to its parent.

1. Go to menu **Tools > Merge**.
   The Merge wizard is opened.
2. Select the option **Merge two different trees**.
3. It is recommended that you select the **Perform pre-merge best practices checks of the working copy target** option to make sure that the working copy target item is ready for the merge operation.
a. Click the **Next** button.

If the **Perform pre-merge best practices checks of the working copy target** option is selected, the **Pre-Merge Checks wizard page** (on page 2772) is displayed.

![Note:]

If errors are found you need to solve them before proceeding.

4. Click the **Next** button.

The **Merge two different trees** wizard is displayed.

5. In the **From (starting URL and revision)** section, enter the URL of the first branch (on page 2848).

You may also click the **Browse** button to browse the repository and find the desired branch. If you have previously merged from this branch, then you can simply use the drop-down menu, which displays a history of previously used URLs.

![Tip:]

If you are using this method to merge a feature branch back to its parent branch, you need to start the merge wizard from within a working copy of the parent. In this field enter the full URL of the parent branch. This may sound wrong, but remember that the parent is the starting point to which you want to add the branch changes.

![Note:]

If the URL belongs to a different repository than the working copy, the **Ignore ancestry / Disable merge tracking** option (in the **Merge Options wizard page** (on page 2783)) will be selected automatically (and you cannot change this). This is because the **Subversion client cannot track changes between different repositories** (on page 2786).

![Tip:]

You can also specify a **peg revision** (on page 2850) at the end of the URL (for example, URL@rev1234). By default, the **HEAD** revision is assumed.

6. Enter the last revision number at which the two trees were synchronized by choosing between **HEAD revision** and **other revision**.

   - **HEAD revision** - Use this option if you are sure that no one else has committed changes since the last synchronization.
   - **other revision** - Use this option to input a specific revision number and avoid losing recent commits. You can use the **History** button to see a list of all revisions.

7. In the **To (ending URL and revision)** section, enter the URL of the second branch (on page 2848).

You may also click the **Browse** button to browse the repository and find the desired branch. If you have previously merged from this branch, then you can simply use the drop-down menu, which displays a history of previously used URLs.
Tip:
If you are using this method to merge a feature branch back to its parent branch, enter the URL of the feature branch. This way, only the changes unique to this branch will be merged, since the branch should have been periodically synchronized with its parent.

Attention:
The URL must point to the same repository as the one in the From (starting URL and revision) field. Otherwise, the operation will not be allowed, since Subversion cannot compute changes between items from different repositories.

Tip:
You can also specify a peg revision (on page 2850) at the end of the URL (for example, URL@rev1234). By default, the HEAD revision is assumed.

8. Select a revision to compute all changes committed up to that point by choosing between HEAD revision and other revision.
   - HEAD revision - This is the default selected revision.
   - other revision - Use this option if you want to enter a previous revision. You can use the History button to see a list of all revisions.

9. Optionally, if you want to configure the options (on page 2783) for your merge, click the Next button. The Merge Options wizard page (on page 2783) is displayed that allows you to configure options for the operation.

   Warning:
   If the Ignore ancestry / Disable merge tracking option is selected and you chose all revisions in the Revisions to merge section, revisions that were previously merged will also be included, which may result in conflicts.

10. Click the Merge button.
    The merge operation is performed.

    If the merge is completed successfully, all the changes corresponding to the selected revisions should be merged in your working copy.

    It is recommended to look at the results of the merge, in the working copy, to review the changes and see if it meets your expectations. Since merging can sometimes be complicated, you may need to resolve conflicts (on page 2785) after making major changes.
Note:
The merge result is only in your local working copy and needs to be committed to the repository for it to be available to others.

Merge Options

Here is the list of options that can be used when merging:

**Figure 696. Merge Wizard - Advanced Options**

- **Depth** (This option is applicable only for directories) - sets the depth of the merge operation. You can specify how far down into your working copy the merge should go by selecting one of the following values:
  - **Current depth** - Obey the depths registered for the directories in the working copy that are to be switched.
  - **Recursive (infinity)** - Merges all the files and folders contained in the selected folder. This is the recommended depth for most users, to avoid incomplete merges and subtree mergeinfo.
  - **Immediate children (immediates)** - Merges only the child files and folders without recursing subfolders.
  - **File children only (files)** - Merges only the child files.
  - **This folder only (empty)** - Merges only the selected folder (no child files or folders are included).
Note:
The depth term is described in the Sparse checkouts (on page 2807) section. The default depth is the current depth of the working copy item receiving the merge.

• **Ignore ancestry / Disable merge tracking** - Changes the way two items are merged if they do not share a common ancestry. Most merges involve comparing items that are ancestrally related to one another. However, occasionally you may want to merge unrelated items. If this option is not selected, the first item will be replaced with the second item. In these situations, you would want the merge to do a path-based comparison only, ignoring any relations between the items. For example, if two different files have the same name and are in the same relative location, deselecting the option replaces one of the files with the other one, and selecting it merges their contents.

Note:
If the URL of the merge source belongs to a different repository than the URL of the target working copy item (the one receiving the changes), this option is selected automatically (and you cannot change this). This is because the Subversion client cannot track changes between different repositories (on page 2786).

• **Force deletion of modified or non-versioned items, if necessary** - If not selected, when the merge operation involves deleting locally modified or non-versioned items, it will fail. This is done to prevent data loss. This option is only available if there are uncommitted changes in the working copy.

• **Only record the merge (block revisions from getting merged)** - Available when the Ignore ancestry / Disable merge tracking option (on page 2784) is not selected. It enables a special mode of the merge operation that just records it in the local merge tracking information, without actually performing it (does not modify any file contents or the structure of your working copy). You might want to select this option for two possible reasons:
  ◦ You made (or will make) the merge manually, and therefore need to mark the revisions as being merged to make the merge tracking system aware of them. This will exclude them from future merges.
  ◦ You want to prevent one or more particular changes from being fetched in subsequent merges.

• **Ignore line endings** - Allows you to specify how the line ending changes should be handled. By default, all such changes are treated as real content changes, but you can ignore them if you select this option.

• **Ignore whitespaces** - Allows you to specify how the whitespace changes should be handled. By default, all such changes are treated as real content changes, but you can ignore them if you select this option.
• **Ignore whitespace changes** - Ignores changes in the amount of whitespaces or to their type (for example, when changing the indentation or changing tabs to spaces).

  **Note:**
  Whitespaces that were added where there were none before, or that were removed, are still considered to be changes.

• **Ignore all whitespaces** - Ignores all types of whitespace changes.

• **Test merge** - Performs a dry run of the merge operation, allowing you to preview it without actually performing the merge. In the **Console** view you will see a list of the working copy items that will be affected and how they will be affected. This is helpful in detecting whether or not a merge will be successful, and where conflicts may occur.

### Resolving Merge Conflicts

After the merge operation is finished, it is possible to have some items in conflict. This means that some incoming modifications for an item could not be merged with the current working copy version. If there are such conflicts, the **Merge conflicts** dialog box will appear, presenting the items that are in conflict. This dialog box offers you choices for resolving the conflicts.

#### Figure 697. Merge Conflicts Dialog Box

The options to resolve a conflict are as follows:
• **Resolve later** - Used for leaving the conflict as it is, to manually resolve it later.
• **Keep incoming** - This option keeps all the incoming modifications and discards all current ones from your working copy.
• **Keep outgoing** - This option keeps all current modifications from your working copy and discards all incoming ones.
• **Mark resolved** - You should choose this option after you have manually solved the conflict, to instruct the Subversion that it was resolved. To do this, use the **Edit conflict** button, which displays a dialog box that presents the contents of the conflicting items (the content of the working copy version versus the incoming version).

**Additional Notes About the Merge Operation**

**Sub-tree Merges**

It is recommended to perform a merge on the whole working copy (select its root directory when triggering the operation) to avoid **sub-tree mergeinfo**. **Sub-tree mergeinfo** is the **mergeinfo** recorded to describe a **sub-tree merge**. That is, a merge done directly to a child of a branch root that might be needed in certain situations. There is nothing special about **sub-tree merges** or **sub-tree mergeinfo** except that the complete record of merges to a branch may not be contained solely in the **mergeinfo** on the branch root and you may have to look to any **sub-tree mergeinfo** to get a full accounting. Fortunately, Subversion does this for you and rarely will you need to look for it.

**Merging from Foreign Repositories**

Subversion supports merging from foreign repositories. While all merge source URLs must point to the same repository, the merge target (from the working copy) may come from a different repository than the source. However, copies made in the merge source will be transformed into plain additions in the merge target. Also, **merge-tracking** is not supported for merges from foreign repositories.

**Note:** When performing merges from repositories other than the one corresponding to the target item (from the working copy), the **Ignore ancestry / Disable merge tracking** option (on page 2784) in the **Merge Options** wizard page (on page 2783) will be selected automatically (and you cannot change this).

**General Merge Recommendations**

As a recommendation, you should only merge into clean working copies that **do not** contain any of the following:

• Modifications.
• Sparse directories (all directories must be of depth **infinity**).
• Switched items.
Important:
This recommendation becomes mandatory when performing a reintegrate merge (on page 2778) operation. Also, trying to merge to mixed-revision working copies will fail in all types of merge operations.

Remember:
The merge result is only in your local working copy and needs to be committed to the repository for it to be available to others.

Switch the Repository Location

The Switch action is useful when the repository location of a working copy, or an already committed item in the working copy, must be changed within the same repository. The action is available on the Tools menu when a versioned resource is selected in the current working copy that is displayed in the Working Copy view (on page 2813).

Note:
External items cannot be switched using this action. Instead, change the value of the svn:externals property set on the parent directory of the external item and update the parent directory.

Figure 698. Switch Dialog Box

The following options can be configured in the Switch dialog box:

Switch to URL
The new location in the same repository (on page 2848) you are switching to.

**Tip:**
You can switch to items that were deleted, moved, or replaced, by specifying the original URL (before the item was removed) and use a [peg revision (on page 2850)] at the end (for example, URL@rev1234).

**Note:**
For items that are already [switched (on page 2815)] that you want to switch back, the proposed URL is the original location of the item.

### Revision
The specific version of the location that you are switching to.

### Depth - (This option is applicable only for directories)

- **Current depth**
  Obeys the depths registered for the directories in the working copy that are to be switched.

- **Recursive (infinity)**
  Switches the location of the selected folder and all of its files and folders.

- **Immediate children (immediates)**
  Switches the location of the selected folder and its child files and folders without recursing subfolders.

- **File children only (files)**
  Switches the location of the selected folder and its child files.

- **This folder only (empty)**
  Switches the location of the selected folder (no child files or folders are included).

### Ignore "svn:externals" definitions
When selected, external items are ignored in the switch operation. This option is only available if you choose the **Current depth** or **Recursive (infinity)** depth.

### Change the working copy item to the specified depth
Changes the **sticky** depth on the directory in the working copy.

### Ignore ancestry
When not selected, the SVN system does not allow you to switch to a location that does not share a common ancestry with the current location. If selected, the SVN does not check for a common ancestry.
Relocate a Working Copy

Sometimes the base URL of the repository is changed after a working copy is checked out from that URL. For example, if the repository itself is moved to a different server. In such cases, you do not have to check out a working copy from the new repository location. It is easier to change the base URL of the root folder of the working copy to the new URL of the repository (on page 2848).

**Note:**

Peg revisions have no effect for this operation.

This Relocate action is available in the Tools menu when selecting the root item of the working copy.

**Note:**

External items that are defined using absolute URLs and that point to the same repository as the working copy are not affected by the Relocate action (the URL is not updated). To relocate these items, change the value of the `svn:externals` property for each external item (set on their parent directories). For example, if an external item is defined as `externalDir http://host/path/to/repo/to/dir` and the repository was moved to another server (`host2`) and its protocol changed to `https`, then the value of the property needs to be updated to `externalDir https://host2/path/to/repo/to/dir`.

**Tip:**

If you edit external items using the method described in the previous note, on the next update the system will try to fetch the external items again. To avoid this, you can invoke the Relocate action on each of these external items.

Patches

This section explains how to work with patches in Syncro SVN Client.

What is a Patch

Suppose you are working with a set of XML files that you want to tag the project and distribute releases to other team members. While working on the project and correcting problems, you may need to distribute the corrections to other team members. In this case, you can distribute a patch (a collection of differences) that would correct the problems when applied over the last distribution. By default, the Syncro SVN Client generates patches in the Unified Diff format, but it can also use the Git format (on page 2801).

Creating a patch in Apache Subversion™ implies the access to either two revisions of a versioned item, or two different versioned items from the repository:
Two revisions of a version item - the item can be local or remote and you can select two versions of it. This also applies when you need to generate a patch that only contains the changes in the working copy that were not yet committed.

Two different versioned items from the repository - the items are remote and you need to specify a revision for both.

Warning:
The resulting patch file may contain content that was written using a mix of encodings, based upon the encodings of the files that were compared. If you open the generated patch file in a text editor, it may result in unrecognizable content.

Generating a Patch - Local Items

Based on a versioned item (already committed or scheduled for addition) in the working copy, you can generate patches that contain the local changes, the differences between a specific revision of that item and the item itself, or differences between the pristine item and another item from the repository. There are four options for generating a patch based upon local items.

To open the Create patch wizard, use the Create patch action from the Tools menu or from the contextual menu in the Modified, Incoming, Outgoing, or Conflicts modes.

Figure 699. Create Patch Wizard - Local Items
Create Patch from Local Modifications

This is the most commonly used type of patch and contains only the local changes for the selected item.

This option is useful if you want to share changes with other team members and you are not yet ready to commit them. This option is only available for local items that contain modifications. It is not available for items that have a local file status of *unversioned* or *ignored*, and in some cases *missing* or *obstructed (on page 2815)*.

The steps are as follows:

1. Go to menu **Tools > Create patch**.
   This opens the **Create patch** wizard.
2. Select the **Create patch from local modifications** option in the dialog box.
3. Optionally, if you want to configure the options *(on page 2799)* for your patch, click the **Next** button.
   This options page does not remember your selections when creating future patches. It will revert to the default values.
   The **Options** wizard page is displayed.
4. Click the **Create patch** button.
   If the patch is applied on a folder of the working copy and that folder contains *unversioned files (on page 2815)*, this step of the wizard offers the option of selecting the ones that will be included in the patch.

![Create Patch Dialog Box - Add Unversioned Resources](image)

The patch is created and stored in the path specified in the **Output** section of the **Options page (on page 2799)** or in the default location.
Create Patch Against a Specific Revision

This type of patch contains changes between an old revision and the current content from the selected item within the working copy.

This option is useful if you want to obtain differences between an older revision and the current state of the working copy (for instance, to test how current changes apply to an older version).

The steps are as follows:

1. Go to menu Tools > Create patch.
   This opens the Create patch wizard.
2. Select the Create patch against a specific revision option in the dialog box.
3. Click the Next button.
   The second step of the wizard is opened:

   **Figure 701. Create Patch Wizard - Step 2**

   ![Create Patch Wizard - Step 2](image)

4. Select the revision to create patch against.
   You can select between the HEAD revision and a specific revision number. For the other revision option, you can click the History button *(on page 2741)* to display a list of the item revisions.
5. Optionally, if you want to configure the options (on page 2799) for your patch, click the Next button. This options page does not remember your selections when creating future patches. It will revert to the default values.

The Options wizard page is displayed.

6. Click the Create patch button.

The patch is created and stored in the path specified in the Output section of the Options page (on page 2799) or in the default location.

Create Patch Between Two Revisions of an Item

This type of patch contains historical changes between two revisions of a selected item.

This option is useful if you want to share changes between two revisions with other team members.

Tip:

If you need to generate a patch between two revisions of a previously deleted, moved, or replaced item, you should use the Create patch between two repository items option (on page 2794) instead.

The steps are as follows:

1. Go to menu Tools > Create patch.

   This opens the Create patch wizard.

2. Select the Create patch between two revisions of an item option in the dialog box.

3. Click the Next button.

   The second step of the wizard is opened:

   Figure 702. Create Patch Wizard - Step 2

4. Select the starting and ending revisions in the From and To sections.

   You can select between the HEAD revision and a specific revision number. For the other revision option, you can click the History button (on page 2741) to display a list of the item revisions.
Note:
The patch will only include changes between the two specified revisions, starting with the changes that were made after the older revision.

Tip:
If you want to reverse changes done between two revisions by using a patch file, you can specify the newer revision in the From section and the older version in the To section.

5. Optionally, if you want to configure the options (on page 2799) for your patch, click the Next button. This options page does not remember your selections when creating future patches. It will revert to the default values.

The Options wizard page is displayed.

6. Click the Create patch button.

The patch is created and stored in the path specified in the Output section of the Options page (on page 2799) or in the default location.

Create Patch Between Two Repository Items

This type of patch contains changes between one version of an item and a specific version of another item.

This option is useful for generating a patch that contains changes between existing, or even previously deleted, moved, or replaced items from different branches. This is the default option when you do not have a working copy loaded, when no repository items are selected, or when exactly two repository items of the same kind are selected.

Tip:
To access an item that was deleted, moved, or replaced, you need to specify the original URL (before the item was removed) and use a peg revision (on page 2850) at the end (for example, URL@rev1234).

The steps are as follows:

1. Go to menu Tools > Create patch.
   This opens the Create patch wizard.

2. Select the Create patch between two repository items option in the dialog box.

3. Click the Next button.
   The second step of the wizard is opened:
Figure 703. Create Patch Wizard - Step 2

4. Select the starting and ending URLs (on page 2848) and revisions in the From and To sections. You can select between the HEAD revision and a specific revision number. For the other revision option, you can click the History button (on page 2741) to display a list of the item revisions.

   **Important:**
   Both URLs must point to items from the same repository.

   **Note:**
   If you use a peg revision in the URL field, anything specified in the other revision field is ignored.

5. Optionally, if you want to configure the options (on page 2799) for your patch, click the Next button. This options page does not remember your selections when creating future patches. It will revert to the default values. The Options wizard page is displayed.

6. Click the Create patch button.

   The patch is created and stored in the path specified in the Output section of the Options page (on page 2799) or in the default location.
Generating a Patch - Remote Items

Based on a repository item, you can generate patches that contain the differences between two specific revisions of that item, or between a revision of that same item and another revision of another item from the repository. There are two options for generating a patch based upon remote items.

To open the Create patch wizard, use the Create patch action from the Tools menu.

Create Patch Between Two Revisions of an Item

This type of patch contains historical changes between two revisions of a selected item.

This option is useful if you want to share changes between two revisions with other team members.

Tip:
If you need to generate a patch between two revisions of a previously deleted, moved, or replaced item, you should use the Create patch between two repository items option (on page 2794) instead.

The steps are as follows:

1. Go to menu Tools > Create patch.
   This opens the Create patch wizard.
2. Select the Create patch between two revisions of an item option in the dialog box.
3. Click the Next button.
   The second step of the wizard is opened:
4. Select the starting and ending revisions in the **From** and **To** sections.
   You can select between the **HEAD revision** and a specific revision number. For the **other revision** option, you can click the **History** button (on page 2741) to display a list of the item revisions.

   **Note:**
   The patch will only include changes between the two specified revisions, starting with the changes that were made **after** the older revision.

   **Tip:**
   If you want to reverse changes done between two revisions by using a patch file, you can specify the newer revision in the **From** section and the older version in the **To** section.

5. Optionally, if you want to configure the options (on page 2799) for your patch, click the **Next** button.
   This options page does not remember your selections when creating future patches. It will revert to the default values.
   The **Options** wizard page is displayed.

6. Click the **Create patch** button.
   The patch is created and stored in the path specified in the **Output** section of the **Options** page (on page 2799) or in the default location.

**Create Patch Between Two Repository Items**

This type of patch contains changes between one version of an item and a specific version of another item.

This option is useful for generating a patch that contains changes between existing, or even previously deleted, moved, or replaced items from different branches. This is the default option when you do not have a working copy loaded, when no repository items are selected, or when exactly two repository items of the same kind are selected.
Tip:
To access an item that was deleted, moved, or replaced, you need to specify the original URL (before the item was removed) and use a peg revision (on page 2850) at the end (for example, URL@rev1234).

The steps are as follows:

1. Go to menu Tools > Create patch.
   This opens the Create patch wizard.
2. Select the Create patch between two repository items option in the dialog box.
3. Click the Next button.
   The second step of the wizard is opened: 

   **Figure 706. Create Patch Wizard - Step 2**

4. Select the starting and ending URLs (on page 2848) and revisions in the From and To sections.
   You can select between the HEAD revision and a specific revision number. For the other revision option, you can click the History button (on page 2741) to display a list of the item revisions.

Important:
Both URLs must point to items from the same repository.
5. Optionally, if you want to configure the options (on page 2799) for your patch, click the **Next** button. This options page does not remember your selections when creating future patches. It will revert to the default values. The **Options** wizard page is displayed.

6. Click the **Create patch** button.

The patch is created and stored in the path specified in the **Output** section of the **Options** page (on page 2799) or in the default location.

**Patch Options**

*Figure 707. Create Patch Wizard - Options*
Patch Section

**Depth - (This option is applicable only for directories)**

**Current depth**

The depth of recursing the folder for creating the patch is the same as the depth of that same folder in the working copy (available only when generating patches that contain changes from the working copy).

**Recursive (infinity)**

The patch is created on all the files and folders contained in the selected folder.

**Immediate children (immediates)**

The patch is created only on the child files and folders without recursing subfolders.

**File children only (files)**

The patch is created only on the child files.

**This folder only (empty)**

The patch is created only on the selected folder (no child file or folder is included in the patch).

**Ignore content of added files**

When selected, the patch file does not include the content of the *added* items. This option corresponds to the `--no-diff-added` option of the `svn diff` command.

**Ignore content of delete files**

When selected, the patch file does not include the content of the *deleted* items. This option corresponds to the `--no-diff-deleted` option of the `svn diff` command.

**Treat copied files as newly added**

When selected, copied items are treated as new, rather than comparing the items with their sources. This option corresponds to the `--show-copies-as-adds` option of the `svn diff` command.

**Include files having a binary MIME type**

When selected, the application is forced to compare items that are considered binary file types. This option corresponds to the `--force` option of the `svn diff` command.

**Ignore properties**

When selected, differences in the properties of items are ignored. This option corresponds to the `--ignore-properties` option of the `svn diff` command.

**Properties only**
When selected, only differences in the properties of the items are included in the patch file (file content is ignored). This option corresponds to the \texttt{--properties-only} option of the \texttt{svn diff} command.

\begin{quote}
\textbf{Note:}

The \textit{Ignore properties} and \textit{Properties only} options are mutually exclusive.
\end{quote}

\subsection*{Notice ancestry}

If selected, the SVN common ancestry is taken into consideration when calculating the differences. This option corresponds to the \texttt{--notice-ancestry} option of the \texttt{svn diff} command.

\subsection*{Files Comparison Section}

\subsubsection*{Ignore line endings}

If selected, the differences in line endings are ignored when the patch is generated. This option corresponds to the \texttt{--ignore-eol-style} option of the \texttt{svn diff} command.

\subsubsection*{Ignore whitespaces}

If selected, it allows you to specify how the whitespace changes should be handled. When selected, you can then choose between two options:

\begin{itemize}
  \item \textbf{Ignore whitespace changes} (\texttt{--ignore-space-change}) - Ignores changes in the amount of whitespaces or changes to their type (for example, when changing the indentation or changing tabs to spaces).
  \begin{quote}
  \textbf{Note:}
  Whitespaces that are added or removed are still considered to be changes.
  \end{quote}
  \item \textbf{Ignore all whitespaces} (\texttt{--ignore-all-space}) - Ignores all types of whitespace changes.
\end{itemize}

\subsection*{Output Section}

\subsubsection*{Save as}

The patch will be created and saved in the specified file.

\subsubsection*{Use Git extended diff format}

When selected, the patch is generated using the \textit{Git} format. This option corresponds to the \texttt{--git} option of the \texttt{svn diff} command.

\subsection*{Working with Repositories}

This section explains how to locate and browse SVN repositories in Syncro SVN Client.
Importing Resources Into a Repository

Importing resources into a repository is the process of copying local files and directories into a repository so that they can be managed by an Apache Subversion™ server. If you have already been using Subversion and you have an existing working copy you want to use, then you will likely want to follow the procedure for using an existing working copy (on page 2742).

The Import folder and Import Files actions are available from the Import submenu of the Repository menu or of the contextual menu in the Repositories view. These actions open a dialog box that allow you to select the directories or files that will be imported into the selected repository location.

The Import folder action opens the Import folder dialog box.

**Figure 708. Import Folder Dialog Box**

You can configure the following options:

**Folder**

Specify the local folder (on page 2848) by using the text box or the Browse button. This folder should not be empty or already under version control.

**Important:**

By default, the SVN system only imports the content of the specified folder, and not the folder itself. Therefore, it is recommended to use the Browse button to select the local folder so that the client will automatically append the name of it to the specified URL.

**URL**

Specify the repository location (on page 2848) that will be used to access the folder to be imported.

**Note:**

Peg revisions have no effect for this operation since it is used to send information to the repository.
Attention:
If the new location already exists, make sure that it is an empty directory to avoid mixing your project content with other files (if items exist with the same name, an error will occur and the operation will not proceed). Otherwise, if the address does not exist, it is created automatically.

Depth

Recursive (infinity)
Imports all the files and folders contained in the selected folder.

Immediate children (immediates)
Imports only the child files and folders without recursing subfolders.

File children only (files)
Imports only the child files.

This folder only (empty)
Imports only the selected folder (no child file or folder is included).

Share files matching global ignore patterns
When selected, the file names that match the patterns defined in either of the following locations are also imported into the repository:

- The global-ignores property in the SVN configuration file (on page 2848).
- The File name ignore patterns option (on page 284) in the SVN > Working Copy preferences page (on page 283).

Enable automatic properties/Disable automatic properties
Enables or disables automatic property assignment (per runtime configuration rules), overriding the enable-auto-props runtime configuration directive, defined in the SVN configuration file (on page 2848).

Note:
This option is available only when there are defined properties to be applied automatically for newly added items under version control. You can define these properties in the SVN config file (in the auto-props section). Based on the value of the enable-auto-props runtime configuration directive, the presented option is either Enable automatic properties, or Disable automatic properties.

Exporting Resources From a Repository
This is the process of taking a resource from the repository and saving it locally in a clean form, with no version control information. This is very useful when you need a clean build for an installation kit.
The Export dialog box is similar to the Check out dialog box:

**Figure 709. Export from Repository Dialog Box**

![Export from Repository Dialog Box](image)

You can configure the following options:

**URL**

Specify the source directory from the repository (on page 2848) by using the text box or the Browse button.

**Tip:**

To export an item that was deleted, moved, or replaced, you need to specify the original URL (before the item was removed) and use a peg revision (on page 2850) at the end (for example, URL@rev1234).

**Note:**

The content of the selected directory from the repository and not the directory itself will be exported to the file system.

**Revision**

You can choose between the HEAD or Other revision. If you need to export a specific revision, specify it in the Other text box or use the History button and choose a revision from the History dialog box.

**Export to**

Specify the location where you want to export (on page 2848) the repository directory by typing the local path in the text box or by using the Browse button. If the specified local path does not point to an existing directory, it will automatically be created.
Important:
By default, the SVN system only exports the content of the directory specified by the URL, and not the directory itself. Therefore, it is recommended to use the Browse button to select the export location so that the client will automatically append the name of the remote directory to the path of the selected directory.

Warning:
The destination directory should be empty. If files exist, they will be overwritten by exported files with matching names.

Depth

Recursive (infinity)
Exports all the files and folders contained in the selected folder.

Immediate children (immediates)
Exports only the child files and folders without recursing subfolders.

File children only (files)
Exports only the child files.

This folder only (empty)
Exports only the selected folder (no child file or folder is included).

Ignore "svn:externals" definitions
When selected, external items are ignored in the export operation. This option is only available if you choose the Recursive (infinity) depth.

EOL style
Defines the end-of-line (EOL) marker that should be used when exporting files that have the value or the svn:eol-style property set to native. You can choose between the following styles:

- Default - It uses the system-specific end-of-line marker.
- LF - The Unix / macOS-specific end-of-line marker (line feed).
- CR - The macOS 9 (or older)-specific end-of-line marker (carriage return).

Ignore keywords
When selected, the export operation does not expand the SVN keywords found inside the files.
Copy / Move / Delete Resources From a Repository

Once you have a location defined in the Repositories view (on page 2808), you can run commands (such as copy, move, and delete) directly on the repository. The commands correspond to the following actions in the contextual menu:

The Copy to and Move to action allows you to copy and move individual or multiple resources to a specific directory from the HEAD revision of the repository.

Figure 710. Copy/Move Items in Repository

The dialog box used to copy or move items allows you to browse the HEAD revision of the repository and select the destination of the items, presenting its repository URL below the tree view.

The Source section presents relevant options regarding the item(s) that you move or copy:

- **URL** - This field is displayed only if you copy/move a single item.
- **Revision** - Presents the revision that will have one or more items copied, allowing you to also choose another revision.
Note:
Since only items from the HEAD revision can be moved, the Revision options are not presented for the Move to action.

Note:
When you copy a single item while browsing a revision other than HEAD, the Revision options present this revision but does not allow you to change it. The same applies if copying multiple items.

- **New name** - This option is presented when you copy or move a single item, allowing you to also rename it.

Another useful action is **Delete**, allowing you to erase resources directly from the repository.

All three actions are commit operations and you will be prompted with the Commit message dialog box.

**Sparse Checkout**

Sometimes you only need to check out certain parts of a directory tree. In this case, you can check out the top directory (using the **Check out** action from the **Repositories** view (on page 2809)) and then recursively update only the needed directories (using the **Update** action from the **Working Copy** view (on page 2821)).

Each directory then has a depth set to it, with four possible values:

- **Recursive (infinity)** - Updates all descendant directories and files recursively.
- **Immediate children (immediates)** - Updates the directory, including direct child directories and files, but does not populate the child directories.
- **File children only (files)** - Updates the directory, including only child files without the child directories.
- **This folder only (empty)** - Updates only the selected directory, without updating any children.

For some operations, you can use as depth the current depth registered on the directories from the working copy (the value **Current depth**). This is the depth value defined in a previous check out or update operation.

The sparse checked out directories are presented in the **Working Copy** view (on page 2813) with a marker corresponding to each depth value, in the top left corner, as follows:

- **Recursive (infinity)** - This is the default value and it is has no mark. The directory has no limiting depth.
- **Immediate children (immediates)** - The directory is limited to direct child directories (without contents) and files.
- **File children only (files)** - The directory is limited to direct child files only.
- **This folder only (empty)** - The directory has empty depth set.
A depth set on a directory means that some operations process only items within the specified depth range. For example, **Synchronize** on a working copy directory reports the repository modified items within the depth set on the directory and those existing in the working copy outside of this depth.

The depth information is also presented in the **SVN Information** dialog box and in the tooltip displayed when hovering a directory in the **Working Copy** view.

### Syncro SVN Client Views

The main working area occupies the center of the application window, which contains the most important views:

- **Repositories View (on page 2808)**
- **Working Copy View (on page 2813)**
- **History View (on page 2828)**
- **Console View (on page 2843)**

The other views that support the main working area are also presented in this section.

### Repositories View

The **Repositories** view allows you to define and manage Apache Subversion™ repository locations and browse repositories. If the view is not displayed, it can be opened by selecting it from the **Window > Show View** menu.

If no connections to your repository are available, you can [add a new repository location (on page 2734)](on_page_2734). Repository files and folders are presented in a tree view with the repository locations at the first level, where each location represents a connection to a specific repository. More information about each resource is displayed in a tabular form:

- **Date** - Date when the resource was last modified.
- **Revision** - The revision number at the time the resource was last modified.
- **Author** - Name of the person who made the last modification on the resource.
- **Size** - Resource size on disk.
- **Lock information** - Information about the lock status of a file. When a repository file is locked by a user the **🔒** icon is displayed in this column. If no icon is displayed the file is not locked. The tooltip of this column displays the details about the lock:
  - **Owner** - The name of the user who created the lock.
  - **Date** - The date when the user locked the file.
  - **Expires on** - Date when the lock expires. Lock expiry policy is set in the repository options, on the server side.
  - **Comment** - The message attached when the file was locked.
- **Type** - Contains the resource type or file extension.
The **Repositories** view's toolbar contains the following buttons:

- **New Repository Location** - Allows you to enter a new repository location by means of the **Add SVN Repository** dialog box.
- **Move Up** - Move the selected repository up one position in the list of repositories in the **Repositories** view.
- **Move Down** - Move the selected repository down one position in the list of repositories in the **Repositories** view.
- **Collapse all** - Collapses all repository trees.
- **Stop** - Stops the current repository browsing operation executed when a repository node is expanded. This is useful when the operation takes too long or the server is not responding.
- **Settings** - Allows you to configure the resource table appearance.

**Repositories View Contextual Menu Actions**

The **Repositories** view contextual menu contains various actions, depending on the selected item. If a repository location is selected, the following management actions are available:

- **New Repository Location** (*Ctrl + Alt + N (Command + Alt + N on macOS)*)
Displays the Add SVN Repository dialog box. This dialog box allows you to define a new repository location.

Figure 712. Add SVN Repository Dialog Box

If the Validate repository connection option is selected, the URL connection is validated before being added to the Repositories view.

🔗 Edit Repository Location (Ctrl + Alt + E (Command + Alt + E on macOS))

Context-dependent action that allows you to edit the selected repository location using the Edit SVN Repository dialog box. It is active only when a repository location root is selected.

Change the Revision to Browse (Ctrl + Alt + B (Command + Alt + B on macOS))

Context-dependent action that allows you to change the selected repository revision using the Change the Revision to Browse dialog box. It is active only when a repository location root is selected.

✗ Remove Repository Location (Ctrl + Alt + R (Command + Alt + R on macOS))

Allows you to remove the selected repository location from the view. It shows you a confirmation dialog box before removal. It is active only when a repository location root is selected.

The following actions are common to all repository resources:

Open

Opens the selected file in the Editor view in read-only mode.

Open with

Displays the Open with dialog box to specify the editor where the selected file is opened. If multiple files are selected, only external applications can be used to open the files.

Save as

Saves the selected files locally, as they are in the browsed revision.

🔄 Refresh (F5)

Refreshes the resource selected in the Repositories view.

📊 Check out (Ctrl + Alt + O (Command + Alt + O on macOS))

Allows you to create a working copy from a repository directory, on your local file system. To read more about this operation, see Check out a working copy (on page 2739).

Branch/Tag
Oxygen XML Editor 24.1

| Tools | 2811 |

Allows you to create a branch or a tag from the selected folder in the repository. To read more about how to create a branch/tag, see the Creation and management of Branches/Tags (on page 2768) section.

**Share project**

Allows you to share a new project (on page 2737) using an SVN repository. The local project is automatically converted into an SVN working copy.

**Import:**

**Import folder (Ctrl + Shift + L (Command + Shift + L on macOS))**

Allows you to import the contents of a specified folder from the file system into the selected folder in a repository. To read more about this operation, see the section Importing resources into a repository (on page 2802).

![Note:](image)

The difference between the **Import folder** and **Share project** actions is that the latter also converts the selected directory into a working copy.

**Import Files (Ctrl + Shift + I (Command + Shift + I on macOS))**

Imports the files selected from the files system into the selected folder in the repository.

**Export**

Opens the Export dialog box (on page 2803) that allows you to configure options for exporting a folder from the repository to the local file system.

**Show History (Ctrl + H (Command + T on macOS))**

Displays the history of the selected resource. At the start of the operation, you can set filtering options.

**Show Annotation (Ctrl + Shift + A (Command + Shift + A on macOS))**

Opens the Show Annotation dialog box that computes the annotations for a file and displays them in the Annotations view (on page 2835), along with the history of the file in the History view.

**Revision Graph (Ctrl + G (Command + G on macOS))**

This action allows you to see the graphical representation of a resource history. For more details about a resource revision graph see Revision Graph (on page 2843). This operation is available for any resource selected in the Repositories view or Working Copy view.

**Copy URL Location (Ctrl + Alt + U (Command + Alt + U on macOS))**

Copies to clipboard the URL location of the selected resource.

**Copy to**
Copies to a specified location the currently selected resource(s). This action is also available when you browse other revisions than the latest one (HEAD), to allow restoring previous versions of an item.

**Move to (Ctrl + M (Command + M on macOS))**

Moves to a specified location the currently selected resource(s).

**Rename (F2)**

 Renames the selected resource.

**Delete (Delete)**

 Deletes selected items from the repository via an immediate commit.

**New Folder (Ctrl + Shift + F (Command + Shift + F on macOS))**

 Allows you to create a folder in the selected repository path (available only for folders).

**Locking**

The following options are available only for files:

- **Lock (Ctrl + K (Command + K on macOS))**
  
  Allows you to lock certain files that need exclusive access. For more details on the use of this action, see [Locking a file](on page 2750).

- **Unlock (Ctrl + Shift + K (Command + Shift + K on macOS))**
  
  Releases the exclusive access to a file from the repository. You can also choose to unlock it by force (break the lock).

- **Show SVN Properties (Ctrl + Shift + P (Command + Shift + P on macOS))**
  
  Brings up the Properties view (on page 2841) displaying the SVN properties for the selected resource. This view does not allow adding, editing, or removing SVN properties of a repository resource. These operations are allowed only for working copy resources.

- **Show SVN Information (Ctrl + I (Command + I on macOS))**
  
  Provides additional information for the selected resource. For more details, go to Obtain information for a resource (on page 2766).

**Assistant Actions**

When there is no repository configured, the Repositories view mode lists the following two actions:

**Figure 713. Repositories View Actions**

- Add a new repository
  
  Add a new repository to work on or to checkout a new working copy from it.

- Checkout a new working copy
  
  You can start using Syncro SVN Client by checking out a new working copy.
Drag and Drop Operations

The structure of the files tree can be changed with drag and drop operations inside the Repositories view. These operations behave in the same way with the Copy to/Move to operations.

Working Copy View

The Working Copy view allows you to manage the content of an SVN working copy. If the view is not displayed, it can be opened by selecting it from the Window > Show View menu.

The toolbar contains the following:

- The list of defined working copies.
- A set of view modes that allow you to filter the content of the working copy based on the resource status (such as incoming or outgoing changes).
- Settings menu.

If you click any of the view modes (All Files, Modified, Incoming, Outgoing, Conflicts), the information displayed changes as follows:

- All Files - Resources (files and folders) are presented in a hierarchical structure with the root of the tree representing the location of the working copy on the file system. Each resource has an icon representation that describes the type of resource and also depicts the state of that resource with a small overlay icon.

Figure 714. Working Copy View - All Files View Mode
- **Modified** - The resource tree presents resources modified locally (including those with conflicting content) and remotely. Decorator icons are used to differentiate between various resource states:
  - Incoming modification from repository:
    - File content or properties modified remotely.
    - New file added remotely.
    - File deleted remotely.
  - Outgoing modification to repository:
    - File content or properties modified locally.
    - New file added locally.
    - File deleted locally.
  - **Pseudo-conflict state** - A resource being locally and remotely modified at the same time, or a parent directory of such a resource.
  - **Real conflict state** - A resource that had both incoming and outgoing changes and not all the differences could be merged automatically through the update operation (manually editing the local file is necessary for resolving the conflict).

**Figure 715. Working Copy View - Modified View Mode**

- **Incoming** - The resource tree presents only incoming changes.
- **Outgoing** - The resource tree presents only outgoing changes.
- **Conflicts** - The resource tree presents only conflicting changes (real conflicts and pseudo-conflicts).
The following columns provide information about the resources:

- **Name** - Resource name. Resource icons can have the following decorator icons:
  - Additional status information:
    - ![Propagated modification marker](image) - A folder marked with this icon indicates that the folder itself presents some changes (such as modified properties) or a child resource has been modified.
    - ![External](image) - This indicates a mapping of a local directory to the URL of a versioned resource. It is declared with a `svn:externals` property in the parent folder and it indicates a working copy not directly related with the parent working copy that defines it.
    - ![Switched](image) - This indicates a resource that has been switched from the initial repository location to a new location within the same repository. The resource goes to this state as a result of the Switch action (on page 2787) executed from the contextual menu of the Working Copy view.
    - ![Grayed](image) - A resource with a grayed-out icon, but no overlaid icon, is an ignored resource. It is obtained with the Add to `svn:ignore` action.
  - Current SVN depth of a folder:
    - ![Immediate children (immediates)](image) (a variant of sparse checkout (on page 2807)) - The directory contains only direct file and folder children. Child folders ignore their content.
    - ![File children only (files)](image) (a variant of sparse checkout (on page 2807)) - The directory contains only direct file children, disregarding any child folders.
    - ![This folder only (empty)](image) (a variant of sparse checkout (on page 2807)) - The directory discards any child resource.

**Note:**
- Any folder not marked with one of the depth icons, has recursive depth (*infinity*) set by default (presents all levels of child resources).
- Although folders not under version control can have no depth set, Oxygen XML Editor presents *unversioned* and *ignored* folders with *empty* depth when Show unversioned directories content or Show ignored directories content options are not selected.

- **Local file status** - Shows the changes of working copy resources that were not committed to the repository yet. The following icons are used to mark resource status:
  - ![Resource is not under version control (unversioned)](image)
  - ![Resource is being ignored because it is not under version control and its name matches a file name pattern defined in one of the following places](image)
• **global-ignores** section in the SVN client-side config file (on page 2731).

**Attention:**
If you do not explicitly set the **global-ignores** runtime configuration option (either to your preferred set of patterns or to an empty string), Subversion uses the default value.

• **Application global ignores** option (on page 284) of Oxygen XML Editor.
  • The value of a svn:ignore property (on page 2745) set on the parent folder of the resource being ignored.

  ◦ - Marks a newly created resource, scheduled for addition to the version control system.
  ◦ - Marks a resource scheduled for addition, created by copying a resource already under version control and inheriting all its SVN history.
  ◦ - The content of the resource has been modified.
  ◦ - Resource has been replaced in your working copy (the file was scheduled for deletion, and then a new file with the same name was scheduled for addition in its place).
  ◦ - Resource is deleted (scheduled for deletion from Repository upon the next commit).
  ◦ - The resource is incomplete (as a result of an interrupted check out or update operation).
  ◦ - The resource is missing because it was moved or deleted without using an SVN-aware application.
  ◦ - The contents of the resource is in real conflict state (on page 2753).
  ◦ - Resource is in a name conflict state.
  ◦ - Resource is in tree conflict state after an update operation because:
    • Resource was locally modified and incoming deleted from repository.
    • Resource was locally scheduled for deletion and incoming modified.
  ◦ - Resource is obstructed (versioned as one kind of object: file, directory, or symbolic link, but has been replaced outside Syncro SVN Client by a different kind of object).

• **Local properties status** - Marks the resources that have SVN properties, with the following possible states:
  ◦ - The resource has SVN properties set.
  ◦ - The resource properties have been modified.
  ◦ - Properties for this resource are in real conflict (on page 2753) with property updates received from the repository.

• **Revision** - The current revision number of the resource.
• **Date** - Date when the resource was last time modified on the disk.
• **BASE Revision** - The revision number of the pristine version of the resource.
• **BASE Date** - Date when the pristine version of the resource was last time committed in the repository.
• **Author** - Name of the person who made the last modification on the pristine version of the resource.
• **Remote file status** - Shows changes of resources recently modified in the repository. The following icons are used to mark incoming resource status:
  ◦ - Resource is newly added in repository.
  ◦ - The content of the resource has been modified in repository.
- Resource was replaced in repository.
- Resource was deleted from repository.

• Remote properties status - Resources marked with the icon have incoming modified properties from the repository.
• Remote revision - Revision number of the resource latest committed modification.
• Remote date - Date of the resource latest modification committed on the repository.
• Remote author - Name of the author who committed the latest modification on the repository.

• Lock information - Shows the lock state of a resource. The lock mechanism is a convention intended to help you signal other users that you are working with a particular set of files. It minimizes the time and effort wasted in solving possible conflicts generated by clashing commits. A lock gives you exclusive rights over a file, only if other users follow this convention and they do not try to bypass the lock state of a file.

A folder can be locked only by the SVN client application, completely transparent to the user, if an operation in progress was interrupted unexpectedly. As a result, folders affected by the operation are marked with the symbol. To clear the locked state of a folder, use the action.

Note: Users can lock only files.

The following lock states are displayed:

- no lock - the file is not locked. This is the default state of a file in the SVN repository.

- remotely locked ( ) - shown when:
  • Another user has locked the file in the repository.
  • The file was locked by the same user from another working copy.
  • The file was locked from the view.

If you try to commit a new revision of the file to the repository, the server does not allow you to bypass the file lock.

Note: To commit a new revision, you need to wait for the file to be unlocked. Ultimately, you might try to break or steal the lock, but this is not what other users expect. Use these actions carefully, especially when you are not the file lock owner.

- locked ( ) - displayed after you have locked a file from the current working copy. Now you have exclusive rights over the corresponding file, being the only one who can commit changes to the file in the repository.
**Note:**
Working copies keep track of their locked files, so the locks are presented between different sessions of the application. Synchronize your working copy with the repository to make sure that the locks are still valid (not stolen or broken).

- **stolen** (🔒) - a file already locked from your working copy is being locked by another user. Now the owner of the file lock is the user who stole the lock from you.
- **broken** (❌) - a file already locked from your working copy is no longer locked in the repository (it was unlocked by another user).

**Note:**
To remove the stolen or broken states from your working copy files, you have to Update them.

If one of your working copy files is locked, hover the mouse pointer over the lock icon to see more information:

- Lock type - current file lock state
- Owner - the name of the user who created the lock
- Date - the date when the user locked the file
- Expires on - date when the lock expires. Lock expiry policy is set in the repository options, on the server side
- Comment - the message attached when the file was locked

- **Size** - Resource size on disk
- **Type** - Contains the resource type or file extension

**Note:**
The working copy table allows you to show or hide any of its columns and also to sort its contents by any of the displayed columns. The table header provides a contextual menu that allows you to customize the displayed information.

The toolbar contains the following options for switching to a different working copy:

- **List of Defined Working Copies** - A drop-down menu that contains all the working copies Oxygen XML Editor is aware of. When you select a different working copy from the list, the newly selected working copy content is scanned and displayed in the Working Copy view.
- **Working Copies Manager** (.macOS) - Opens a dialog box that displays the working copies Oxygen XML Editor is aware of. In this dialog box, you can add existing working copies or remove those
you no longer need. If you try to add a folder that is not a valid Subversion working copy, Oxygen XML Editor warns you that the selected directory is not under version control.

**Note:**
Removing a working copy from this dialog box does NOT remove it from your file system; you will have to do that manually.

### Working Copy Settings

The **Settings** button from the toolbar of the **Working Copy** view provides the following options:

- **Show unversioned directories content** - Displays the content of unversioned directories.

**Note:** If this option is not selected, it will be ignored for items that, after a synchronize, are reported as incoming from the repository. This applies for all working copy modes, except **All Files**.

- **Show ignored items** - Displays the ignored resource when **All Files** mode is selected.
- **Show ignored directories content** - Displays the content of ignored directories when **All Files** mode is selected.

**Note:** Although ignored items are not presented in the **Modified**, **Incoming**, and **Conflicts** modes, they will be if, after a synchronize, they are reported as incoming from the repository.

- **Show deleted items** - Displays the deleted resource when **All Files** mode is selected. All other modes always display deleted resources, disregarding this option.

- **Tree / Compressed / Flat** - Affect the way information is displayed inside the **Modified**, **Incoming**, **Outgoing**, and **Conflicts** view modes.

- **Configure columns** - Allows you to customize the structure of the **Working Copy** view data. This action opens the following dialog box:
Figure 716. Configure Columns of Working Copy View

The order of the columns can be changed with the two arrow buttons. The column size can be edited in the **Width of selected column** field. The **Restore Defaults** button reverts all columns to the default order, width, and enabled/disabled state from the installation of the application.

**Working Copy Format**

When an SVN working copy is loaded, Syncro SVN Client first checks the format of the working copy:

- If the format is older than SVN 1.7, you are prompted to upgrade it to SVN 1.8 to load it.
- If the format is 1.7, Syncro SVN Client takes into account the state of the **When loading an old format working copy** option (on page 283).

To change how working copy formats are handled, open the **Preferences** dialog box (Options > Preferences) (on page 124), go to **SVN > Working copy**, and configure the options in the **Administrative area** (on page 283) section.

**Note:**

- The format of the working copy can be downgraded or upgraded at any time with the **Upgrade** and **Downgrade** actions available in the **Tools** menu. These actions allow switching between SVN 1.7 and SVN 1.8 working copy formats.
- SVN 1.7 working copies cannot be downgraded to older formats.
Refresh a Working Copy

A refresh is a frequent operation triggered automatically when you switch between two working copies using the toolbar selector of the Working Copy view and when you switch between Oxygen XML Editor and other applications.

The Working Copy view features a fast refresh mechanism: the content is cached locally when loading the working copy for the first time. Later on, when the same working copy is displayed again, the application uses this cache to detect the changes between the cached content and the current content found on disk. The refresh operation is run on these changes only, thus improving the response time. Improvement is noticeable especially when working with large working copies.

Working Copy View Contextual Menu Actions

The contextual menu in the Working Copy view contains the following actions:

- **Edit conflict (Ctrl + E (Command + E on macOS))**
  
  Opens the Compare editor, allowing you to modify the content of the currently conflicting resources. For more information about editing conflicts, see Edit conflicts (on page 2755).

- **Open in Compare Editor (Ctrl + Alt + C (Command + Alt + C on macOS))**
  
  Displays changes made in the currently selected file.

- **Open (Ctrl + O (Command + O on macOS))**
  
  Opens the selected resource from the working copy. Files are opened with an internal editor or an external application associated with that file type, while folders are opened with the default file system browsing application (Windows Explorer on Windows, Finder on macOS, etc).

- **Open with...**
  
  Submenu that allows you to open the selected resource either with Oxygen XML Editor or with another application.

- **Show in Explorer/Show in Finder**
  
  Opens the parent directory of the selected working copy file and selects the file.

- **Expand All (Ctrl + Alt + X (Command + Alt + X on macOS))**
  
  Displays all descendants of the selected folder. The same behavior is obtained by double-clicking a collapsed folder.

- **Refresh (F5)**
  
  Re-scans the selected resources recursively and refreshes their status in the working copy view.

- **Synchronize (Ctrl + Shift + S (Command + Shift + S on macOS))**
  
  Connects to the repository and determines the working copy and repository changes made to the selected resources. The application switches to Modified view mode if the Always switch to 'Modified' mode option (on page 283) is selected.
**Update (Ctrl + U (Command + U on macOS))**

Updates the selected resources to the HEAD revision (latest modifications) from the repository. If the selection contains a directory, it will be updated depending on its depth.

**Update to revision/depth**

Allows you to update the selected resources from the working copy to an earlier revision from the repository. You can also select the update depth for the current folder. You can find out more about the depth term in the sparse checkouts (on page 2807) section.

**Commit**

Collects the outgoing changes from the selected resources in the working copy and allows you to choose exactly what resources to commit. A directory will always be committed recursively. Unversioned resources will be deselected by default. In the Commit dialog box you can also enter a comment before sending your changes to the repository.

**Revert (Ctrl + Shift + V (Command + Shift + V on macOS))**

Undoes all local changes for the selected resources. It does not contact the repository and the files are obtained from the Apache Subversion™ pristine copy. It is available only for modified resources. See Revert your changes (on page 2757) for more information.

**Override and Update**

Drops any outgoing change and replaces the local resource with the HEAD revision. This action is available on resources with outgoing changes, including conflicting ones. See the Revert your changes (on page 2757) section.

**Override and Commit**

Drops any incoming changes and sends your local version of the resource to the repository. This action is available on conflicting resources. For more information see Drop incoming modifications (on page 2759).

**Mark Resolved (Ctrl + Shift + R (Command + Shift + R on macOS))**

Instructs the Subversion system that you resolved a conflicting resource. For more information, see Merge conflicts (on page 2758).

**Mark as Merged (Ctrl + Shift + M (Command + Shift + M on macOS))**

Instructs the Subversion system that you resolved the pseudo-conflict by merging the changes and you want to commit the resource. Read the Merge conflicts (on page 2758) section for more information about how you can solve the pseudo-conflicts.

**Create patch (Ctrl + Alt + P (Command + Alt + P on macOS))**

Allows you to create a file containing all the differences between two resources, based on the `svn diff` command. To read more about creating patches, see the section about patches (on page 2792).

**Compare with:**
• Latest from HEAD (Ctrl + Alt + H (Command + Alt + H on macOS)) - Performs a 3-way diff operation between the selected file and the HEAD revision from the repository and displays the result in the Compare view. The common ancestor of the 3-way diff operation is the BASE version of the file from the local working copy.

• BASE revision (Ctrl + Alt + C (Command + Alt + C on macOS)) - Compares the working copy file with the BASE revision file (the so-called pristine copy).

• Revision (Ctrl + Alt + R (Command + Alt + R on macOS)) - Displays the History view that contains the log history of that resource.

• Branch/Tag - Opens the Compare with Branch/Tag dialog box that allows you to specify another file from the repository (on page 2848) (To URL field) to compare with the working copy file. You can specify the revision of the repository file by choosing between HEAD revision or specific Other revision.

Tip:
To compare with a file that was deleted, moved, or replaced, you need to specify the original URL (before the file was removed) and use a peg revision (on page 2850) at the end (for example, URL@rev1234).

• Each other - Compares two selected files with each other.

These compare actions are available only if the selected resource is a file.

Replace with:

• Latest from HEAD - Replaces the selected resources with their versions from the HEAD revision of the repository.

• BASE revision - Replace the selected resources with their versions from the pristine copy (the BASE revision).

Note:
In some cases it is impossible to replace the currently selected resources with their versions from the BASE/HEAD revision:

• For the Replace with BASE revision action, the resources being unversioned or added have no BASE revision, and thus cannot be replaced. However, they will be deleted if the action is invoked on a parent folder. The action will never work for missing folders or for obstructing files (folders being obstructed by a file), since you cannot recover a tree of folders.

• For the Replace with latest from HEAD action, you must be aware that there are cases when resources will be completely deleted or reverted to the BASE revision and then updated to a HEAD revision to avoid conflicts. These cases are:
The resource is unversioned, added, obstructed, or modified.

The resource is affected by a \texttt{svn:ignore} or \texttt{svn:externals} property that is locally added on the parent folder and not yet committed to the repository.

**Show History (Ctrl + H (Command + H on macOS))**

Displays the History view where the log history for the selected resource will be presented. For more details about resource history, see the sections about the resource history view (on page 2828) and requesting the history for a resource (on page 2767).

**Show Annotation (Ctrl + Shift + A (Command + Shift + A on macOS))**

Opens the Show Annotation dialog box that computes the annotations for a file and displays them in the Annotations view (on page 2835), along with the history of the file in the History view.

**Revision Graph (Ctrl + G (Command + G on macOS))**

This action allows you to see the graphical representation history of a resource. For more details about the revision graph of resources, see Revision Graph (on page 2843).

**Copy URL Location (Ctrl + Alt + U (Command + Alt + U on macOS))**

Copies the encoded URL of the selected resource from the Working Copy to the clipboard.

**Mark as copied**

You can use this action to mark an item from the working copy as a copy of another item under version control, when the copy operation was performed outside of an SVN client. The Mark as copied action is available when you select two items (both the new item and source item), and it depends on the state of the source item.

**Mark as moved**

You can use this action to mark an item from the working copy as being moved from another location of the working copy, when the move operation was performed outside of an SVN client. The Mark as moved action is available when you select two items from different locations (both the new item and the source item that is usually reported as missing), and it depends on the state of the source item.

**Mark as renamed**

You can use this action to mark an item from the working copy as being renamed outside of an SVN client. The Mark as renamed action is available when you select two items from the same directory (both the new item and the source item that is usually reported as missing), and it depends on the state of the source item.

**Copy to**

Copies the currently selected resource to a specified location.

**Move to (Ctrl + M (Command + M on macOS))**
Moves the currently selected resource to a specified location.

**Rename (F2)**

As with the move command, a copy of the original resource will be made with the new name and the original will be marked as deleted. Note that you can only rename one resource at a time.

**Delete (Delete)**

Schedules selected items for deletion upon the next commit and removes them from the disk. Depending on the state of each item, you are prompted to confirm the operation.

**New:**

- **New File** - Creates a new file inside the selected folder. The newly created file will be added under version control only if the parent folder is already versioned.
- **New Folder (Ctrl + Shift + F (Command + Shift + F on macOS))** - Creates a child folder inside the selected folder. The newly created folder will be added under version control only if its parent is already versioned.
- **New External Folder (Ctrl + Shift + W (Command + Shift + W on macOS))** - This operation allows you to add a new external definition on the selected folder. An external definition is a mapping of a local directory to a [URL of a versioned directory](on page 2848), and ideally a particular revision, stored in the `svn:externals` property of the selected folder.

**Tip:**

You can specify a particular revision of the external item by using a [peg revision](on page 2850) at the end of the URL (for example, URL@rev1234). You can also use peg revisions to access external items that were deleted, moved, or replaced.

The URL used in the external definition format can be relative. You can specify the repository URL that the external folder points to by using one of the following relative formats:

- `../` - Relative to the URL of the directory that the `svn:externals` property is set.
- `^/` - Relative to the root of the repository where the `svn:externals` property is versioned.
- `//` - Relative to the scheme of the URL of the directory that the `svn:externals` property is set.
- `/` - Relative to the root URL of the server that has the `svn:externals` property versioned.

**Important:**

To change the target URL of an external definition, or to delete an external item, do the following:
1. Modify or delete the item definition found in the `svn:externals` property that is set on the parent folder.

2. For the change to take effect, use the **Update** operation on the parent folder of the external item.

**Note:**
Syncro SVN Client does not support definitions of local relative external items.

**Add to "svn:ignore" (Ctrl + Alt + I (Command + Alt + I on macOS))**

Allows you to add files that should not participate in the **version control** operations inside your working copy. This action can only be performed on resources not under **version control**. It actually modifies the value of the `svn:ignore` property in the parent directory of the resource. Read more about this in the **Ignore Resources Not Under Version Control** (on page 2745) section.

**Add to version control (Ctrl + Alt + V (Command + Alt + V on macOS))**

Allows you to add resources that are not under **version control**. For further details, see Add Resources to Version Control (on page 2743) section.

**Remove from version control**

Schedules the selected items for deletion from the repository upon the next commit. The items are not removed from the file system after committing.

**Clean up (Ctrl + Shift + C (Command + Shift + Cd on macOS))**

Performs a maintenance cleanup operation on the selected resources from the working copy. This operation removes the Subversion maintenance locks that were left behind. This is useful when you already know where the problem originated and want to fix it as quickly as possible. It is only active for resources under **version control**.

**Locking:**

- **Scan for locks (Ctrl + L (Command + L on macOS))** - Contacts the repository and recursively obtains the list of locks for the selected resources. A dialog box containing the locked files and the lock description will be displayed. This is only active for resources under **version control**. For more details see **Scanning for locks** (on page 2749).

- **Lock (Ctrl + K (Command + K on macOS))** - Allows you to lock certain files that need exclusive access. You can write a comment describing the reason for the lock and you can also force (steal) the lock. This action is active only on files under **version control**. For more details on the use of this action see **Locking a file** (on page 2750).

- **Unlock (Ctrl + Alt + K (Command + Alt + K on macOS))** - Releases the exclusive access to a file from the repository. You can also choose to unlock it by force (break the lock).
Show SVN Properties (Ctrl + P (Command + P on macOS))

Brings up the Properties view (on page 2841) and displays the SVN properties for the selected resource.

Show SVN Information (Ctrl + I (Command + I on macOS))

Provides additional information for the selected resource from the working copy. For more details, go to Obtain information for a resource (on page 2766).

Drag and Drop Operations

The structure of the files tree can be changed with drag and drop operations inside the Working Copy view. These operations behave in the same way with the Copy to/Move to operations.

Also, files and folders can be added to the file tree of the view as unversioned resources by drag and drop operations from other applications (for example, from Windows Explorer or macOS Finder). In this case, the items from the file system are only copied, without removing them from their original location.

Attention:

When you drag items from the working copy to a different application, the performed operation is controlled by that application. This means that the moved items are left as missing in the working copy (items are moved in the file system only, but no SVN versioning meta-data is changed).

Assistant Actions

To ensure a continuous and productive work flow, when a view mode has no files to present, it offers a set of guiding actions with some possible paths to follow.

Initially, when there is no working copy configured the All Files view mode lists the following two actions:

Figure 717. All Files Panel

For Modified, Incoming, Outgoing, Conflicts view modes, the following actions may be available, depending on the current working copy state in various contexts:

- Synchronize with Repository - Available only when there is nothing to present in the Modified and Incoming view modes.
- Switch to Incoming - Selects the Incoming view mode.
- Switch to Outgoing - Selects the Outgoing view mode.
- **Switch to Conflicts** - Selects the Conflicts view mode.

- **Show all changes/incoming/outgoing/conflicts** - Depending on the currently selected view mode, this action presents the corresponding resources after a synchronize operation was executed only on a part of the working copy resources.

- **(Information message)** - Informs you why there are no resources presented in the currently selected view mode.

**History View**

In Apache Subversion™, both files and directories are versioned and have a history. If you want to examine the history for a selected resource and find out what happened at a certain revision you can use the History view that can be accessed from Repositories view (on page 2808), Working Copy view (on page 2813), Revision Graph (on page 2843), or Directory Change Set view (on page 2833). From the Working copy view you can display the history of local versioned resources. If the view is not displayed, it can be opened by selecting it from the Window > Show View menu.

The view consists of four distinct areas:

- The table showing details about each revision, such as revision number, commit date and time, number of changes (more details available in the tooltip), author's name, and a fragment of the commit message.

Some revisions may be highlighted to emphasize:

- The current revision of the resource that has the history displayed - a bold font revision.
- The last revision where the content or properties of the resource were modified - blue font revision.

**Note:**
Both font highlights may be applied for the same revision.

- The complete commit message for the selected revision.
- A tree structure showing the folders where the modified resources are located. You can compress this structure to a more compact form that focuses on the folders that contain the actual modifications.
- The list of resources modified in the selected revision. For each resource, the type of action done against it is marked with one of the following symbols:
  - ☥ - A newly created resource.
  - ☥ - A newly created resource, copied from another repository location.
  - ☥ - The content/properties of the resource were modified.
  - ❌ - Resource was replaced in the repository.
  - ❌ - Resource was deleted from the repository.
You can group revisions in predefined time frames (today, yesterday, this week, this month), by pressing the **Group by date** button from the toolbar.

### History Filter Dialog Box

The **History view** does not always show all the changes ever made to a resource because there may be thousands of changes and retrieving the entire list can take a long time. Normally you are interested in the more recent ones. That is why you can specify the criteria for the revisions displayed in the **History view** by selecting one of several options presented in the **History** dialog box that is displayed when you invoke the **Show History** action.
Figure 719. History Filters Dialog Box

Options for the set of revisions presented in the History view are:

- All revisions of the selected resource.
- Only revisions between a start revision number and an end revision number.
- Only revisions added in a period of time (such as today, last week, last month, etc.)
- Only revisions between a start and an end date.
- Only revisions committed by a specified SVN user.

The toolbar of the History view has two buttons for extending the set of revisions presented in the view: Get next 50 and Get all.

History Filter Field

When only the history entries that contain a specified substring need to be displayed in the History view, the filter field displayed at the top of this view is a useful tool. Just enter the search string in the field next to the Find label. Only the items (with an author name, commit message, revision number, or date) that match the search string are kept in the History view. When you click the Search button, the filter action is executed and the content of the table is updated.

History View Contextual Menu Actions

The History view contains the following contextual menu actions:

- Compare with working copy
  
  Compares the selected revision with your working copy file. It is available only when you select a file.

- Open
Opens the selected revision of the file into the Editor. This is available only for files.

Open with

Displays the Open with dialog box to specify the editor where the selected file will be opened.

Get Contents

Replaces the current version from the working copy with the contents of the selected revision from the history of the file. The BASE version of the file is not changed in the working copy so that after this action the file will appear as modified in a synchronization operation, that is newer than the BASE version, even if the contents is from an older version from history.

Save as

Allows you to save the contents of a file as it was committed at a certain revision. This option is available only when you access the history of a file.

Copy to

Copies to the repository the item whose history is displayed, using the selected revision. This option is active only when presenting the history for a repository item (URL).

Note:

This action can be used to resurrect deleted items also.

Revert changes from this revision

Reverts changes that were made in the selected revisions. The are reverted only in the working copy and do not affect the repository items. It does not replace your working copy items with those from the selected revisions. This action is available when the resource history was launched for a local working copy resource.

Note:

For items displayed in the Affected Paths section that were added, deleted, or replaced, this action has no effect because such changes are considered to be changes to the parent directory. To revert these types of changes, follow these steps:

1. Request the history for the parent directory.
2. Identify the revision that contains the changes you want to revert.
3. Invoke the action on that revision.

Warning:

There are instances where the SVN Client is not able to identify the corresponding working copy item for the selected item in the Affected Paths section. In this case, the action does not proceed and an error message is displayed. For example, the selected
item in the **Affected Paths** section is from a different repository location than the working copy item that has the history displayed.

**Update to revision**

Updates your working copy resource to the selected revision. This is useful if you want your working copy to reflect a time in the past. It is best to update a whole directory in your working copy, not just one file. Otherwise, your working copy is inconsistent and you are unable to commit your changes.

**Check out**

Checks out a new working copy of the directory that has the history presented, from the selected revision.

**Export**

Opens the Export dialog box (on page 2803) that allows you to configure options for exporting a folder from the repository to the local file system.

**Show Annotation (Ctrl + Shift + A (Command + Shift + A on macOS))**

Opens the Show Annotation dialog box that computes the annotations for a file and displays them in the Annotations view (on page 2835), along with the history of the file in the History view.

**Change**

Allows you to change commit data for a file:

- *Author* - Changes the name of the SVN user that committed the selected revision.
- *Message* - Changes the commit message of the selected revision.

When two resources are selected in the **History** view, the contextual menu contains the following actions:

**Compare revisions**

When the resource is a file, the action compares the two selected revisions using the **Compare** view. When the resource is a folder, the action displays the set of all resources from that folder that were changed between the two revision numbers.

**Revert changes from these revisions**

Similar to the `svn merge` Command, it merges two selected revisions into the working copy resource. This action is only available when the resource history was requested for a working copy item.

For more information about the **History view** and its features, see the Request history for a resource (on page 2767) and Using the resource history view (on page 2828) sections.
Directory Change Set View

The result of comparing two reference revisions from the history of a folder resource is a set with all the resources changed between the two revision numbers. The changed resources can be contained in the folder or in a subfolder of that folder. These resources are presented in a tree format. For each changed resource all the revisions committed between the two reference revision numbers are presented.

**Figure 720. Directory Change Set View**

The set of changed resources displayed in the tree is obtained by running the action **Compare revisions** available on the contextual menu of the **History** view when two revisions of a folder resource are selected in the **History** view.

The left side panel of the view contains the tree hierarchy with the names of all the changed resources between the two reference revision numbers. The right side panel presents the list with all the revisions of the resource selected in the left side tree. These revisions were committed between the two reference revision numbers. Selecting one revision in the list displays the commit message of that revision in the bottom area of the right side panel.

Double-clicking a file listed in the left-side tree performs a diff operation between the two revisions of the file corresponding to the two reference revisions. Double-clicking one of the revisions displayed in the right-side list of the view performs a diff operation between that revision and the previous one for the same file.

The contextual menu of the right side list contains the following actions:

**Compare with previous version**

Performs a diff operation between the selected revision in the list and the previous one.

**Open**

Opens the selected revision in the associated editor type.

**Open with**
Displays a dialog box with the available editor types and allows you to select the editor type for opening the selected revision.

**Save as**

Saves the selected file as it was in the selected revision.

**Copy to**

Copies to the repository the item whose history is displayed, using the selected revision.

*Note:*

This action can be used to resurrect deleted items also.

**Check out**

Checks out a new working copy of the selected directory, from the selected revision.

**Export**

Opens the Export dialog box (on page 2803) that allows you to configure options for exporting a folder from the repository to the local file system.

**Show Annotation (Ctrl + Shift + A (Command + Shift + A on macOS))**

Opens the Show Annotation dialog box that computes the annotations for a file and displays them in the Annotations view (on page 2835), along with the history of the file in the History view.

**Show SVN Information (Ctrl + I (Command + I on macOS))**

Provides additional information for a selected resource. For more details, go to Obtain information for a resource (on page 2766).

**Editor Panel of SVN Client**

You can open a file for editing in an internal built-in editor. There are default associations between frequently used file types and the internal editors in the File Types preferences panel (on page 297).

The internal editor can be accessed either from the Working copy view (on page 2813) or from the History view (on page 2828). No actions that modify the content are allowed when the editor is opened with a revision from history.

Only one file at a time can be edited in an internal editor. If you try to open another file it will be opened in the same editor window. The editor provides syntax highlighting for known file types. This means that a different color will be used for each recognized token type found in the file. If the file's content type is unknown you will be prompted to choose the proper way the file should be opened.

After editing the content of the file in an internal editor you can save it to disk by using the Save action from the File (on page 2719) menu or the Ctrl + S (Command + S on macOS) key shortcut. After saving your file you can see the file changed status in the Working Copy view (on page 2813).
If the internal editor associated with a file type is not the XML Editor, then the encoding set in the preferences for Encoding for non XML files (on page 168) is used for opening and saving a file of that type. This is necessary because in the case of XML files, the encoding is usually declared at the beginning of the XML file in a special declaration or it assumes the default value UTF-8, but in the case of non-XML files, there is no standard mechanism for declaring the encoding for the file.

**Annotations View**

Sometimes you need to know not only what was changed in a file, but also who made those changes. The Annotations view displays the revision and the author that changed every line in a file. The annotations of a file are computed and this view is opened with the Show Annotation action, which is available in the History menu, and from the contextual menu of the following views: Repositories view (on page 2808), Working copy view (on page 2813), History view (on page 2828), and Directory Change Set view (on page 2833).

This action opens a dialog box that allows you to configure some options for showing the annotations.

**Figure 721. Show Annotation Options Dialog Box**

![Show Annotation Options Dialog Box](image)

Once you have configured the options and click OK, the Annotations view is displayed (by default, on the right side of the application). You can click a line in the editor panel where the file is opened to see the revision where the line was last modified. The same revision is highlighted in the History view and you can also see all the lines that were changed in the same revision highlighted in the editor panel. Also, the entries of the Annotations view corresponding to that revision are highlighted. Therefore, the Annotations view,
History view, and annotations editor panel are all synchronized. Clicking a line in one of them highlights the corresponding lines in the other two.

Figure 722. Annotations View

The following options can be configured in the Show Annotation dialog box:

From Revision Section

Select the revision to have the annotation computed. If you click the History button, the History dialog box (on page 2741) is displayed, which allows you to select a revision.

To Revision Section

Select the ending revision by choosing between the HEAD revision or specify it in the Other text box. If you click the History button, the History dialog box (on page 2741) is displayed, which allows you to select a revision.

Encoding

Select the encoding to be used when the annotation is computed. For each line of text, the SVN Client looks through the history of the file to be annotated see when it was last modified, and by whom. It is required that it is in the form of a text file. Therefore, encoding is needed to properly decode and read the file content. By default, the encoding of the operating system is used.

Ignore MIME type

If selected, the file is treated as a text file and ignores what the SVN system infers from the svn:mime-type property.

Ignore line endings

If selected, the differences in line endings are ignored when the annotation is computed.

Ignore whitespaces
If selected, it allows you to specify how the whitespace changes should be handled. When selected, you can then choose between two options:

- **Ignore whitespace changes** - Ignores changes in the amount of whitespaces or to their type (for example, when changing the indentation or changing tabs to spaces).

  Note:
  Whitespaces that were added where there were none before, or that were removed, are still considered to be changes.

- **Ignore all whitespaces** - Ignores all types of whitespace changes.

  Tip:
  Selecting any of these *ignore* options can help you better determine the last time a meaningful change was made to a given line of text.

After you configure the options and click **OK**, the annotations will be computed and the **Annotations** view is displayed, where all the users that modified the selected resource will be presented, along with the specific lines and revision numbers modified by each user.

Note:
If the file has a very long history, the computation of the annotation data can take a long time to process.

**Compare View**

In the Oxygen XML Editor, there are three types of files that can be checked for differences: text files, image files and binary files. For the text files and image files you can use the built-in **Compare** view. This view is automatically opened if you select two files and use the **Compare with > Each Other** action in the contextual menu.
At the top of each of the two editors, there are presented the name of the open file, the corresponding SVN revision number (for remote resources) and the author who committed the associated revision.

When comparing text, the differences are computed using a **line differencing algorithm**. The view can be used to show the differences between two files in the following cases:

- After obtaining the outgoing status of a file with a **Refresh** operation, the view can be used to show the differences between your working file and the pristine copy. In this way you can find out what changes you will be committing.
- After obtaining the incoming and outgoing status of the file with the **Synchronize** operation, you can examine the exact differences between your local file and the **HEAD** revision file.
- You can use the **Compare view** from the **History view** to compare the local file and a selected revision or compare two revisions of the same file.

The Compare view contains two editors. Edits are allowed only in the left editor and only when it contains the working copy file. To learn more about how the view can be used, see **View Differences** (on page 2751).

**Compare View Toolbar**

The toolbar of the **Compare** view contains the operations that can be performed on the source and target files.
The following actions are available:

**Algorithm**

The algorithm to be used for performing a comparison. The following options are available:

- **Auto** - Selects the most appropriate algorithm, based on the compared content and its size (selected by default).
- **Lines** - Computes the differences at line level, meaning that it compares two files or fragments looking for identical lines of text. This algorithm is not available when the file comparison is in Author comparison mode.
- **XML Fast** - Comparison that works well on large files or fragments, but it is less precise than XML Accurate.
- **XML Accurate** - Comparison that is more precise than XML Fast, at the expense of speed. It compares two XML files or fragments looking for identical XML nodes.

**Save action**

Saves the content of the left editor when it can be edited.

**Perform Files Differencing**

Looks for differences between the two files displayed in the left and right side panels.

**Ignore Whitespaces**

Enables or disables the whitespace ignoring feature. Ignoring whitespace means that before performing the comparison, the application normalizes the content and trims its leading and trailing whitespaces.

**Synchronized scrolling**

Toggles synchronized scrolling. When toggled on, a selected difference can be seen in both panels.

**Format and Indent Both Files (Ctrl + Shift + P (Command + Shift + P on macOS))**

Formats and indents both files before comparing them. Use this option for comparisons that contain long lines that make it difficult to spot differences.

**Note:**

When comparing two JSON files, the Format and Indent Both Files action will automatically sort the keys in both files the same to make it easier to compare.

**Copy Change from Right to Left**
Copies the selected difference from the file in the right panel to the file in the left panel.

**Copy All Changes from Right to Left**

Copies all changes from the file in the right panel to the file in the left panel.

**Next Block of Changes (Ctrl + Period (Command + Period on macOS))**

Jumps to the next block of changes. This action is not available when the cursor is positioned on the last change block or when there are no changes.

**Note:**

A change block groups one or more consecutive lines that contain at least one change.

**Previous Block of Changes (Ctrl + Comma (Command + Comma on macOS))**

Jumps to the previous block of changes. This action is not available when the cursor is positioned on the first change block or when there are no changes.

**Next Change (Ctrl + Shift + Period (Command + Shift + Period on macOS))**

Jumps to the next change from the current block of changes. When the last change from the current block of changes is reached, it highlights the next block of changes. This action is not available when the cursor is positioned on the last change or when there are no changes.

**Previous Change (Ctrl + Shift + Comma (Command + Shift + M on macOS))**

Jumps to the previous change from the current block of changes. When the first change from the current block of changes is reached, it highlights the previous block of changes. This action is not available when the cursor is positioned on the first change or when there are no changes.

**Ignore Nodes by XPath**

You can use this text field to enter an XPath expression (on page 1999) to ignore certain nodes from the comparison. It will be processed as XPath version 2.0. You can also enter the name of the node to ignore all nodes with the specified name (for example, if you want to ignore all ID attributes from the document, you could simply enter @id). This field is only available when comparing XML documents using the XML Fast or XML Accurate algorithms.

**Note:**

If an XPath expression is specified in the Ignore nodes by XPath option (on page 288) in the Diff / File Comparison preferences page, that one is used as a default when the application is started. If you then enter an expression in this field on the toolbar, this one will be used instead of the default. If you delete the expression from this field, neither will be used.

**First Change (Ctrl + B (Command + B on macOS))**

Jumps to the first change.
Most of these actions are also available from the Compare (on page 2719) menu.

**Image Preview**

You can view your local files by using the built-in **Image Preview** component. The view can be accessed from the Working copy view (on page 2813) or from the Repository view (on page 2808). It can also be used from the History view (on page 2828) to view a selected revision of a image file.

Only one image file can be opened at a time. If an image file is opened in the *Image preview* and you try to open another one it will be opened in the same window. Supported image types are GIF, JPEG/JPG, PNG, BMP. Once the image is displayed in the *Image Preview* panel using the actions from the contextual menu, you can scale the image at its original size (1:1 action) or scale it down to fit in the view’s available area (*Scale to fit* action).

**Compare Images View**

The images are compared using the **Compare Images** view. This view is automatically opened if you select two image files and use the **Compare with > Each Other** action in the contextual menu. The images are presented in the left and right part of the view, scaled to fit the available area. You can use the contextual menu actions to scale the images at their original size or scale them down to fit the view's available area.

The supported image types are: GIF, JPG / JPEG, PNG, BMP.

**Properties View**

The properties view presents Apache Subversion™ properties for the currently selected resource from either the Working Copy view or the Repositories view. If the view is not displayed, it can be opened by selecting it from the Window > Show View menu.

**Figure 725. Properties View**
The table includes four columns:

- **State** - Can be one of the following:
  - *(empty)* - Normal unmodified property (same current and base values)
  - *(asterisk)* - Modified property (current and base values are different)
  - *(plus sign)* - New property
  - *(minus sign)* - Removed property
- **Name** - The property name.
- **Current value** - The current value of the property.
- **Base value** - The base (original) value of the property.

**svn:externals Property**

The `svn:externals` property can be set on a folder or a file. In the first case, it stores the URL of a folder from another repository (on page 2825). In the second case, it stores the URL of a file from another repository. The external file will be added into the working copy as a versioned item. There are a few differences between directory and external file:

- The path to the external file must be in a working copy that is already checked out. While external directories can place the external directory at any depth and it will create any intermediate directories, external files must be placed into a working copy that is already checked out.
- The external file URL must be in the same repository as the URL that the external file will be inserted into (inter-repository external files are not supported).
- While commits do not descend into an external directory, a commit in a directory containing an external file will commit any modifications to the external file.

The difference between a normal versioned file and an external file is that external files cannot be moved or deleted (the `svn:externals` property must be modified instead. However, external files can be copied).

An external file is displayed as an X in the switched status column.

**Toolbar / Contextual Menu**

The properties view toolbar and contextual menu contain the following actions:

- **➕ Add a new property** - This button invokes the `Add property` dialog box where you can specify the property name and value.
- **☁️ Edit property** - This button invokes the `Edit property` dialog box where you can change the property value and also see its original(base) value.
- **✗ Remove property** - This button will prompt a dialog box to confirm the property deletion. You can also specify if you want to remove the property recursively.
- **↻ Refresh** - This action will refresh the properties for the current resource.
Console View

The **Console View** shows the traces of all the actions performed by the application. If the view is not displayed, it can be opened by selecting it from the **Window > Show View** menu.

Part of the displayed messages mirror the communication between the application and the Apache Subversion™ server. The output is expressed as subcommands to the Subversion server and simulates the Subversion command-line notation. For a detailed description of the Subversion console output read the [SVN User Manual](#).

The view has a simple layout, with most of its space occupied by a message area. On its right side, there is a toolbar holding the following buttons:

- **Clear**
  - Erases all the displayed messages.

- **Lock scroll**
  - Disables the automatic scrolling when new messages are appended in the view.

The maximum number of lines displayed in the console (length of the buffer) can be modified in the **Preferences (on page 281)** page. By default, this value is set to 100.

Dynamic Help View

**Dynamic Help view** is a help window that changes its content to display the help section that is specific to the currently selected view. As you change the focused view, you can read a short description of it and its functionality. If the view is not displayed, it can be opened by selecting it from the **Window > Show View** menu.

Revision Graph of an SVN Resource

The history of an SVN resource can be watched on a graphical representation of all the revisions of that resource together with the tags in which the resource was included. The graphical representation is identical to a tree structure and very easy to follow.

The graphical representation of a resource history is invoked with the **Revision graph** action available on the right-click menu of an SVN resource in the **Working Copy view (on page 2813)** and the **Repository view (on page 2808)**.
In every node of the revision graph an icon and the background color represent the type of operation that created the revision represented in that node. The commit message associated with that revision, the repository path, and the revision number are also contained in the node. The tooltip displayed when the mouse pointer hovers over a node specifies the URL of the resource, the SVN user who created the revision of that node, the revision number, the date of creation, the commit message, the modification type and the affected paths (on page 2741).

The types of nodes used in the graph are:

**Added resource**

The icon for a new resource added to the repository and a green background.

**Copied resource**

The icon for a resource copied to other location (for example, when an SVN tag is created and a green background).

**Modified resource**

The icon for a modified resource and a blue background.

**Deleted resource**

The icon for a resource deleted from the repository and a red background.

**Replaced resource**
The icon for a resource removed and replaced with another one on the repository and an orange background.

**Indirect resource**

The icon for a revision from where the resource was copied or an indirectly modified resource, that is a directory where a resource was modified and a gray background. The *Modification type* field of the tooltip specifies how that revision was obtained in the history of the resource.

A directory resource is represented with two types of graphs:

**Simplified graph**

Lists only the changes applied directly to the directory;

**Complete graph**

Lists also the indirect changes of the directory resource, that is the changes applied to the resources contained in the directory.

**Figure 727. Revision Graph of a Directory (Direct Changes)**

![Revision Graph of a Directory (Direct Changes)](image-url)
The Revision graph toolbar contains the following actions:

- **Save as image**
  
  Saves the graphical representation as image. For a large revision graph you have to set more memory in the startup script (on page 339). The default memory size is not enough when there are more than 100 revisions that are included in the graph.

- **Show/Hide indirect modifications**

  Switches between simplified and complete graph.

- **Zoom In**

  Zooms in the graph.

- **Zoom Out**

  Zooms out the graph. When the font reaches its minimum size, the graph nodes will display only the icons, leading to a very compact representation of the graph.

- **1:1 Reset scale**

  Resets the graphical scale to a default setting.

- **Print**

  Prints the graph.

- **Print preview**
Offers a preview of the graph to allow you to check the information to be printed.

The contextual menu of any of the graph nodes contains the following actions:

**Open**

Opens the selected revision in the editor panel. Available only for files.

**Open with**

Opens the selected revision in the editor panel. Available only for files.

**Save as**

Saves the file that had the revision graph generated, based on the selected node revision.

**Copy to**

Copies to the repository the item whose revision graph is displayed, using the selected revision.

**Note:**

This action can be used to resurrect deleted items also.

**Compare with HEAD**

Compares the selected revision with the HEAD revision and displays the result in the diff panel. Available only for files.

**Show History**

Displays the history of the resource in the History view (on page 2828). Available for both files and directories.

**Check out**

Checks out (on page 2739) the selected revision of the directory. Available only for directories.

**Export**

Opens the Export dialog box (on page 2803) that allows you to configure options for exporting a folder from the repository to the local file system.

When two nodes are selected in the revision graph of a file the right-click menu of this selection contains only the **Compare** for comparing the two revisions corresponding to the selected nodes. If the resource that had the revision graph built is a folder then the right-click menu displayed for a two nodes selection also contains the **Compare** action but it computes the differences between the two selected revisions as a set of directory changes. The result is displayed in the Directory Change Set (on page 2833) view.

**Attention:**

Generating the revision graph of a resource with many revisions may be a slow operation. You should enable caching for revision graph actions so that future actions on the same repository will not request the same data again from the SVN server, which will finish the operation much faster.
Oxygen XML Editor SVN Preferences

The options used in the SVN client are saved and loaded independently from the Oxygen XML Editor options. However, if Oxygen XML Editor cannot determine a set of SVN options to be loaded at startup, some of the preferences are imported from the XML Editor options (such as the License key and HTTP Proxy settings).

There is also an additional set of preferences applied to the SVN client that are set in global SVN files. There are two editing actions available in the Global Runtime Configuration submenu of the Options menu. These actions, Edit 'config' file and Edit 'servers' file, contain parameters that act as defaults applied to all the SVN client tools that are used by the same user on their login account.

Entering Local Paths and URLs

The Oxygen XML Editor includes a variety of option configuration pages or wizards that contain text boxes where you specify paths to local resources or URLs of items inside remote repositories. The Oxygen XML Editor provides support in these text boxes to make it easier to specify these paths and URLs.

Local Item Paths

The text boxes used for specifying local item paths support the following:

- **Absolute Paths** - In most cases, the Oxygen XML Editor expects absolute paths for local file system items.
- **Relative Paths** - The Oxygen XML Editor only accepts relative paths in the form `${user home directory}/[/...], where ~ is the user home directory.
- **Path Validation** - Oxygen XML Editor validates the path as you type and invalid text becomes red.
- **Drag and Drop** - You can drag files and folders from the file system or other applications and drop them into the text box.
- **Automatic Use of Clipboard Data** - If the text box is empty when its dialog box is opened, any data that is available in the system clipboard is used, provided that it is valid for that text box.

Repository Item URLs

- **Local Repository Paths** - You can use local paths (absolute or relative) to access local repositories. When you use the Browse button, the Oxygen XML Editor will convert the file path to a file:// form of URL, provided that the location is a real repository.
  - **Absolute Paths** - In most cases, the Oxygen XML Editor expects absolute paths for local file system items.
  - **Relative Paths** - The Oxygen XML Editor only accepts relative paths in the form `${user home directory}/[/...], where ~ is the user home directory.
- **Peg Revisions** - For URL text boxes found inside dialog boxes where you are pulling information from the repository, you can use peg revisions at the end of the URLs (on page 2850) (for example, URL@rev1234).
Note:
If you try to use a peg revision number in a dialog box where you are sending information to the repository then the peg revision number will become part of the name of the item rather than searching for the specified revision. For example, in the URL http://host/path/inside/repo/item@100, the item name is considered to be item@100.

Tip:
You can even use peg revisions with local repository paths. For example, C:\path\to\local\repo@100 will be converted to file:///C:/path/to/local/repo@100 and the Repository browser will display the content of the local repository as it is at revision 100.

- **URL Validation** - Oxygen XML Editor validates the URLs as you type and invalid text becomes red. Even paths to local repositories are not accepted unless using the **Browse** button to convert them to valid URLs.

- **Drag and Drop** - You can drag URLs from other applications or text editors and drop them into the URL text box. You can also drag folders that point to local repositories, from the local file system or from other applications, and they are automatically converted to valid file:/// type URLs.

- **Automatic Use of Clipboard Data** - If the URL text box is empty when its dialog box is opened, any data that is available in the system clipboard is used, provided that it is valid for that text box. Even valid local paths will be automatically converted to file:/// type URLs.

Note:
The text boxes that are in the form of a combo box also allow you to select previously used URLs, or URLs defined in the **Repositories** view.

Technical Issues
This section contains special technical issues found during the use of Syncro SVN Client.

**Authentication Certificates Not Saved**
If Syncro SVN Client prompts you to enter the authentication certificate, although you already provided it in a previous session, then you should make sure that your local machine user account has the necessary rights to store certificate files in the Subversion configuration folder (write access to Subversion folder and all its subfolders). Usually, it is located in the following locations:

- **Windows**: [HOME_DIR]\AppData\Roaming\Subversion
- **macOS and Linux**: [HOME_DIR]/.subversion
Updating Newly Added Resources

When you want to get a resource that is part of a newly created structure of folders from the repository, you need to also get its parent folders.

Figure 729. Incoming Changes

Syncro SVN Client allows you to choose how you want to deal with the entire structure from that moment onwards:

Update ancestor directories recursively

This option brings the entire newly added folders structure into your working copy. In this case, the update time depends on the total number of newly incoming resources, because of the full update operation (not updating only selected resource).

Update selected files only (leave ancestor directories empty)

This option brings a skeleton structure composed of the resource's parent folders only, and the selected resource at the end of the operation. All of the parent directories will have depth set to empty in your working copy, thus subsequent Synchronize operations will not report any remote modifications in those folders. If you need to update the folders to full-depth, you can use the Update to revision/depth action (on page 2822) from the working copy view.

Accessing Old Items from a Repository

Usually, you point to an item from a repository using a URL. However, sometimes this might not be enough because the URL alone might point to a different item than the one you want and a peg revision is needed.

A Subversion repository tracks any change made to its items by using revisions, which contain information such as the name of the author who made the changes, the date when they were made, and a number that uniquely identifies each of them. Over time, an item from a specific location in a repository evolves as a result of committing changes to it. When an item is deleted, its entire life cycle (all changes made to it from the moment it was created) remains recorded in the history of the repository. If a new item is created, with the same name and in the same location of the repository as a previously existing one, then both items are identified by the same URL even though they are located in different time frames. This is when a peg revision comes in handy. A peg revision is nothing more than a normal revision, but the difference between them is
made by their usage. Many of the Subversion commands also accept a peg revision as a way to precisely identify an item in time, beside an operative revision (the revision regarding the used command).

Example:

To illustrate an example, consider the following:

- A new repository file `config` was created, identified by the URL `http://host.com/myRepository/dir/config`.
- The file has been created at revision `10`.
- Over time, the file was modified by committing revisions `12`, `15`, `17`.

To access a specific version of the file identified by the `http://host.com/myRepository/dir/config` URL, you need to use a corresponding revision (the operative revision):

- If a revision number is used that is lower than `10`, an error is triggered, because the file has not been created yet.
- If a revision number is used that is between `10` and `19`, the specific version you are interested in would be obtained.

Note:

Although the file was modified in revisions `12`, `15`, `17`, it existed in all revisions between `10` and `19`. Starting with a revision where the file is modified, it has the same content across all revisions generated in the repository until another revision where it is modified again.

At this point, the file is deleted, creating revision `20`. Now, no version of the file can be accessed because it no longer exists in the latest repository revision. This is due to the fact that Subversion automatically uses the HEAD revision as a peg revision (it assumes any item currently exists in the repository if not instructed otherwise). However, using any of the revision numbers from the `10-19` interval (when the file existed) as a peg revision (beside the operative revision), will help Subversion to properly identify the time frame when the file existed, and access the file version corresponding to the operative revision. If you use a revision number greater than `19`, this will also trigger an error.

Continuing the example, suppose that at revision `30`, a directory called `config` is created in the same repository location as the deleted file. This means that the new directory will be identified by the same repository address: `http://host.com/myRepository/dir/config`. If you only use this URL in any Subversion command, you will access the new directory. You will also access the same directory if you use any revision number equal to or greater than `30` as peg revision. However, if you are interested in accessing an old version of the previously existing file, then you must use one of the revisions that existed (`10-19`), as a peg revision, similar to the previous case.
Checksum Mismatch Error

A Checksum Mismatch error could happen if an operation that sends or retrieves information from the repository to the working copy is interrupted. This means that there is a problem with the synchronization between a local item and its corresponding remote item.

If you encounter this error, try the following:

1. Identify the parent directory of the file that caused the error (the file name should be displayed in the error message).

   Note:
   If the parent directory is the root of the working copy or if it contains a large amount of items it is recommended that you check out the working copy again, rather than continuing with the rest of this procedure.

2. Identify the current depth (on page 2815) of that directory.
3. Update the parent directory using the Update to revision/depth action that is available from the contextual menu or the Working copy menu. For the Depth option, select This folder only (empty).

   Warning:
   If you have files with changes in this directory, those changes could be lost. You should commit your changes or move the files to another directory outside the working copy prior to proceeding with this operation.

4. After clicking OK the contents of the directory will be erased and the directory is be marked as having an empty depth (on page 2815).
5. Once again, update the same directory using the Update to revision/depth action. This time, for the Depth option, select the depth that was previously identified in step 2.
6. If you moved modified files to another directory outside the working copy, move them back to the original location inside the working copy.

If this procedure does not solve the error, you need to check out the working copy again and move possible changes from the old working copy to the new one.

External Tools

A command-line tool can be started in the Oxygen XML Editor user interface as if from the command line of the operating system shell. Oxygen XML Editor offers you the option of integrating such a tool by specifying just the command line for starting the executable file and its working directory. To integrate such a tool, open the Preferences dialog box (Options > Preferences) (on page 124) and go to External Tools (on page 292) (or select Configure from the Tools > External Tools menu).
The **External Tools** preferences page *(on page 292)* presents a list of the external tools that have been configured. Once a tool has been configured *(on page 292)*, you can open it by selecting it from the **Tools > External Tools** menu or from the **External Tools** drop-down menu on the toolbar (the **Tools** toolbar needs to be selected in the **Configure Toolbars** dialog box *(on page 365)*). You can also assign a keyboard shortcut *(on page 294)* to be used to launch the tool.

If the external tool is applied on one of the files opened in Oxygen XML Editor, the **Save all files before calling external tools** option *(on page 202)* (in the **Save** preference page) should be selected so that all edited files are automatically saved when an external tool is applied.

When an external tool is launched, the icon on the toolbar changes to a stop icon (■) and you can use this button to stop the tool. When the tool has finished running (or you close it), the icon changes back to the original icon (▶).

Note:
Even though you can stop the external tool by invoking the stop action while it is running, that does not necessarily mean it will also stop the processes spawned by that external tool. For instance, if you stop an external tool that runs a batch file, the batch may be stopped but without actually stopping the processes that the batch was running at that time.

**Example: Integrating the Ant Tool**

This is an example procedure for integrating the **Ant build tool** into Oxygen XML Editor:

1. **Download** and **install Apache Ant** *(on page 3216)* on your computer.
2. Test your **Ant** installation from the command-line interface in the directory where you want to use **Ant** from. For example, run the **clean** target of your **build.xml** file:
   ```
   ant clean
   ```
3. **Open the Preferences** dialog box *(Options > Preferences)* *(on page 124)* and go to **External Tools** *(or select **Configure** from the **Tools > External Tools** menu).*
4. Click the **New** button to create a new external tool entry and enter the following information:
   - **Name** - For example, **Ant tool**.
   - **Working directory** - For example, **C:\projects\XMLproject**.
   - **Command line** - For example, "**C:\projects\XMLproject\ant.bat**" **clean**.
5. Click **OK** to add the new tool to the list of external tools.
6. Run the tool from **Tools > External Tools > Ant tool**. You can see the output in the system console:

   ```
   Started: "C:\projects\XMLproject\ant.bat" clean
   Buildfile: build.xml
   clean:
   ```
[echo] Delete output files.

[delete] Deleting 5 files from C:\projects\XMLproject

BUILD SUCCESSFUL

Total time: 1 second
21. Troubleshooting

This section provides a collection of common performance and other types of problems that might be encountered when using Oxygen XML Editor, along with their possible solutions.

Performance Problems and Solutions

This section contains solutions for some common performance problems that may appear when running Oxygen XML Editor.

Related Information:

- Documents with Long Lines (on page 471)
- Loading Large Documents (on page 469)
- External Tools (on page 2852)

Display Problems on Linux or Solaris

Problem

I experience display problems (such as screen freezes) on Linux or Solaris.

Cause

This is possibly a rendering problem with the off-screen pixmap support.

Solution

Add the following startup parameter (on page 339): `-Dsun.java2d.pmoffscreen=false`.

Out of Memory on External Processes

Problem

Oxygen XML Editor throws an Out Of Memory error when trying to generate PDF output with the built-in Apache FOP processor.

Cause

The amount of allocated memory might be insufficient.
Solutions

- Open the Preferences dialog box (Options > Preferences) (on page 124), go to XML > PDF Output > FO Processors, and increase the value of the Memory available to the Apache FOP option (on page 262).
- For external XSL-FO processors, XSLT processors, and external tools, the maximum value of the allocated memory is set in the command line of the tool using the -Xmx parameter set to the Java virtual machine.

Related Information:
FO Processors Preferences (on page 261)
Custom Engines Preferences (on page 259)
External Tools Preferences (on page 292)
How to Enable Debugging for FO Processor Transformations (on page 1529)

Too many nested apply-templates calls Error When Running a Transformation

Problem

I'm getting the error message Too many nested apply-templates calls when I try to transform my DocBook file to HTML using default Oxygen XML Editor DocBook to HTML transformation scenario.

Cause

Most likely, this is the result of a masked stack overflow error.

Solution

Try setting a new VM option with the value -Xss4m. You can also try to slowly increase this to larger values (e.g. -Xss5m or -Xss6m). Note that this consumes memory on a per thread basis (Oxygen XML Editor can have tens of threads), so using a very large value here can backfire and leave Oxygen XML Editor without memory.

Related Information:
Setting a Java Virtual Machine Parameter when Launching Oxygen XML Editor (on page 339)

Performance Issues with Large Documents

Problem

The performance of the application slows down considerably over time when working with large documents.

Cause

A possible cause is that the application needs more memory to run properly.
Solutions

• You can increase the maximum amount of memory available to Oxygen XML Editor by setting the `-Xmx` parameter in a configuration file (on page 339) that is specific to the platform that runs the application.

⚠️ **Attention:**

The maximum amount of memory should be less than 75% of the physical amount of memory available on the machine. Otherwise, the operating system and other applications will have no memory available.

• When installed on a multiple user environment, each instance of Oxygen XML Editor will be allocated the amount stipulated in the memory value. To avoid degrading the general performance of the host system, ensure that the amount of memory available is optimally apportioned for each of the expected instances.

📝 **Note:**

When starting Oxygen XML Editor from the icon created on the Start menu or Desktop in Windows (or from the shortcut created on the Linux desktop), the default maximum memory available to the application is set to 40% of the amount of physical RAM (but not more than 1 GB for 32-bit distributions or 4 GB for 64-bit distributions).

When starting Oxygen XML Editor from a command-line script, the default maximum memory is 1 GB for 32-bit distributions or 4 GB for 64-bit distributions.

**Performance Issues when Using Oxygen XML Editor with Remote Desktop**

**Problem**

When trying to run Oxygen XML Editor in a Remote Desktop Protocol (RDP) environment, the performance is slow and choppy.

**Cause**

Running a standalone version of Oxygen XML Editor over a slow RDP connection may result in performance issues.

**Solution**

As a workaround, you try to run Oxygen XML Editor as an Eclipse plugin when working with a slow RDP connection.

**Misc Problems and Solutions**

This chapter presents common problems that may appear when running the application along with solutions for these problems.
Address Family Not Supported by Protocol Family

Problem

I have experienced the following error: "Address Family Not Supported by Protocol Family; Connect". How do I solve it?

Cause

This seems to be an IPv6 connectivity problem. By default, the Java runtime used by Oxygen XML Editor prefers to create connections via IPv6, if the support is available. However, even though it is available in appearance, IPv6 sometimes happens to be configured incorrectly on some systems.

Solution

A quick solution for this problem is to set the `java.net.preferIPv4Stack` Java property to `true` (`java.net.preferIPv4Stack=true`), by following this procedure:

1. Create a file named `custom_commons.vmoptions` and on a single line, add `-Djava.net.preferIPv4Stack=true`. Then save the file and copy it to the Oxygen XML Editor installation folder (may need admin access).
2. Restart Oxygen XML Editor.
3. Make sure the procedure was successful by going to Help > About > System properties and check that the value of the `java.net.preferIPv4Stack` property is `true`.

Application Reports Errors During Startup After Installing a New Version

Problem

Sometimes, after installing a new version of Oxygen XML Editor, various errors are reported when the application starts.

Cause

This problem may occur if you install the application in a folder where an older version of the application was previously installed. Especially on macOS, there is a possibility for older resources and libraries from the previous application to remain in the installation folder and break the functionality of the newer version of the application.

Solution

Close the application and completely uninstall it (on page 122), then install it again. The user-specific settings are saved in a separate folder in the user home directory so they will not be lost.

- On macOS, you can move the entire application installation folder to the Trash, then re-install.
- On Linux and Windows, you can uninstall using the facilities provided by the installer (on page 122), then re-install.
If you intentionally want to load extra Java libraries with Oxygen XML Editor, you have the following choices:

- If the libraries are necessary for XSLT transformations, each XSLT transformation scenario has an Extensions button that allows you to reference the libraries directly from the transformation scenario.
- If the libraries are necessary for database connections, you can configure them when you define the data sources.
- You can add a plugin in Oxygen XML Editor that contributes libraries to the global libraries list. The plugin can be distributed as an add-on. An example of such a plugin can be found here: https://github.com/oxygenxml/oxygenxml.xlsx.import.
- In the Oxygen XML Editor lib folder, there is a file called libraries.list. You can edit that file and add the names of the extra libraries present in the folder. You can also choose to delete that libraries.list completely if you want to inhibit the libraries checking completely.

**Application Takes Several Minutes to Start**

**Problem**

Oxygen XML Editor seems to take an abnormally long amount of time to start.

**Cause**

Some anti-virus software can cause Java applications, such as Oxygen XML Editor, to start very slowly due to scanning compressed archives (such as the JAR libraries that all Java applications use).

**Solution**

A possible solution is to add the Oxygen XML Editor folder to the list of exceptions in the anti-virus software settings.

**Archive Distribution Fails to Run on macOS 10.12 (Sierra)**

**Problem**

For versions prior to 18.1, the classic archive distributions of Oxygen XML Editor (.zip or .tar.gz) fail to run on macOS 10.12 (Sierra).

**Cause**

This happens because the archives get quarantined and macOS 10.12 (Sierra) treats quarantined apps differently than older versions and isolates them from their parent folder at launch. If Oxygen XML Editor was already installed when you upgraded to macOS 10.12 (Sierra), there are no problems.

**Solution**

If Oxygen XML Editor was not already installed, or you need to reinstall an older version (prior to version 18.1), the quarantine flag must be removed for the entire content of the Oxygen XML Editor installation directory or the individual applications. To resolve this issue, follow these steps:
1. Open a *Terminal* window and change the directory to the folder where Oxygen XML Editor is located.

2. Run the following command:

   ```
   xattr -dr com.apple.quarantine oxygen/
   ```

   where “oxygen” is the actual name of the Oxygen XML Editor directory.

   If Oxygen XML Editor is in a location that requires administrator privileges for write access, you need to run the command from an administrator account and prefix the command with `sudo`. You will then be prompted to enter your password.

### Cannot Connect to SVN Repository from Repositories View

**Problem**

I cannot connect to an SVN repository from the **Repositories** view of SVN Client. How can I find more details about the error?

**Solution**

First check that you entered the correct URL of the repository in the **Repositories** view. Also, check that an SVN server is running on the server machine specified in the repository URL and is accepting connections from SVN clients. You can check that the SVN server accepts connections with the command-line SVN client from CollabNet.

If you try to access the repository with an `svn+ssh` URL, also check that an SSH server is running on port 22 on the server machine specified in the URL.

If the above conditions are verified and you cannot connect to the SVN repository, generate logging files on your computer and send them to support@oxygenxml.com. For generating the logging files, follow these steps:

1. Create a text file called `logback.xml` in the application installation folder with the following content:

   ```xml
   <configuration>
     <appender name="R2" class="ch.qos.logback.core.rolling.RollingFileAppender"  
       <file>${user.home}/Desktop/oxygenLog/oxygen.log</file>
       <rollingPolicy class="ch.qos.logback.core.rolling.FixedWindowRollingPolicy">
         <fileNamePattern>${user.home}/Desktop/oxygenLog/oxygen%i.log.gz</fileNamePattern>
         <minIndex>1</minIndex>
         <maxIndex>20</maxIndex>
       </rollingPolicy>
       <triggeringPolicy class="ch.qos.logback.core.rolling.SizeBasedTriggeringPolicy">
         <maxFileSize>12MB</maxFileSize>
       </triggeringPolicy>
     </appender>
     <encoder>
       <pattern>%r %marker %p [ %t ] %c - %m%n</pattern>
   </configuration>
   ```
2. Restart the application.
3. Reproduce the error.
4. Close the application.
5. Delete the logback.xml file because it might cause performance issues if you leave it in the installation folder.

⚠️ Important:

The logging mode may severely decrease the performance of the application. Therefore, do not forget to delete the logback.xml file when you are done with the procedure.

Result: The resulting logging files are named oxygen.log and oxygen#.log.gz (for example, oxygen.log, oxygen1.log.gz, oxygen2.log.gz, etc.) and are located in the Desktop\oxygenLog folder.

Cannot Uninstall Oxygen XML Editor in Windows

Problem

When I try to uninstall Oxygen XML Editor in Windows, I get an error that says it cannot find the files.log file.

Cause

The install4j installer that is used by Oxygen XML Editor creates the files.log file during the installation process. If you cannot uninstall the product, then most likely something went wrong with this file during the installation process.

Solution

To solve this, simply reinstall the software in the same directory as the current installation. This will automatically uninstall the old version or overwrite it with a clean install. You should then be able to uninstall this new installation.
Compatibility Issue Between Java and Certain Graphics Card Drivers

Problem

Under certain settings, a compatibility issues can appear between Java and some graphics card drivers, which results in the text from the editor (in Author or Text mode) being displayed garbled.

Solution

If you encounter this problem, update your graphics card driver. Another possible workaround is, open the Preferences dialog box (Options > Preferences) (on page 124), go to Appearance > Fonts, and set the value of the Text antialiasing option (on page 134) to ON.

Note:
If this workaround does not resolve the problem, set the Text antialiasing option (on page 134) to other values.

Crash at Startup on Windows with an Error About the nvogl32.dll File

Problem

I try to start Oxygen XML Editor on Windows but it crashed with an error message about "Fault Module Name: nvogl32.dll". What is the problem?

Cause

It is most likely an OpenGL driver issue.

Solution

This can be avoided by creating an empty file called opengl32.dll in the Oxygen XML Editor install folder (if you start Oxygen XML Editor with the shortcut created by the installer on the Start menu or on Desktop) or in the subfolder bin of the home folder of the Java virtual machine that runs Oxygen XML Editor (if you start Oxygen XML Editor with the oxygen.bat script). The home folder of the Java virtual machine that runs Oxygen XML Editor is the value of the java.home property that is available in the System properties tab of the About dialog box (Help > About).

Crash in Java Runtime Thread

Problem

I encounter consistent crashes in one of the Java runtime threads (for example, AWT-EventQueue, Java2D Queue Flusher, or C2 CompilerThread).

Cause

This is usually caused by unknown errors somewhere in the implementation of the Java Runtime.
Solution

Although this does not guarantee that it will solve the problem, you could try updating the Java Runtime Environment (JRE) that comes bundled with Oxygen XML Editor to the latest supported version.

⚠️ Important:
You should only use a version of Java 8.

Windows/Linux

1. Download the latest Java 8 JRE from here: http://www.oracle.com/technetwork/java/javase/downloads/jre8-downloads-2133155.html. Download the JRE in the tar.gz format with the same architecture as your Oxygen XML Editor installation (x86=32bit / x64=64bit).
2. Navigate to the Oxygen XML Editor installation folder and rename the jre folder to jre.old (you will need admin privileges if it is installed in Program Files).
3. Unpack the tar.gz and rename the extracted folder to jre.
4. Move that folder to the Oxygen XML Editor installation folder.
5. Verify the paths in the jre folder. The path of the JRE bin folder should be: oxygen/jre/bin.
6. Start Oxygen XML Editor and check the Java version in Help > About > System properties. The java.runtime.version should be match the one you unpacked.

macOS

2. Extract the archive contents in a location of your choice (for example, Downloads). You will end up with a folder named something like this: jrel.8.0_181.jre (depending on the version).
3. Move or rename the JRE that was bundled with Oxygen XML Editor. The bundled JRE folder (jre.bundle) is located inside a hidden folder named .install4j (in the Oxygen XML Editor installation folder), so you cannot see it in the Mac Finder. Thus, you need to move or rename it via the command line.

   For example, if you have Oxygen XML Editor deployed in the Applications folder, the command should look similar to this:

   ```
   mv "~/Applications/Oxygen XML Editor/.install4j/jre.bundle" 
   "~/Applications/Oxygen XML Editor/.install4j/jre.bundle.old"
   ```

4. Move the unpacked JRE folder from the location where you extracted it in the second step to a folder named .install4j/jre.bundle inside the Oxygen XML Editor installation direction.

   For example, if you have Oxygen XML Editor deployed in the Applications folder, the command should look similar to this:

   ```
   mv "~/Downloads/jrel.8.0_181.jre"/Applications/Oxygen XML Editor/.install4j/jre.bundle"
   ```
The default bundled JRE should now be successfully replaced by the newer one.

5. Start Oxygen XML Editor and check the Java version in Help > About > System properties. The java.runtime.version should be match the one you unpacked.

### Damaged File Associations on macOS

#### Problem

After upgrading macOS and Oxygen XML Editor, it is no longer associated to the appropriate file types (such as XML, XSL, XSD). How can I re-create the file associations?

#### Cause

The upgrade damaged the file associations in the LaunchService Database on your macOS machine.

#### Solution

You can rebuild the LaunchService Database with the following procedure. This will reset all file associations and rescan the entire file system searching for applications that declare file associations and collect them in a database used by Finder.

1. Find all the Oxygen XML Editor installations on your hard drive.
2. Delete them by dragging them to the Trash.
3. Clear the Trash.
4. Unpack the Oxygen XML Editor installation kit on your desktop.
5. Copy the contents of the archive into the folder `/Applications/Oxygen`.
6. Run the following command in a Terminal:

   ```bash
   /System/Library/Frameworks/CoreServices.framework/Versions/A/Frameworks/LaunchServices.framework/Versions/A/Support/lsregister -kill -r --domain local --domain system --domain user
   ```
7. Restart Finder with the following command:

   ```bash
   killall Finder
   ```
8. Create an XML or XSD file on your desktop. It should have the Oxygen XML Editor icon.
10. Accept the confirmation.

**Result:** When you start Oxygen XML Editor, the file associations should work correctly.

### Details to Submit in a Request for Technical Support Using the Online Form

#### Problem

What details should I add to my request for technical support on the online form in the product website?
Solution

When completing a request for Technical Support using the online form, include as many details as possible about your problem. For problems where a simple explanation may not be enough for the Technical Support team to reproduce or address the issue (such as server connection errors, unexpected delays while editing a document, an application crash, etc.), you should generate log files and attach them to the problem report. In the case of a crash, you should also attach the crash report file generated by your operating system.

If the text content of an XML document you want to send to the support team contains sensitive or private information, you can use the Randomize XML text content action (on page 53) to create filler content. Before using this action, you need to copy the initial XML resources and save them in a separate folder. Otherwise, you might lose your original information.

To generate the Oxygen XML Editor log files, follow these steps:

1. Create a text file called logback.xml in the application installation folder, with the following content:

   ```xml
   <configuration>
     <appender name="R2" class="ch.qos.logback.core.rolling.RollingFileAppender">
       <file>${user.home}/Desktop/oxygenLog/oxygen.log</file>
       <rollingPolicy class="ch.qos.logback.core.rolling.FixedWindowRollingPolicy">
         <fileNamePattern>${user.home}/Desktop/oxygenLog/oxygen%i.log.gz</fileNamePattern>
         <minIndex>1</minIndex>
         <maxIndex>20</maxIndex>
       </rollingPolicy>
       <triggeringPolicy class="ch.qos.logback.core.rolling.SizeBasedTriggeringPolicy">
         <maxFileSize>12MB</maxFileSize>
       </triggeringPolicy>
       <encoder>
         <pattern>%r %marker %p [%t] %c - %m%n</pattern>
       </encoder>
     </appender>
     <root level="debug">
       <appender-ref ref="R2"/>
     </root>
   </configuration>
   ```

2. Restart the application.
3. Reproduce the error.
4. Close the application.
5. Delete the logback.xml file because it might cause performance issues if you leave it in the installation folder.

**Important:**
The logging mode may severely decrease the performance of the application. Therefore, do not forget to delete the logback.xml file when you are done with the procedure.

**Result:** The resulting log files are named oxygen.log and oxygen#.log.gz (for example, oxygen.log, oxygen1.log.gz, oxygen2.log.gz, etc.) and are located in the Desktop\oxygenLog folder.

### Dialog Boxes Cannot Be Resized on Mac

**Problem**
When using Oxygen XML Editor on macOS Big Sur, dialog boxes (for example the Find/Replace or Preferences dialog box) cannot be resized.

**Cause**
This is caused by an issue with resizing dialog boxes in Oxygen XML Editor on macOS Big Sur (and possibly later versions) causing crashes if the main application is in full screen mode.

**Solution**
Until this limitation is resolved in a future Oxygen XML Editor version, dialog boxes cannot be maximized or resized on macOS Big Sur.

### DITA Map Transformation Fails (Cannot Connect to External Location)

**Problem**
DITA map (on page 3218) transformation fails because it cannot connect to an external location.

**Solution**
The transformation is run as an external Ant process so you can continue using the application as the transformation unfolds. All output from the process appears in the DITA Transformation tab.

The HTTP proxy settings are used for the Ant transformation, so if the transformation fails because it cannot connect to an external location, you can check the Proxy preferences page (on page 301)

### DITA Map WebHelp Transformation Fails (Duplicate Topic References Found)

**Problem**
DITA Map WebHelp transformation fails with a message that indicates duplicate topic references were found.
**Cause**

By default the WebHelp transformation uses the `force-unique` parameter set to `true` to force the transformation to create unique output files for each instance of a resource when a map contains multiple references to a single topic. However, there are cases when this feature does not work as expected and the duplicate topic references are not handled properly.

**Solution**

To solve this issue, you should manually set a unique `@copy-to` attribute on any duplicate topic reference that was not handled automatically by DITA-OT:

```xml
<map>
  ...
  <topicref href="../topics/MyTopic.dita"/>
  ...
  <topicref href="../topics/MyTopic.dita" copy-to="../topics/MyTopic-2.dita"/>
</map>
```

**DITA-OT Transformation Takes a Long Time to Process**

**Problem**

A DITA transformation takes an extremely long time to process (over an hour, for example).

**Cause**

Large delays in DITA-OT processing are usually caused by intensive disk operations, CPU usage, or connections to remote websites. The DITA-OT processing is very disk-intensive, each stage takes the entire content from the transformation temporary files folder, reads it, modifies it, and then writes it back.

**Solution**

There are several things you can try to troubleshoot this problem:

- If you are using a shared or remote drive, it is recommended to specify a local drive for the output and temporary files directory (edit the transformation scenario and in the **Output** tab, select a local directory for **Temporary files directory** and **Output directory**).
- If you want to test if the publishing has a problem downloading remote resources, you could disable the network adapter on the computer and then try to publish. The purpose is to see if the publishing finishes without any reported error about obtaining a certain HTTP resource.
- Using DTDs instead of XML Schemas is faster. This is because of a default transformation parameter called `args.grammar.cache` that only works for DTD-based DITA topics.
- You can increase the memory available to Oxygen XML Editor (on page 2856). Sometimes, just increasing the amount of memory available to the DITA-OT process may be enough to lower the time necessary for the publishing to run.
• You can enable some logging to help you determine which stage in the process is taking a long time. Edit the transformation scenario and in the Advanced tab, enter logger org.apache.tools.ant.listener.ProfileLogger in the Additional arguments field. Then go to Options > Preferences > DITA > Logging and select Always for the Show console output option.

• You could try disabling antivirus applications since the publishing process is very disk intensive and certain antivirus application might slow down the process.

• If the published DITA map is part of a larger DITA project with lots of maps and topics, references from topics in the current map to topics in other sub-projects might result in problems resolving those references. You could look in the output folder to see if the number of HTML documents match the number of DITA topics in your map.

DITA PDF Transformation Fails

Problem

The DITA to PDF transformation fails.

Cause

To generate the PDF output, Oxygen XML Editor uses the DITA Open Toolkit. This process sometimes results in errors. For information about some of the most common errors, see DITA PDF Processing Common Errors (on page 3129).

Solution

If your transformation fails, you can detect some of the problems that caused the errors by running the Validate and Check for Completeness action (on page 2941). Depending on the options you select when you run it, this action reports errors such as topics referenced in other topics but not in the DITA map (on page 3218), broken links, and missing external resources.

You can analyze the Results tab of the DITA transformation and search for messages that contain text similar to [fop] [ERROR]. If you encounter this type of error message, edit the transformation scenario you are using and set the clean.temp parameter to no and the retain.topic.fo parameter to yes. Run the transformation, go to the temporary directory of the transformation, open the topic.fo file and go to the line indicated by the error. Depending on the XSL FO context try to find the DITA topic that contains the text that generates the error.

If none of the above methods helps you, go to Help > About > Components > Frameworks and check what version of the DITA Open Toolkit you are using. Copy the whole output from the DITA-OT console output and either report the problem on the DITA User List or send it to support@oxygenxml.com.

Related Information:

How to Enable Debugging for FO Processor Transformations (on page 1529)
DITA PDF Processing Common Errors

There are cases when the PDF processing fails when trying to publish DITA content to a PDF file. This topic lists some of the common problems and possible solutions.

Problem: Cannot Save PDF

The FO processor cannot save the PDF at the specified target. The console output contains messages like this:

```
[fop] [ERROR] Anttask - Error rendering fo file:
C:\samples\dita\temp\pdf\oxygen_dita_temp\topic.fo
<Failed to open C:\samples\dita\out\pdf\test.pdf>
Failed to open samples\dita\out\pdf\test.pdf
............
[fop] Caused by: java.io.FileNotFoundException:
C:\Users\default\Desktop\bev\out\pdf\test.pdf
(The process cannot access the file because it is being used by another process)
```

Solution: Cannot Save PDF

Such an error message usually means that the PDF file is already opened in a PDF reader application. The solution is to close the open PDF before running the transformation.

Problem: Table Contains More Cells Than Defined in Colspec

One of the DITA tables contains more cells in a table row than the defined number of `<colspec>` elements. The console output contains messages like this:

```
[fop] [ERROR] Anttask - Error rendering fo file:
D:\projects\eXml\samples\dita\flowers\temp\pdf\oxygen_dita_temp\topic.fo
<net.sf.saxon.trans.XPathException: org.apache.fop.fo.ValidationException:
The column-number or number of cells in the row overflows the number of
fo:table-columns specified for the table.
(See position 179:-1)>net.sf.saxon.trans.XPathException:
org.apache.fop.fo.ValidationException: The column-number or number of cells
in the row overflows the number of fo:table-columns specified for the table.
(See position 179:-1)
[fop]  at org.apache.fop.tools.anttasks.FOPTaskStarter.renderInputHandler
(Fop.java:657)
[fop]  at net.sf.saxon.event.ContentHandlerProxy.startContent
(ContentHandlerProxy.java:375)
............
[fop]  D:\projects\samples\dita\flowers\temp\pdf\oxygen_dita_temp\topic.fo ->
D:\projects\samples\dita\flowers\out\pdf\flowers.pdf
```
Solution: Table Contains More Cells Than Defined in Colspec

To resolve this issue, correct the `@colspec` attribute on the table that caused the issue. To locate the table that caused the issue:

1. Edit the transformation scenario and set the parameter `clean.temp` to `no`.
2. Run the transformation, open the `topic.fo` file in Oxygen XML Editor, and look at it at the line specified in the error message (See position 179:-1).
3. Look around that line in the `XSL-FO` file to find relevant text content that you can use (for example, with the `Find/Replace in Files` action in the `DITA Maps Manager view (on page 2897)` to find the original DITA topic where the table was generated.

Problem: Broken Link

There is a broken link in the generated `XSL-FO` file. The PDF is generated but contains a link that is not working. The console output contains messages like this:

```
[fop] 1248 WARN [ main ] org.apache.fop.apps.FOUserAgent -
Page 6: Unresolved ID reference "unique_4_Connect_42_wrongID" found.
```

Solution: Broken Link

To resolve this issue:

1. Use the `Validate and Check for Completeness` action available in the `DITA Maps Manager view (on page 2897)` to find such problems.
2. If you publish to PDF using a `DITAVAL` filter, select the same `DITAVAL` file in the `DITA Map Completeness Check` dialog box.
3. If the `Validate and Check for Completeness` action does not discover any issues, edit the transformation scenario and set the `clean.temp` parameter to `no`.
4. Run the transformation, open the `topic.fo` file in Oxygen XML Editor, and search for the `unresolved ID references` (for example: `unique_4_Connect_42_wrongID`).
5. Look in the `XSL-FO` file to find relevant text content that you can use (for example, with the `Find/Replace in Files` action in the `DITA Maps Manager view (on page 2897)`) to find the original DITA topic where the table was generated.

Related Information:

How to Enable Debugging for FO Processor Transformations (on page 1529)

DITA to CHM Transformation Fails - Cannot Open File

Problem

The DITA to CHM transformation fails with the following error: `[exec] HHC5010: Error: Cannot open "fileName.chm". Compilation stopped.`
Cause

This error occurs when the CHM output file is opened and the transformation scenario cannot rewrite its content.

Solution

To solve this issue, close the CHM help file and run the transformation scenario again.

Tip:

It is a good practice to validate the DITA map (on page 3218) before executing the transformation scenario. To do so, run the Validate and Check for Completeness action (on page 2941). Depending on the selected options, it will report detected errors, such as topics referenced in other topics (but not in the DITA map), broken links, and missing external resources.

Related Information:

DITA Map CHM (Compiled HTML Help) Transformation (on page 3099)

DITA to CHM Transformation Fails - Compilation Failed

Problem

The DITA to CHM transformation fails with the following error: [exec] HHC5003: Error: Compilation failed while compiling fileName.

Cause 1

One possible cause for this error is that the processed file does not exist.

Solution 1

To solve this issue, fix the file reference before executing the transformation scenario again.

Cause 2

Another possible cause for this error is that the processed file has a name that contains space characters.

Solution 2

To solve the issue, remove any spacing from the file name and run the transformation scenario again.

Tip:

It is a good practice to validate the DITA map (on page 3218) before executing the transformation scenario. To do so, run the Validate and Check for Completeness action (on page 2941). Depending on the selected options, it will report detected errors, such as topics referenced in other topics (but not in the DITA map), broken links, and missing external resources.
Handshake Failure Error When Accessing an HTTPS (SSL) Resource

Problem

When attempting to access an HTTPS (SSL) resource, I receive a `handshake_failure` error.

Cause

The issue is most likely due to the limitation of Java cryptography capabilities.

Solution

A solution might be to download and deploy *Java Cryptography Extension (JCE) Unlimited Strength Jurisdiction Policy Files 8 (for Java 8)*.

⚠️ Warning:

It is possible that this may not be legal in your country. Be advised that you bare legal responsibility for activating unlimited strength Java cryptography capabilities, so you must have the legal right to use such cryptography (consult your export/import control counsel or attorney to determine the exact requirements for your jurisdiction).

To deploy it in Oxygen XML Editor, unpack the downloaded zip archive and move the two jar files (`local_policy.jar` and `US_export_policy.jar`) from `UnlimitedJCEPolicyJDK8` to the following directory, overwriting existing files:

- **Windows** - `{OXYGEN_INSTALL_DIR}/jre/lib/security`
- **Linux** - `{OXYGEN_INSTALL_DIR}/jre/lib/security`
- **macOS** - `{OXYGEN_INSTALL_DIR}/jre.bundle/Contents/Home/jre/lib/security`

Hunspell Spell Checker is Unusable on Your Platform Error

Problem

When trying to use the Check Spelling option, I receive the error *Hunspell spell checker is unusable on your platform. It has crashed the application in a previous session*.

Cause

There are instances where Oxygen XML Editor determines that an internal component (such as the spell checker) has crashed the application and disables that component from running in the future (to prevent a possible future crash).
**Solution**

To re-enable the spell checker component, follow these steps:

1. Close Oxygen XML Editor.
2. Open the %APPDATA%/com.oxygenxml folder and look for a file called something like HunspellCrashGuard*.txt. Delete that file.
3. Restart Oxygen XML Editor.

**High Resolution Scaling Issues**

**Problem**

I encounter scaling detection issues in a high resolution display (for example, some GUI components are too small).

**Cause**

This sometimes happens when using multiple displays with different resolutions because the application cannot detect the correct scaling setting.

**Solution**

**Windows with Java 8** - You can use the `com.oxygenxml.hidpi.scaling` custom system property (on page 336) to force a specific scaling setting. For example: `-Dcom.oxygenxml.hidpi.scaling=1.5` for 150% scaling.

**Windows/Linux/MacOS with OpenJDK 11 or later** - You can use the `sun.java2d.uiScale` Java system property to instruct Java to use a particular scaling factor:

```
-Dsun.java2d.uiScale=1.5
```

**High Resolution Scaling Issues on Linux**

**Problem**

On Linux bundled with Oracle OpenJDK 11 or newer, Oxygen XML Editor does not automatically scale high-resolution images when using the system's scaling settings.

**Cause**

This happens because Java 11 (and higher) does not detect the system scaling setting for HiDPI displays on Linux operating system.

**Solution**

In the Oxygen XML Editor installation folder, create a new file named `custom_commons.vmoptions`. Inside the file, manually add `-Dsun.java2d.uiScale=2`. This command instructs Java to use 2x (200%) scaling.
Images Appear Stretched Out in the PDF Output

Problem

When publishing XML content (DITA, DocBook, etc.), images are sometimes scaled up in the PDF outputs but are displayed perfectly in the HTML (or WebHelp) output.

Solution

PDF output from XML content is obtained by first obtaining an intermediary XML format called XSL-FO and then applying an XSL-FO processor to it to obtain the PDF. This stretching problem is caused by the fact that all XSL-FO processors take into account the DPI (dots-per-inch) resolution when computing the size of the rendered image.

The PDF processor that comes out of the box with the application is the open-source Apache FOP processor. Here is what Apache FOP does when deciding the image size:

1. If the XSL-FO output contains width, height or a scale specified for the image `<external-graphic>` tag, then these dimensions are used. This means that if in the XML (DITA, DocBook, etc.) you set explicit dimensions to the image they will be used as such in the PDF output.
2. If there are no sizes (width, height or scale) specified on the image XML element, the processor looks at the image resolution information available in the image content. If the image has such a resolution saved in it, the resolution will be used and combined with the image width and height to obtain the rendered image dimensions.
3. If the image does not contain resolution information inside, Apache FOP will look at the FOP configuration file for a default resolution. The FOP configuration file for XSLT transformations that output PDF is located in the `[OXYGEN_INSTALL_DIR]/lib/fop.xconf`. DITA publishing uses the DITA Open Toolkit that has the Apache FOP configuration file located in `[DITA-OT-DIR/plugins/org.dita.pdf2.fop/fop/conf/fop.xconf]`. The configuration file contains two XML elements called `<source-resolution>` and `<target-resolution>`. The values set to those elements can be increased (usually a DPI value of 110 or 120 should render the image in PDF the same as in the HTML output).

The commercial RenderX XEP XSL-FO processor behaves similarly but as a fallback it uses 120 as the DPI value instead of using a configuration file.

Tip:

It is best to save your images without any DPI resolution information. For example, when saving a PNG image in the open-source GIMP image editor, you do not want to save the resolution.
Increasing the Memory for the Ant Process

Problem

The Ant build process runs out of memory.

Solution

For details about setting custom JVM arguments to the Ant build process, see JVM Arguments (on page 3117).

Java Virtual Machine (JVM) Crash on macOS

Problem

Oxygen XML Editor crashed the Java virtual machine or it could not start up on my macOS computer due to a JVM crash.

Cause

Usually it is an incompatibility between the JVM and a native library of the host system. More details are available in the crash log file generated by macOS and reported in the crash error message.

Solution

If this happens, it is best to send a copy of the logs via email to support@oxygenxml.com. Usually, the operating system will offer a prompt that allows you to see the logs. If not, you should be able to find the logs in the Console app (Applications > Utilities, under ~/Library/Logs/DiagnosticReports. They are usually named JavaApplicationStub*.crash/.hang.

JPEG CMYK Color Space Issues

Problem

JPEG images with the CMYK color profile and have the color profiles embedded in the image aren't rendered in the Author mode.

Solution

If the color profile information is missing from the JPEG image but you have the ICC file available, you can copy the profileFileName.icc to the \{OXYGEN_INSTALL_DIR\}/lib directory. If the color space profile is missing, JPEG images that have the CMYK color space are rendered without taking the color profile into account. The Unsupported Image Type message is displayed above the image.
Keyboard Language Resets to Default on Windows

Problem
In Windows, I have set a specific language for my keyboard and while using Oxygen XML Editor, it keeps getting reset to the default language.

Cause
When multiple languages are enabled in Windows, the default shortcut key combination for switching the input language is **Left Alt + Shift**. Trying to use various shortcuts in Oxygen XML Editor that involves combinations of those two keys is probably resetting your input language to the default setting if you press those two keys without a third combination.

Solution
You can change the Windows shortcut keys that are assigned to the input language by going to the control panel and searching for the **Switch input languages** option. For example, in Windows 10, go to **Control Panel > Language > Advanced Setting**. In the **Switching input methods** section, click on **Change language bar hot keys**. Click the **Change Key Sequence** button. This opens a dialog box that allows you to switch the shortcut keys for the input language or keyboard layout.

Keyboard Shortcuts Do Not Work At All

Problem
The keyboard shortcuts listed in the **Menu Shortcut Keys preferences page** on page 294 do not work.

Cause
Usually this happens when a special keyboard layout is set (in the operating system) that generates non-standard characters. For example, an extended Greek layout will generate special characters that are not present in the default Greek layout. This causes the Java virtual machine that runs the application to receive unexpected key codes.

Solution
Reset the keyboard layout to the standard layout for your particular language.

Keyboard Shortcuts Result in Unexpected Behavior

Problem
In some rare cases, using certain keyboard shortcuts listed in the **Menu Shortcut Keys preferences page** on page 294 result in something different than what is listed in that preferences page.
**Cause**

This is usually caused by the operating system intercepting the keyboard shortcut. For example, certain applications or hardware drivers intercept certain keyboard shortcuts by default. Another example is if you have multiple input sources configured, the operating system might intercept shortcuts if they match the ones used to change between the input sources.

**Solution**

Assign a different keyboard shortcut for the particular action in the Menu Shortcut Keys preferences page (on page 294) or refer to documentation for your operating system or hardware equipment to see if there are options to change the behavior.

**Mac Touch Bar Function Keys Do Not Work**

**Problem**

I am using a Mac that has a Touch Bar but its function keys do not work in Oxygen XML Editor.

**Causes**

By default, the Touch Bar function keys are not enabled for Oxygen XML Editor.

**Solution**

To enable the Touch Bar function keys for Oxygen XML Editor, follow these steps:

1. Go to System Preferences and select Keyboard.
2. Click Shortcuts.
3. From the left sidebar, select Function Keys.
4. Click the + symbol, select Oxygen from the list of apps, and click Add.

**Keyboard no Longer Works After Inserting Accented Characters on Mac**

**Problem**

The macOS has a method for inserting accented characters that is triggered by holding a key and then selecting the desired character from a contextual menu. When using this method in macOS 10.14 (Mojave) to insert accented characters in Oxygen XML Editor, the keyboard can no longer be used unless I restart the application.

**Causes**

This problem is caused by some sort of incompatibility issue with Java 8 and macOS 10.14 (Mojave).

**Solution**

To prevent this problem, you should use an Oxygen XML Editor distribution that includes OpenJDK 17:
1. Uninstall Oxygen XML Editor:
   a. All data stored in the installation directory will be removed, including any customizations or any other data you have stored within that directory. Therefore, you should make a backup of any data you want to keep.
   b. Remove the application by manually deleting the installation folder and all its contents.

2. Reinstall Oxygen XML Editor using the distribution that includes OpenJDK:
   a. Go to the Oxygen Download page.
   b. In the macOS tab, click on the macOS 10.11 and later (Includes OpenJDK 17) link to download this particular distribution.
   c. Follow the instructions to install the downloaded installation package.

**Server Signature Mismatch Error**

**Problem**

I receive an error indicating that the current license was already activated on a License Server or that the License Server's Signature does not match.

During the license activation process, the license key becomes bound to a particular license server deployment. This means that a code that uniquely identifies your license server deployment (called Server Signature) is sent to the Oxygen XML Editor servers, which in turn will sign the license key. The Server Signature is computed from the list of network interfaces of the server where you deployed the license.

When starting the license server, if you receive an error stating that your Server Signature does not match, there are several possible causes:

**Possible Cause 1**

The license key was moved to a new server that hosts your license server.

**Solution**

Revert to your previous configuration.

**Possible Cause 2**

A new network interface was changed, added, or activated in the server that hosts your license server.

**Note:**

A specific example of when this could happen is if the Bluetooth or the WiFi module is activated/deactivated.

**Solution**

If reverting is not possible, contact the Oxygen support team.
Possible Cause 3

The license server was restarted from a different location as the previous restart. For example, some server configurations will have the Apache Tomcat server installed in a versioned folder (`/usr/local/apache-tomcat-V.V.V`) with a symbolic link to the typical folder (`/usr/local/tomcat`). The server can be restarted from either location, but it is recommended to always restart from the typical folder (`/usr/local/tomcat`) and always restart from the same location.

Solution

The server simply needs to always be restarted from the same location.

MSXML 4.0 Transformation Issues

Problem

When running a transformation scenario that uses the MSXML 4.0 transformer, I receive an error that looks like this:

```
Could not create the 'MSXML2.DOMDocument.4.0' object.
Make sure that MSXML version 4.0 is correctly installed on the machine.
```

Cause

It is likely that the latest MSXML 4.0 service pack is not installed on your computer.

Solution

To fix this issue, go to the Microsoft website and get the latest MSXML 4.0 service pack.

Navigation to a Web Page is Canceled when Viewing CHM on a Network Drive

Problem

When viewing a CHM on a network drive, I only see the TOC and an empty page that displays the message: `Navigation to the web page was canceled`.

Cause

This is actually normal behavior. The Microsoft viewer for CHM does not display the topics for a CHM open on a network drive.

Solution

As a workaround, copy the CHM file on your local system and view it there.
Out Of Memory Error When Opening Large Documents

Problem
I am trying to open a file larger than 100 MB in Oxygen XML Editor, but it runs out of memory (OutOfMemoryError).

Solution
You should make sure that the value of the Optimize loading in the Text edit mode for files over option (on page 200) is less than the size of your document. This will enable the optimized support for large documents. If that fails and you still get an Out Of Memory error you should increase the memory available to Oxygen XML Editor (on page 2856).

Other tips:
- Make sure that you close other files before opening the large file.
- You can set the default editing mode in the Preferences dialog box (on page 171). The Text editing mode uses less memory than other editing modes.
- If the file is too large for the editor to handle, you can open it in for viewing in Large File Viewer (on page 2674).

Problem With Shortcut

Problem
When browsing for files (using the file browser) on Windows with Java 8u202, I get a warning with the title of Problem With Shortcut and the following message:

The drive or network connection that the shortcut 'X.lnk' refers to is unavailable.

Cause
This message is triggered by the file browser from Java 8u202 (which is bundled with Oxygen XML Editor version 21.0 and later). This message usually appears if you have broken shortcuts on your Windows Desktop.

Note:
The message may appear multiple times (once for each broken desktop shortcut).

Solution
The only way to avoid this issue with Java 8u202 is to remove (or move) the broken shortcuts from the Windows Desktop, especially shortcuts that point to locations that you do not have access to (e.g. inaccessible network locations) or shortcuts to files/folders that no longer exist.
Tip:
Alternatively, you can download and install the distribution of Oxygen XML Editor that is bundled with OpenJDK. It is named Windows 64-bit (Includes OpenJDK) and can be found on the Download page for Oxygen XML Editor at http://oxygenxml.com.

Rectangular Selection Feature Does Not Work On Windows

Problem

When trying to use the Rectangular Selection feature (on page 530) on Windows using both the keyboard and mouse, releasing the Alt key shifts the focus to the main menu instead of entering the edit mode.

Cause

This seems to be an issue caused by something that was changed in Java 8u172.

Solution

A workaround is to use Shift or Ctrl along with Alt and release them simultaneously.

References Outside the Main DITA Map Folder

Problem

A reference to a DITA topic, map, or binary resource (for example, an image) that is located outside of the folder where the main DITA map (on page 3218) is located leads to problems when publishing the content using the DITA Open Toolkit.

Cause

DITA-OT often has trouble resolving references that are outside the directory where the published DITA map is found. By default, it does not even copy the referenced topics to the output directory.

Solution

To solve this, try one of the following solutions:

- Create another DITA map that is located in a folder path above all referenced folders and reference the original DITA map from this new map. Then transform this DITA map instead.
- Edit the transformation scenario and in the Parameters tab, change the value of the fix.external.refs.com.oxygenxml parameter to true. This parameter is used to specify whether or not the application tries to fix such references in a temporary files folder before the DITA Open Toolkit is invoked on the fixed references. The fix has no impact on your edited DITA content.
**Important:**
The `fix.external.refs.com.oxygenxml` parameter is only supported when the DITA-OT transformation process is started from Oxygen XML Editor or using the `transform` script.

- For PDF output, you can edit the transformation scenario and in the **Parameters** tab set the value of the `generate.copy.outer` parameter to 3. This parameter specifies whether to generate output files for content that is not located in or beneath the directory containing the DITA map file. By setting the value of this parameter to 3, the transformation scenario shifts the output directory so that it contains all output for the publication.

**Important:**
This method is recommended for transformation scenarios that use an external DITA-OT.

### Saxon 9.7 Transformer Issues

#### Problem

I have upgraded to Oxygen XML Editor version 19.0 (which comes bundled with Saxon 9.7) and I am experiencing issues when trying to use the Saxon 9.7 transformer. Is it possible to use the Saxon 9.6 transformer with Oxygen XML Editor version 19.0 or later?

#### Solution

There is a plugin available that can be installed and it allows you to use Saxon 9.6. To install it, follow these instructions:

1. Go to **Help > Install new add-ons** to open an add-on selection dialog box.
2. Select the `default` update site from the drop-down menu ([https://www.oxygenxml.com/InstData/Addons/default/updateSite.xml](https://www.oxygenxml.com/InstData/Addons/default/updateSite.xml)).
3. Select the **Saxon 9.6 XSLT and XQuery Transformer** plugin and click **Next**.
4. Select the **I accept all terms of the end user license agreement** option and click **Finish**.
5. Restart the application.

**Result:** When you configure the transformation scenario, you will now have the option to choose the Saxon 9.6 transformer.

### Scroll Function of my Notebook Trackpad is Not Working

#### Problem

I got a new notebook (Lenovo Thinkpad™ with Windows) and noticed that the scroll function of my trackpad is not working in Oxygen XML Editor.
Cause

It is most likely a problem with the Synaptics™ trackpads.

Solution

Try adding the following lines to the C:\Program Files\Synaptics\SynTP\TP4table.dat file (depending on your version of Oxygen XML Editor). For example:

```
*,*,oxygen20.1.exe,*,*,WheelStd,1,9
*,*,oxygenAuthor20.1.exe,*,*,WheelStd,1,9
*,*,oxygenDeveloper20.1.exe,*,*,WheelStd,1,9
*,*,syncroSVNClient.exe,*,*,WheelStd,1,9
*,*,diffDirs.exe,*,*,WheelStd,1,9
*,*,diffFiles.exe,*,*,WheelStd,1,9
```

Special Characters are Replaced with a Square

Problem

My file was created with another application and it contains special characters (such as é, ©, ®, etc.) but Oxygen XML Editor displays a square for these characters.

Solution

You must set a font that is able to render the special characters in the Font preferences (on page 133). If it is a text file, you must also set the encoding used for non XML files (on page 168). If you want to set a font that is installed on your computer but that font is not accessible in the Font preferences, this means the Java virtual machine is not able to load the system fonts (probably because it is not a True Type font). It is a problem of the Java virtual machine and a possible solution is to copy the font file in the [JVM_DIR]/lib/fonts folder. [JVM_DIR] is the value of the property java.home that is available in the System properties tab of the About dialog box that is opened from menu Help > About.

TocJS Transformation Does not Generate All Files for a Tree-Like TOC

Problem

The TocJS transformation of a DITA map (on page 3218) does not generate all the files needed to display the tree-like table of contents.

Solution

To get a complete set of output files, follow these steps:

1. Run the XHTML transformation on the same DITA map. Make sure the output gets generated in the same output folder as for the TocJS transformation.
2. Copy the content of the DITA-OT-DIR/plugins/com.sophos.tocjs/basefiles folder to the transformation output folder.
3. Copy the `DITA-OT-DIR/plugins/com.sophos.tocjs/sample/basefiles/frameset.html` file to the transformation output folder.


5. Locate element `<frame name="contentwin" src="concepts/about.html">`.

6. Replace `"concepts/about.html"` with `"index.html"`.

Text Rendering Issues on macOS

**Problem**

On macOS, I sometimes encounter issues where text is not rendered properly. For example, when tags are displayed in **Author** mode, sometimes the tag icon is rendered over the top of text (hiding the text) and sometimes text flows outside of code blocks.

**Cause**

This is an uncommon error that cannot be fixed in current versions.

**Solution**

Open the **Preferences** dialog box (**Options > Preferences**) (on page 124), go to **Editor > Edit modes > Author**, and deselect the **Fast text layout** option (on page 177).

Wrong Words are Highlighted when Searching in a User Manual

**Problem**

When I do a keyword search in the User Manual that is included with the Oxygen XML Editor application, the search highlights the wrong word in the text. Sometimes the highlighted word is several words after the matched word.

**Cause**

This does not happen when Oxygen XML Editor runs with a built-in Java virtual machine that was installed by Oxygen XML Editor in a subfolder of the installation folder (for example, on Windows and Linux when installing Oxygen XML Editor with the installation wizard specific for that platform). However, when Oxygen XML Editor runs from an All Platforms installation, it uses whatever JVM was found on the host system, which may be incompatible with the JavaHelp indexer used for creating the built-in User Manual. Such a JVM may offset the highlight of the matched word with several characters, usually to the right of the matched word.

**Solution**

To see the highlight the exact matched word, it is recommended to install the application with the specific installation wizard for your platform (available only for Windows and Linux).
XML Document Takes a Long Time to Open

Problem

Oxygen XML Editor takes a long time to open an XML document.

Cause

It takes longer to open an XML document if the whole content of your document is on a single line or if the document size is very large.

Solution

If the content is on a single line, you can speed up loading by selecting the **Format and indent the document on open option** *(on page 204)* (in the **Format** preferences page).

If the document is very large (above 30 MB), make sure that the value of the **Optimize loading in the Text edit mode for files over option** *(on page 200)* (in the **Open** preferences page) is greater than the size of your document.

If that fails and you get an Out Of Memory error (**OutOfMemoryError**) you can **increase the memory available to Oxygen XML Editor.** *(on page 2856)*

XSLT Debugger Is Very Slow

Problem

When I run a transformation in the **XSLT Debugger perspective** *(on page 3221)*, it is very slow.

Solution

If the transformation produces HTML or XHTML output, you can **disable rendering of output in the XHTML output view** *(on page 256)* during the transformation process. To view the XHTML output result do one of the following:

- Run the transformation in the **Editor perspective** *(on page 3221)* and make sure the **Open in Browser/System Application option** *(on page 1474)* is selected.
- Run the transformation in the **XSLT Debugger perspective** *(on page 3221)*, save the text output area to a file, and use a browser application for viewing it (for example, Firefox or Chrome).
22.

DITA Authoring

*DITA* is an XML standard, an architectural approach, and a writing methodology, developed by technical communicators for technical communicators. It provides a standardised architectural framework for a common structure for content that promotes the consistent creation, sharing, and re-use of content.

Some of the benefits of using DITA include the following:

- **Flexibility** - DITA is a topic-based architecture and it offers flexibility in content organization.
- **Modularity** - DITA allows for content reuse that saves time and reduces the number of modifications.
- **Structured Authoring** - DITA offers a standardized, methodological approach that helps to reduce authoring time and improve consistency.
- **Single-Source Publishing** - DITA provides the ability to change content in one place and have the change propagate everywhere.
- **Multiple Output Formats** - DITA supports multiple types of output.
- **Inheritance** - The DITA inheritance model makes it easy to specialize topics or elements within topics and you only have to define how the element is different from its immediate ancestor.
- **Process Automation** - DITA offers various ways to automate processes, such as with index or glossary production, output delivery, validation, and more.
- **Specialization** - DITA allows you to define your own information types and semantic elements/attributes to suit the needs of your particular content model.
- **Multi-Lingual** - DITA is a translation-friendly structure that supports numerous languages and text encodings.
- **Conditional Profiling** - DITA supports conditional text processing and profiling to filter content in the publishing stage.

This chapter is designed to be a guide to help content authors who use DITA. It also presents the Oxygen XML Editor features that are specific to working with DITA documents and concepts.

**DITA Resources**

For more general information and technical details about working with DITA, refer to the following resources:

- [The DITA Specifications](#).
- [The DITA Style Guide Best Practices for Authors](#).
- Various sample DITA topics and maps can be found in the `[OXYGEN_INSTALL_DIR]/samples/dita` folder.
- Webinar: [Getting Started with DITA Using Oxygen](#).
Getting Started with DITA

The information in this topic is meant to be a very basic starting point for those who are just getting started using DITA in Oxygen XML Editor. Oxygen XML Editor makes it easy to create, edit, manage, and publish DITA content, but it requires at least some basic DITA knowledge. To truly get the most out of Oxygen XML Editor and all of its DITA-related features, you should explore resources in the online DITA community to acquire knowledge of its concepts and uses.

Understanding DITA Topics

It is important to understand the role that a DITA topic plays in a DITA project. A DITA topic is not associated with a single published document. It is a separate entity that can potentially be included in many different books, help systems, or websites. Therefore, when you write a DITA topic you are not writing a book, a help system, or a website. You are writing an individual piece of content. This affects how you approach the writing task and how Oxygen XML Editor works to support you as you write.

Most of your topics are actually related to other topics, and those relationships can affect how you write and handle things such as links and content reuse. Oxygen XML Editor helps you manage those relationships. Depending on how your topics are related, you can use the tools provided in Oxygen XML Editor, along with the features of DITA, in a variety of ways.

Creating a DITA Topic in Oxygen XML Editor

To create a DITA topic (on page 2961):

1. Select File > New or click the New button on the toolbar.

   Step Result: The New Document Wizard (on page 368) is displayed:
2. Go to **Framework templates** > **DITA** > **topic** and select the type of topic that you want to create.

**Note:**
If your organization has created DITA customizations, the appropriate template files may be in another location, and various types of topics may be provided for your use. Check with the person who manages your DITA system to see if you should be using templates from another directory.

3. Select a file path where it will be saved. You can also optionally specify a title.

4. Click **Create**.

**Result:** Your document is opened in the editor. Eventually, you will need to add a reference to it in your DITA map (on page 2890).

Your DITA topic is an XML document, thus all the editing features that Oxygen XML Editor provides for editing XML documents (on page 34) also apply to DITA topics. Oxygen XML Editor also provides additional specific
DITA-related support for working with DITA topics (on page 2959), their associated DITA maps (on page 2895), and for creating DITA output (on page 3080).

**Role of Maps**

The basic method that DITA uses to express the relationship between topics is through a DITA map (on page 3218). Other relationships between topics, such as cross references, generally need to be made between topics in the same root map. DITA uses maps to determine which topics are part of any output that you create. While customized DITA solutions can use other mechanisms, generally DITA is not used as a way to publish individual topics. Output is created from a map and includes all the topics referenced by the map.

A publication is not always represented by a single map. For instance, if you are writing a book, you might use a submap to create each chapter and then organize the chapters in a main root map to create the book. This helps you to manage your content, offers the possibility of reusing submaps, and segregates content to support multiple people working on the same project.

**Creating a Map in Oxygen XML Editor**

To create a map (on page 2913):

1. Select File > New or click the New button on the toolbar.
2. Go to Framework templates > DITA Map > map and select the type of map you want to create.
3. Choose whether you want to open the map in the Editor or in the DITA Maps Manager (on page 2897). Usually, opening it in the DITA Maps Manager is the best choice. The DITA Maps Manager presents a view of the DITA map that is similar to a table of contents.
Adding Existing Topics to a Map in Oxygen XML Editor

There are several ways to add a topic reference to a map (on page 2917). Perhaps the easiest method is to add a reference to a topic that is already open in the editor:

1. Open the DITA topic in the main editing window.
2. Right-click the DITA map in the DITA Maps Manager view (on page 2897) and choose Reference to the currently edited file from the Append Child, Insert Before, or Insert After submenu.

   **Step Result:** This opens the Insert Reference dialog box (on page 2923) with all of the required fields already filled in for you.

   ![Insert Reference Dialog Box](image)

3. You can fill in additional information in the various tabs in this dialog box or add it to the map later.
4. Select **Insert and close** to add a reference to your topic in the map.
5. Save the DITA map.

Adding New Topics to a Map in Oxygen XML Editor

As you add topics to your map, you may want to create a new topic as a child or sibling of another topic. This is usually done at the map level.

To add a new topic to a map (on page 2917), follow these steps:
1. In the DITA Maps Manager (on page 2897), right-click the node in the current map where you want to add the new topic.

2. Select one of the following actions:
   - Append Child > New - Select this action to insert the new topic as a child of the selected node. This action opens a New file dialog box (on page 2962) that allows you to select the type of document and assists you with naming it. After you have configured your new topic, click Create.
   - Insert Before > New - Select this action to insert the new topic as a sibling to the current node, before it. This action opens a New file dialog box (on page 2962) that allows you to select the type of document and assists you with naming it. After you have configured your new topic, click Create.
   - Insert After > New - Select this action to insert the new topic as a sibling to the current node, after it. This action opens a New file dialog box (on page 2962) that allows you to select the type of document and assists you with naming it. After you have configured your new topic, click Create.
   - Duplicate - Select this action to create a copy of the selected topic and insert it as a sibling. This action opens a dialog box that allows you to choose the file name and location for the newly created copy of the topic. After you have selected the name and path for your new topic, click OK.

   **Note:**
   The value of the root ID is generated taking the Use the file name as the value of the root ID attribute option from the DITA > Topics preferences page (on page 273) into account. When the option is deselected, a unique ID is generated.

   **Step Result:** The new topic is now referenced (as a `<topicref>`) in the DITA map at the location where you inserted it and the new topic is opened in the editor.

3. Save the DITA map.

You can also change the order and nesting of topics in the DITA Maps Manager view by doing either of the following:

- Select the topic to move while holding down the Alt key and use the arrow keys to move it around.
- Use the mouse to drag and drop the topic to the desired location.

The way your parent and child topics are organized in any particular output depends on both the configuration of those topics in the map and the rules of the output transformation that is applied to them. Do not assume that your topics must have the same organization for all output types. The map defines the organization of the topics, not the topics themselves. It is possible to create a variety of maps, each with different organization and configuration options to produce a variety of outputs.
Adding Submaps in Oxygen XML Editor

If you have a large set of information, such as a long book or extensive help system, a single map can become long and difficult to manage. To make it easier to manage, you can break up the content into smaller submaps (on page 2914). A submap might represent a chapter of a book, a section of a user manual, or a page on a website. To build a publication out of these smaller maps, you must add them to a map that represents the overall publication.

To add a child map to the current map (on page 2914):

1. Right-click the parent DITA map in the DITA Maps Manager view (on page 2897) and choose Append child > Map reference.

   **Step Result:** This opens the Insert Reference dialog box (on page 2923) with all of the required fields already filled in for you.

2. You can fill in additional information in the various tabs in this dialog box or add it to the map later.
3. Select **Insert and close** to add a reference to your submap in the main map.
4. Save the main DITA map.

Validating a Map in Oxygen XML Editor

Just as it is with your individual topics, it is important to validate your maps (on page 2941). Oxygen XML Editor provides a validation function for DITA maps that does more than simply validating that the XML is well-formed. It also does the following:

- Validates all of the relationships defined in the maps.
- Validates all of the files that are included in the map.
- Validates all of the links that are expressed in the files.

Validating the map that describes your entire publication validates all the files that make up the publication and all of the relationships between them.

To validate a map:

1. Click the Validate and Check for Completeness button in the DITA Maps Manager view (on page 2897).

   **Step Result:** This opens the DITA Map Completeness Check dialog box (on page 2942).

2. Select any of the various options you want to check.
3. Click **Check** to run the validation process.

Publishing Your Topics in Oxygen XML Editor

As noted previously, in DITA standards you usually do not publish output from an individual topic. Instead, you create published output (on page 3080) by running a DITA transformation on a map. This collects all the topics that are referenced in the map, organizes them, and produces output in a particular format. By
default, Oxygen XML Editor uses the transformations provided by the DITA Open Toolkit for publishing to various output formats (such as PDF, WebHelp or EPUB). Your organization may have created various custom transformations or modified the built-in DITA Open Toolkit transformations. In either case, Oxygen XML Editor manages them by using transformation scenarios.

To publish output for a map:

1. Click the Configure Transformation Scenario(s) button in the DITA Maps Manager view (on page 2897).

   **Step Result:** This opens the Configure Transformation Scenario(s) dialog box (on page 1548).

   ![Figure 733. Configure Transformation Scenarios Dialog Box](image)

2. Select the appropriate transformation depending on the type of output you desire.

3. To change or view the configuration or storage options for a transformation scenario, select the transformation and click Edit.

4. Click Apply associated.

   **Result:** Depending on the configuration of the transformation scenario, when the transformation is finished, your output may automatically be opened in the appropriate application.
DITA Projects

Once you have a basic understanding of DITA and how to work with DITA topics and maps, you probably want to create a DITA project to organize and manage your planned content/resources (on page 399). Oxygen XML Editor includes a Project view (on page 402) that helps you organize your projects and offers a variety of helpful project-related features and makes it easy to share your projects with other members of your team.

Tip:
There are several sample project templates available for DITA users that can be used as a starting point or for inspiration. These sample project templates are found in the Framework templates > DITA folder in the New Project wizard: (on page 399)

- **Sample DITA Project** - This is a basic DITA project meant to help new users see how a DITA project is structured.
- **Startup DITA Project** - This is a startup DITA project that imposes a custom set of options (e.g. spell check settings and custom dictionaries), a custom DITA framework extension (e.g. custom new file templates. custom actions, custom CSS used for visual editing) and a folder structure for a DITA project according to best practices. Once created, the project contains a Readme.html file that explains all customizations and their benefits. If you plan to start your own DITA project using a version control system (such as Git), you can use this startup DITA project template to customize various aspects of DITA editing and share them with your team.

Resources

For more information about getting started with DITA and how to work with DITA in Oxygen XML Editor, see our compiled collection of DITA-related webinars that are meant to help you with your journey into working with DITA: Webinars: Working with DITA in Oxygen.

Related information
DITA Authoring (on page 2886)
Editing XML Documents in Author Mode (on page 588)
https://www.oxygenxml.com/dita/1.3/specs/
Webinars: Working with DITA in Oxygen

Working with Projects in DITA

Oxygen XML Editor provides the ability to organize your DITA resources in projects, the same as with other XML-related files. This helps you manage and organize your files and projects allow you to perform batch operations (such as validation and transformation) over multiple files or to use Main Files support to rename or move DITA resources (on page 3181) while updating the references to them. You can also share your project settings and transformation/validation scenarios (on page 415) with other users.
To learn how to create a new project from a template and how to add resources and manage it, see Creating a New Project (on page 399).

To help you get more familiar with how to use projects in DITA, there are two DITA-specific sample project templates available when using the New Project action (available, for example, from the Project menu):

- **DITA Project With Editing Customizations** - This sample DITA project imposes custom general settings and an editing behavior using a DITA framework extension. More details can be found in the project's Readme.html file.
- **Sample DITA Project** - This sample DITA project is a best practice example that shows how DITA content can be organized to provide a scalable and flexible project structure. More details can be found in the project's Readme.html file.

**Resources**

For more information about working with DITA projects, see our webinar: Working with DITA in Oxygen - Quick Start with the DITA Startup Project.

**Related information**

Using Projects to Group Documents (on page 398)

---

**Working with DITA Maps**

In the DITA standard architecture you create documents by collecting topics into maps.

**DITA Maps**

A **DITA map (on page 3218)** organizes a set of topics into a hierarchy. In most output formats, the structure of the map becomes the structure of the table of contents. Oxygen XML Editor provides support for creating (on page 2913) and managing **DITA maps (on page 2916)** through the DITA Maps Manager (on page 2897). There are also specialized types of **DITA maps**, such as a **bookmap (on page 3216)**, which is intended for creating the structure of a book.

**Submaps**

You do not have to create an entire publication using a single map. It is generally good practice to break up a large publication into several smaller **submaps (on page 2914)** that are easier to manage. You can reuse submaps in multiple publications by including them in each of the main maps. The DITA Maps Manager (on page 2897) provides support for easily creating and managing submaps.

**Opening a DITA Map**

There are several ways to open a **DITA map** and you can choose to open it in the DITA Maps Manager (on page 2897) or in the XML Editor. Use any of the following methods to open a map:
To open a submap in its own tab in the DITA Maps Manager, simply double-click it (or right-click it and select Open).

To open a map in the XML editor from the DITA Maps Manager, right-click it and select Open Map in Editor.

Drag a DITA map file from your system browser and drop it in the XML editor. This will open the map in the editor.

If you open a file with the .ditamap or .bookmap extension (from the Project view (on page 402) or a system browser), a dialog box is opened that offers you the choice of opening it in the XML editor or in the DITA Maps Manager.

Note:
If you select the Do not show the dialog again option, it will always be opened in the method that you choose and you will not be asked in the future. However, you can reset this by selecting Always ask for the When opening a map option in the DITA preferences page (on page 270).

To open a map in the DITA Maps Manager, you can right-click a map file in the Project view (on page 402) and select Open with > DITA Maps Manager.

If you have a DITA map file open in the XML editor, you can open it in the DITA Maps Manager by right-clicking the title tab and selecting Open in DITA Maps Manager View.

**Chunking DITA Maps**

By default, many output types place a single topic on each output page. In some cases you may want to output multiple topics as a single output page (also known as chunking) (on page 2941). To support this, Oxygen XML Editor provides an Edit Properties dialog box (on page 2932) that allows you to easily configure the attributes of a topic to control how your table of contents and topics are rendered in the output.

**Validating a Map**

You should validate your maps (on page 2941) to make sure that the individual topics are valid and that the relationships between them are working. Oxygen XML Editor provides a validation function for DITA maps that performs a comprehensive validation of a map and its topics.

**Resources**

For more information about getting started with DITA and how to work with DITA in Oxygen XML Editor, see our compiled collection of DITA-related webinars that are meant to help you with your journey into working with DITA: Webinars: Working with DITA in Oxygen.

**Related information**

- DITA Map Document Type (Framework) (on page 1361)
- DITA Map Author Mode Actions (on page 2947)
DITA Maps Manager

Oxygen XML Editor provides a view for managing and editing DITA maps. The DITA Maps Manager view presents a DITA map as a tree or table of contents. It allows you to navigate the topics and maps, make changes, and apply transformation scenarios to obtain various output formats. By default, it is located to the left of the main editor. If the view is not displayed, it can be opened by selecting it from the Window > Show View menu.

The DITA Maps Manager includes a variety of useful actions to help you edit and organize the structure of your DITA maps and topics. The actions that are available and their functions depend on the type of nodes that are selected in the DITA Maps Manager. If you select multiple sibling nodes, the result of the actions will be applied to all the selected nodes. If you select multiple nodes that are not on the same hierarchical level, the actions will be applied to the parent node and the child nodes will inherit certain attributes from the parent node.

Figure 734. DITA Maps Manager View

Each node is displayed with an icon representing the type of DITA resource. For example, a DITA Map is displayed with an icon, a DITA Topic is displayed with , a DITA Task is displayed with , etc. Any node that has processing-role="resource-only" set in its properties is displayed with a gray dot in the bottom-right corner of the icon ( ).

Opening Maps in the DITA Maps Manager

The DITA Maps Manager view supports opening multiple maps at the same time, with each one presented in its own tab. To open a DITA map in the DITA Maps Manager, use any of the following methods:

- To open a submap in its own tab, simply double-click it (or right-click it and select Open).
- If you open a file with a .ditamap or .bookmap extension (from the Project view (on page 402) or a system browser), a dialog box is opened that offers you the choice of opening it in the DITA Maps Manager or the XML editor.
Note:
If you select the Do not show the dialog again option, it will always be opened in the method that you choose and you will not be asked in the future. However, you can reset this by selecting the Always ask choice for the When opening a map option in the DITA preferences page (on page 270).

- Right-click a map file in the Project view (on page 402) and select Open with > DITA Maps Manager.
- If you have a DITA map file open in the XML editor, you can right-click the title tab and select Open in DITA Maps Manager View.

By default, when a map is opened in the DITA Maps Manager, its index is automatically refreshed. You can disable this feature by deselecting the Refresh index when opening a map in DITA Maps Manager option (on page 298) in the Open/Find Resource preferences page.

Submap Nodes

If your root map (on page 3223) (main DITA map) references other maps (submaps), they can be expanded and you can navigate their content in the DITA Maps Manager, but the references within those submaps are not editable, by default, unless you open the submap separately in its own tab. The references within submap nodes are shown with a gray background.

Tip:
If you want to be able to edit submaps when the main (root/parent) map is open in the DITA Maps Manager, go to Options > Preferences > DITA > Maps and select the Allow referenced submaps to be edited option (on page 271).

Moving Nodes in the DITA Maps Manager

You can move topics or nodes within the same map, or other maps, by dragging and dropping them into the desired position. You can arrange the nodes by dragging and dropping one or more nodes at a time. You can arrange multiple topics by dragging them while pressing the Ctrl or Shift key. Drop operations can be performed before, after, or as child of the targeted node.

Operations include:

**Copy**

Select the nodes you want to copy and start dragging them. Before dropping them in the appropriate place, press and hold the Ctrl key. The mouse pointer changes to a symbol to indicate that a copy operation is being performed.

**Move**
Select the nodes you want to move and drag and drop them in the appropriate place.

**Promote (Alt + LeftArrow) / Demote (Alt + RightArrow)**

You can move nodes between child and parent nodes by using the **Promote (Alt + LeftArrow)** and **Demote (Alt + RightArrow)** operations.

### DITA Maps Manager Toolbar

The toolbar includes the following actions (also available in the DITA Maps menu) and their availability depend on the nodes that are selected:

**Note:**

If multiple nodes are selected, the availability of the actions depends on the nodes that are selected.

- **New DITA Map**
  
  Opens the New Document wizard (on page 368) that you can use to create a new DITA map document.

- **Open Drop-down Menu**
  
  You can use this drop-down menu to open new DITA maps or to reopen recently viewed maps. The drop-down menu contains the following:

  - List of recently viewed DITA maps that can be selected to reopen them.
  - Clear history - Clears the history list of the recently viewed DITA maps.
  - Open - Allows you to open the map in the DITA Maps Manager view (on page 2897). You can also open a map by dragging it from the file system explorer and dropping it into the DITA Maps Manager view (on page 2897).
  - Open URL - Displays the Open URL dialog box where you can specify a URL (defined by a protocol, host, resource path, and an optional port) or use the browsing actions in the Browse for remote file drop-down menu.

- **Save (Ctrl + S (Meta + S on macOS))**
  
  Saves the current DITA map.

- **Validation drop-down menu**
  
  This drop-down menu contains options for validating the current map. The following options are available:

  - **Validate and Check for Completeness**
    
    Opens the DITA Map Completeness Check dialog box where you can configure options for checking the validity and integrity (on page 2941) of the map.

  - **Validate**
Validates the current map that is open in the DITA Maps Manager using the associated validation scenarios.

Configure Validation Scenario(s)

Use this option to configure validation scenarios and their associations (on page 798).

Apply Transformation Scenario(s)

Applies the DITA Map transformation scenario (on page 1433) that is associated with the current map.

Configure Transformation Scenario(s)

Opens the Configure Transformation Scenarios(s) dialog box (on page 1548) where you can edit or create transformation scenarios or associate a DITA Map transformation scenario (on page 1490) with the current map.

Open Map in Editor with Resolved Topics

Opens the DITA map in the main editor area with content from all topic references expanded in-place. Referenced content is presented as read-only by default. To edit it, you must use the Edit Reference contextual menu action to open the source topic that contains the referenced content.

If you want to edit the referenced topics directly without having to open the source document, go to Options > Preferences > Editor > Edit Modes > Author and select the Allow referenced content to be edited option (on page 179). Since a single topic may be referenced in multiple places in the DITA map, be careful not to make conflicting changes to that topic.

Tip:

If you want to print the expanded content, you should consider changing selecting + Print ready from the Styles drop-down menu on the toolbar.

Open Map in Editor

For complex operations that cannot be performed in the simplified DITA Maps Manager view (for instance, editing a relationship table) you can open the map in the main editing area.

Note:

You can also use this action to open referenced DITA maps in the Editor.

Link with Editor

Toggles the synchronization between the file path of the current editor and the selected topic reference in the DITA Maps Manager view. If enabled, it results in the following types of synchronizations:
• If you select a topic tab in the main editing area and it is referenced in the map currently opened in the DITA Maps Manager, the reference to that topic is selected in the DITA Maps Manager.

• If you have a map opened in both the DITA Maps Manager and the main editor, selecting the map tab in the main editing area opens that map in the DITA Maps Manager.

• If you have a map opened in both the DITA Maps Manager and the main editor (Author mode) and, selecting one or more topicrefs in the DITA Maps Manager will also select the same topicrefs in the main editor.

• If you have a map opened in both the DITA Maps Manager and the main editor (Author mode) and, selecting one or more topicrefs in the main editor will also select the same topicrefs in the DITA Maps Manager.

**Settings**

**Show extended toolbar**

Toggles whether or not the extended toolbar will be displayed in the DITA Maps Manager toolbar.

**Show context toolbar**

Toggles whether or not the Context option (on page 2901) will be displayed in the DITA Maps Manager toolbar.

**Show topic titles**

Toggles how topics are presented in the DITA Maps Manager. If selected, the title of each topic is shown. Otherwise, the file path (value of the @href attribute) for each topic is shown.

**Show key reference values**

Toggles how key references are presented in the DITA Maps Manager. If selected, the value of the @keyref attribute for each key reference is shown.

**Root Map Drop-down menu**

The drop-down menu displayed after Context can be used to specify the DITA root map (on page 3223) that Oxygen XML Editor uses to define a hierarchical structure of submaps and to establish a key space (on page 3220) that defines the keys that are propagated throughout the entire map structure. For more information, see Selecting a Root Map (on page 2914).

**Choose Root Map Drop-down menu**

You can use this drop-down menu to browse for root maps with the following choices:

- **Browse for local file** - Opens a local file browser dialog box, allowing you to select a local root map.

- **Browse for remote file** - Displays the Open URL dialog box (on page 387) that allows you to select a remotely stored root map.
- **Browse for archived file** - Displays the Archive Browser (on page 2008) that allows you to browse the content of an archive and choose a root map.

- **Browse Data Source Explorer** - Opens the Data Source Explorer (on page 2015) that allows you to browse the data sources defined in the Data Sources preferences page (on page 276).

Tip:
You can open the Data Sources preferences page by using the Configure Database Sources shortcut from the Open URL dialog box.

- **Search for file** - Displays the Find Resource dialog box (on page 426) to search for a root map.

### Profiling/Conditional Text Drop-down Menu
You can use this drop-down menu to select and apply a defined profiling condition set (on page 3142) to filter the content based on that condition set. The drop-down menu also contains the following other options:

- **Show Profiling Colors and Styles** - Select this option to turn on conditional styling. To configure the colors and styles open the Preferences dialog box (Options > Preferences) (on page 124) and go to Editor > Edit modes > Author > Profiling/Conditional Text > Colors and Styles.

- **Show Profiling Attributes** - Select this option to display the values of the profiling attributes at the end of the titles of topic references. When selected, the values of the profiling attributes are displayed in both the DITA Maps Manager view and in the Author view.

- **Show Excluded Content** - Controls if the content filtered out by a particular condition set is hidden or grayed-out in the editor area and in the Outline (on page 539) and DITA Maps Manager views. When this option is selected, the content filtered by the currently applied condition set is grayed-out. To show only the content that matches the currently applied condition set, deselect this option.

- **Profiling Settings** - Opens the preferences page for adding and editing the profiling conditions that you can apply in the DITA Maps Manager view and the Author mode editing pane. When a profiling condition set (on page 675) is applied, the keys that are defined in the DITA map are gathered by filtering out the excluded content.

The following additional actions are displayed in the toolbar when the Show extended toolbar option is selected in the Settings menu:

#### Insert Topic Reference
Opens the Insert Reference dialog box (on page 2923) that allows you to insert references to targets such as topics, maps, topic sets, or key definitions.
**Refresh References**

You can use this action to manually trigger a refresh and update of all referenced documents. This action is useful when the referenced documents are modified externally. When they are modified and saved from Oxygen XML Editor, the DITA map is updated automatically.

**Edit Properties**

Opens the Edit Properties dialog box that allows you to configure the properties of a selected node. For more details about this dialog box, see Edit Properties Dialog Box (on page 2932).

**Edit Attributes**

Opens a small in-place editor that allows you to edit the attributes of a selected node. You can find more details about this action in the Attributes View in Author Mode (on page 627) topic.

**Delete**

Deletes the selected node.

**Move Up**

Moves the selected node up within the DITA map tree.

**Move Down**

Moves the selected node down within the DITA map tree.

**Promote (Alt + LeftArrow)**

Moves the selected node up one level to the level of its parent node.

**Demote (Alt + RightArrow)**

Moves the selected node down one level to the level of its child nodes.

### Contextual Menu of the DITA Maps Manager

**Root Map**

The following actions can be invoked from the contextual menu on the root map (on page 3223) of an opened DITA map (many of them are also available in the DITA Maps menu):

**Open Map in Editor**

For complex operations that cannot be performed in the simplified DITA Maps Manager view (for instance, editing a relationship table) you can open the map in the main editing area.

**Open Map in Editor with Resolved Topics**

Opens the DITA map in the main editor area with content from all topic references, expanded in-place. Content from the referenced topics is presented as read-only and you have to use the contextual menu action Edit Reference to open the topic for editing.
Export DITA Map

Opens a dialog box that allows you to choose a destination for exporting the DITA map. It also includes an Export as Zip archive option that allows you to package the DITA map as a zip archive. The result will contain all directly and indirectly referenced topics from the DITA Map.

Find Unreferenced Resources

Allows you to search for orphaned resources that are not referenced in the DITA maps.

Add to Review Task

This action can be used to add the selected documents to a task in the Content Fusion Tasks Manager view. Oxygen Content Fusion is a flexible, intuitive collaboration platform designed to adapt to any type of documentation review workflow. This functionality is available through a pre-installed connector add-on (on page 2527). To fully take advantage of all of the benefits and features of Content Fusion, your organization will need an Oxygen Content Fusion Enterprise Server. For more information, see the Oxygen Content Fusion website.

Show Feedback Comments Manager

Opens the Feedback Comments Manager view. This view is for those who use Oxygen Feedback to provide a commenting component in WebHelp output. This view makes it possible to see all the comments added by users in WebHelp output directly in Oxygen XML Editor.

Edit Properties

Opens the Edit Properties dialog box that allows you to configure the properties of a selected node. For more details about this dialog box, see Edit Properties Dialog Box (on page 2932).

Fast Create Topics

Opens the Fast Create Topics dialog box (on page 2964) that allows you to quickly create multiple skeleton topics at once and you can specify their hierarchical structure within the DITA map (on page 3218).

Append Child submenu

Container sub-menu for a number of actions that create a map node as a child of the currently selected node:

- **New** - Opens a dialog box that allows you to configure some options for inserting a new topic (on page 2961).
- **Reference** - Inserts a reference to a topic file. You can find more details about this action in the Inserting References (on page 2923) topic.
• **Reference to the currently edited file** - Inserts a reference to the currently edited file. You can find more details about this action in the [Inserting References](on page 2923) topic.

• **Key Reference** - Opens an Insert Key Definition dialog box that allows you to insert a targeted key definition ([on page 2932]) (for example, to target a resource such as an image or external link).

• **Key Reference with Keyword** - Opens an Insert Key Definition dialog box that allows you to define a key and a value inside a keyword ([on page 2931]).

• A set of actions that open the Insert Reference dialog box ([on page 2923]) that allow you to insert various reference specializations (such as Anchor Reference, Glossary Reference, Map Reference, Navigation Reference, Topic Group, Topic Head, Topic Reference, Topic Set, Topic Set Reference).

**Search References**

Searches all references to the current topic in the entire DITA map ([on page 3218]). It also reports references that are defined as related links in relationship tables. If you have enabled Main Files support ([on page 3181]), it also searches for references in the DITA maps added to the Main Files folder.

**Refactoring submenu**

The following actions are available from this submenu when invokes from a root map:

**Rename resource**

Allows you to change the name of a resource linked in the edited DITA map ([on page 2919]) and you have the option of updating all the references to the renamed DITA resource. If you have enabled Main Files support ([on page 3181]), it also searches for references in the DITA maps added to the Main Files folder and it provides the option of updating all the references even for non-DITA resources.

**Move resource**

Allows you to change the location on disk of a resource linked in the edited DITA map ([on page 2919]) and you have the option of updating all the references to the moved DITA resources. If you have enabled Main Files support ([on page 3181]), it also searches for references in the DITA maps added to the Main Files folder and it provides the option of updating all the references even for non-DITA resources.

**Rename Key**
Use this operation to rename a key. It also updates all references to it. Note that it does not work on DITA 1.3 key scopes.

**Convert to Concept**

Use this operation to convert a DITA topic (of any type) to a DITA Concept topic type (for example, Topic to Concept).

**Convert to General Task**

Use this operation to convert a DITA topic (of any type) to a DITA General Task topic type (for example, Task to General Task). A DITA General Task is a less restrictive alternative to the Strict Task information type.

**Convert to Reference**

Use this operation to convert a DITA topic (of any type) to a DITA Reference topic type (for example, Topic to Reference).

**Convert to Task**

Use this operation to convert a DITA topic (of any type) to a DITA Task topic type (for example, Topic to Task).

**Convert to Topic**

Use this operation to convert a DITA topic (of any type) to a DITA Topic (for example, Task to Topic).

**Convert to Troubleshooting**

Use this operation to convert a DITA topic (of any type) to a DITA Troubleshooting topic type (for example, Topic to Troubleshooting).

**Generate IDs**

Use this operation to automatically generate unique IDs for elements.

**Other XML Refactoring Actions**

For your convenience, the last 5 XML Refactoring tool operations (on page 840) that were finished or previewed will also appear in this submenu.

**XML Refactoring**

Opens the XML Refactoring tool wizard (on page 840) that presents refactoring operations to assist you with managing the structure of your XML documents.

**Find/Replace in Files**

Opens the Find/Replace in Files dialog box (on page 436) that allows you to find and replace content across multiple files.

**Check Spelling in Files**
Allows you to spell check multiple files (on page 459).

**Paste**

Allows you to paste content from the clipboard into the *DITA map*.

**Paste Before**

Pastes the content of the clipboard (only if it is a part of the *DITA map*) before the currently selected *DITA map* node.

**Paste After**

Pastes the content of the clipboard (only if it is a part of the *DITA map*) after the currently selected *DITA map* node.

**Expand All**

Allows you to expand the entire *DITA map* structure.

**Collapse All**

Allows you to collapse the entire *DITA map* structure.

**Editable Child Nodes**

The following actions are available when the contextual menu is invoked on an editable child node of a *DITA map* (submaps need to be opened in the *DITA Maps Manager* to access these actions since they are in a read-only state in the parent map):

**Note:**

If multiple nodes are selected, the availability of the actions depends on the nodes that are selected.

**Open**

Opens the selected resource in the editor.

**Add to Review Task**

This action can be used to add the selected documents to a task in the *Content Fusion Tasks Manager* view. *Oxygen Content Fusion* is a flexible, intuitive collaboration platform designed to adapt to any type of documentation review workflow. This functionality is available through a pre-installed connector add-on (on page 2527). To fully take advantage of all of the benefits and features of *Content Fusion*, your organization will need an *Oxygen Content Fusion Enterprise Server*. For more information, see the *Oxygen Content Fusion* website.

**Edit Properties**

Opens the *Edit Properties* dialog box that allows you to configure the properties of a selected node. For more details about this dialog box, see *Edit Properties Dialog Box* (on page 2932).
Fast Create Topics

Opens the Fast Create Topics dialog box (on page 2964) that allows you to quickly create multiple skeleton topics at once and you can specify their hierarchical structure within the DITA map (on page 3218).

Append Child submenu

Container sub-menu for a number of actions that create a map node as a child of the currently selected node:

- **New** - Opens a dialog box that allows you to configure some options for inserting a new topic (on page 2961).
- **Reference** - Inserts a reference to a topic file. You can find more details about this action in the Inserting References (on page 2923) topic.
- **Reference to the currently edited file** - Inserts a reference to the currently edited file. You can find more details about this action in the Inserting References (on page 2923) topic.
- **Key Reference** - Opens an Insert Key Definition dialog box that allows you to insert a targeted key definition (on page 2932) (for example, to target a resource such as an image or external link).
- **Key Reference with Keyword** - Opens an Insert Key Definition dialog box that allows you to define a key and a value inside a keyword (on page 2931).
- A set of actions that open the Insert Reference dialog box (on page 2923) that allow you to insert various reference specializations (such as Anchor Reference, Glossary Reference, Map Reference, Navigation Reference, Topic Group, Topic Head, Topic Reference, Topic Set, Topic Set Reference).

Insert Before submenu

Container sub-menus for a number of actions that create a map node as a sibling of the currently selected node, above the current node in the map:

- **New** - Opens a dialog box that allows you to configure some options for inserting a new topic (on page 2961).
- **Reference** - Inserts a reference to a topic file. You can find more details about this action in the Inserting References (on page 2923) topic.
- **Reference to the currently edited file** - Inserts a reference to the currently edited file. You can find more details about this action in the Inserting References (on page 2923) topic.
- **Key Reference** - Opens an Insert Key Definition dialog box that allows you to insert a targeted key definition (on page 2932) (for example, to target a resource such as an image or external link).
• **Key Reference with Keyword** - Opens an Insert Key Definition dialog box that allows you to define a key and a value inside a keyword (on page 2931).

• A set of actions that open the **Insert Reference** dialog box (on page 2923) that allow you to insert various reference specializations (such as Anchor Reference, Glossary Reference, Map Reference, Navigation Reference, Topic Group, Topic Head, Topic Reference, Topic Set, Topic Set Reference).

### Insert After submenu

Container sub-menus for a number of actions that create a map node as a sibling of the currently selected node, below the current node in the map:

• **New** - Opens a dialog box that allows you to configure some options for inserting a new topic (on page 2961).

• **Reference** - Inserts a reference to a topic file. You can find more details about this action in the Inserting References (on page 2923) topic.

• **Reference to the currently edited file** - Inserts a reference to the currently edited file. You can find more details about this action in the Inserting References (on page 2923) topic.

• **Key Reference** - Opens an Insert Key Definition dialog box that allows you to insert a targeted key definition (on page 2932) (for example, to target a resource such as an image or external link).

• **Key Reference with Keyword** - Opens an Insert Key Definition dialog box that allows you to define a key and a value inside a keyword (on page 2931).

• A set of actions that open the **Insert Reference** dialog box (on page 2923) that allow you to insert various reference specializations (such as Anchor Reference, Glossary Reference, Map Reference, Navigation Reference, Topic Group, Topic Head, Topic Reference, Topic Set, Topic Set Reference).

### Search References

Searches all references to the current topic in the entire DITA map (on page 3218). It also reports references that are defined as related links in relationship tables. If you have enabled Main Files support (on page 3181), it also searches for references in the DITA maps added to the Main Files folder.

### Refactoring submenu

The following actions are available from this submenu:

**Convert Markdown to DITA Topic** (Available for Markdown documents)
Opens a dialog box that allows you to configure options for converting the Markdown document into a DITA topic (on page 3025).

**Rename resource**

Allows you to change the name of a resource linked in the edited DITA map (on page 2919) and you have the option of updating all the references to the renamed DITA resource. If you have enabled Main Files support (on page 3181), it also searches for references in the DITA maps added to the Main Files folder and it provides the option of updating all the references even for non-DITA resources.

**Move resource**

Allows you to change the location on disk of a resource linked in the edited DITA map (on page 2919) and you have the option of updating all the references to the moved DITA resources. If you have enabled Main Files support (on page 3181), it also searches for references in the DITA maps added to the Main Files folder and it provides the option of updating all the references even for non-DITA resources.

**Extract to New DITA Map**

Use this operation to extract editable topics into a new DITA map. The operation will open a map creation dialog box where you can select the type of map and configure the title or file name. Click Create to complete the operation and a new DITA map will be inserted at the location where the action was invoked with the selected topic references moved into the new map.

**Rename Key**

Use this operation to rename a key. It also updates all references to it. Note that it does not work on DITA 1.3 key scopes.

**Convert Nested Topics to New Topics (Available from the contextual menu of editable maps/nodes in the DITA Maps Manager (on page 2897))**

Use this operation on topics that contain nested <topic> elements to convert each nested topic to a new topic. Also, the new topics are added in the DITA Maps Manager as the first child topics of the original topic.

**Convert Sections to New Topics (Available from the contextual menu of editable maps/nodes in the DITA Maps Manager (on page 2897))**

Use this operation on topics that contain multiple sections to convert each section to a new topic. Also, the new topics are added in the DITA Maps Manager as the first child topics of the original topic.
Convert to Concept
Use this operation to convert a DITA topic (of any type) to a DITA Concept topic type (for example, Topic to Concept).

Convert to General Task
Use this operation to convert a DITA topic (of any type) to a DITA General Task topic type (for example, Task to General Task). A DITA General Task is a less restrictive alternative to the Strict Task information type.

Convert to Reference
Use this operation to convert a DITA topic (of any type) to a DITA Reference topic type (for example, Topic to Reference).

Convert to Task
Use this operation to convert a DITA topic (of any type) to a DITA Task topic type (for example, Topic to Task).

Convert to Topic
Use this operation to convert a DITA topic (of any type) to a DITA Topic (for example, Task to Topic).

Convert to Troubleshooting
Use this operation to convert a DITA topic (of any type) to a DITA Troubleshooting topic type (for example, Topic to Troubleshooting).

Generate IDs
Use this operation to automatically generate unique IDs for elements.

Other XML Refactoring Actions
For your convenience, the last 5 XML Refactoring tool operations (on page 840) that were finished or previewed will also appear in this submenu.

XML Refactoring
 Opens the XML Refactoring tool wizard (on page 840) that presents refactoring operations to assist you with managing the structure of your XML documents.

Find/Replace in Files
Opens the Find/Replace in Files dialog box (on page 436) that allows you to find and replace content across multiple files.

Check Spelling in Files
Allows you to spell check multiple files (on page 459).
Cut

Deletes the currently selected node and copies it to the clipboard.

Copy

Copies the currently selected node to the clipboard.

Paste

Allows you to paste content from the clipboard into the DITA map.

Paste Before

Pastes the content of the clipboard (only if it is a part of the DITA map) before the currently selected DITA map node.

Paste After

Pastes the content of the clipboard (only if it is a part of the DITA map) after the currently selected DITA map node.

Delete

Deletes the currently selected node from the DITA map.

Remove from Disk

This action can be used to remove the selected resource(s) from disk. Selecting this action will open a confirmation dialog box where you can also choose to remove the descendants by selecting the Also remove all descendants option. If you proceed, a search for references is triggered. If multiple references are detected for any of the selected resources, you will have the option to review them since this would lead to broken links. If you have enabled Main Files support (on page 3181), it also searches for references in the DITA maps added to the Main Files folder.

Organize

Allows you to organize the DITA map with the several submenu actions:

- Move Up - Moves the selected node up within the DITA map tree.
- Move Down - Moves the selected node down within the DITA map tree.
- Promote (Alt + LeftArrow) - Moves the selected node up one level to the level of its parent node.
- Demote (Alt + RightArrow) - Moves the selected node down one level to the level of its child nodes.

Expand All

Allows you to expand the entire DITA map structure.

Collapse All
Allows you to collapse the entire DITA map structure.

Other Nodes

The following additional actions are available when the contextual menu is invoked from other nodes, such as a submap node or a relationship table:

Open Map in Editor (available when invoking on a submap)

Opens the currently selected DITA map in the editor.

Open parent DITA map (available when invoking on a read-only topic reference or a submap reference)

Opens the parent DITA map of the currently selected reference in the DITA Maps Manager.

Edit Attributes (only available for relationship table nodes)

Opens a small in-place editor that allows you to edit the attributes of a selected node. You can find more details about this action in the Attributes View in Author Mode (on page 627) topic.

Edit Profiling Attributes (only available for relationship table nodes)

Allows you to change the profiling attributes (on page 668) defined on the selected node.

For more information about the DITA Maps Manager view and many of its features, watch our video demonstration:

https://www.youtube.com/embed/ozFZz6YZMCI

Related information

- DITA Map Validation and Completeness Check (on page 2941)
- DITA Map Author Mode Actions (on page 2947)
- Find/Replace in Multiple Files (on page 436)

Creating a Map

To create a DITA map (on page 3218), subject scheme map (on page 3224), bookmap (on page 3216), or other types of DITA maps, follow these steps:

1. Use the New Document wizard (on page 368) to start creating your map.

Tip:

If you want the map to be a submap, you can create it the same way by right-clicking the place in the current map where you want to add it (in the DITA Maps Manager (on page 2897)) and selecting New from the Append Child, Insert Before, or Insert After submenu.
2. Select one of the DITA Map templates from the Framework templates folder.
3. Click the Create button.
4. Select whether you want to open the map in the DITA Maps Manager (on page 2897) or the Editor.
5. Save the map using the Save button on the toolbar of the DITA Maps Manager view (on page 2897).

Related Information:
Customizing Profiling Values with a Subject Scheme Map (on page 3151)
Managing DITA Maps (on page 2916)

Selecting a Root Map

Oxygen XML Editor allows you to select a root map (on page 3223) (a main DITA map (on page 3218)) that defines a hierarchical structure of submaps and establishes a key space (on page 3220) that defines the keys used in all the other DITA maps and topics in the project. Specifying the correct root map helps to prevent validation problems when you work with keyrefs and also acts as the foundation for content completion. All the keys that are defined in a root map are available in the submaps that are contained within the root map.

There are several ways to select or change the root map:

- The easiest method is to use the Context drop-down menu (on page 2901) in the DITA Maps Manager (on page 2897) toolbar to select the appropriate root map.
- If you insert a key reference using the Cross Reference action from the Link drop-down menu (from the toolbar or Link submenu of the contextual menu) and keys are not gathered from the expected DITA map, you can change the root map by using the Change Root Map link in the Choose Key dialog box that is opened when you click the Choose Key Reference button.
- If you insert a content key reference or key reference using the Reuse Content action (from the toolbar, DITA menu, or Reuse submenu of the contextual menu) and keys are not gathered from the expected DITA map, you can change the root map by using the Change Root Map link in the Choose Key dialog box that is opened when you click the Choose Key Reference button.

Creating DITA Submaps

You can break up a large DITA map (on page 3218) into more manageable pieces by creating submaps. A submap is simply a DITA map that is included by another DITA map. There is no separate markup for a submap.

For example, if you are creating a book, you might use one submap for each chapter of the book. If you are reusing a set of topics in multiple publications, you might collect them into a map and reuse the map as a submap in multiple other maps, rather than referencing the topics individually from the new maps.

You add a submap to a map the same way that you would add a new topic or insert an existing topic into a map (on page 2917), except you choose a map rather than a topic to create or add. When adding a submap
to a map make sure that you use a `<mapref>` element or a `<topicref>` element with the `@format` attribute set to `ditamap`. In most cases, Oxygen XML Editor takes care of this for you.

**Adding a Submap to a Map**

To add a submap to a map:

1. Right-click the place in the current map where you want to add the new submap.

2. To insert the submap as a child of the selected node, select **Append Child > New**. To insert the submap as a sibling to the current node, select **Insert After > New** or **Insert Before > New**.

   **Step Result:** This opens a New DITA file dialog box *(on page 2961)* that allows you to select the type of document and assists you with naming it.

3. Select the type of map in one of the folders inside the **DITA Map** folder and give it a name (the file should have a `.ditamap` file extension).

4. Click **Create** to insert the submap.

You can manage and move submaps the same as you would with topics. They can also be expanded and you can navigate their content in the **DITA Maps Manager** when the root (main) DITA map is open, but the references within those submaps are not editable, by default, unless you open the submap separately in its own tab.

**Tip:**

If you want to be able to edit submaps when the main (root/parent) map is open in the **DITA Maps Manager**, go to **Options > Preferences > DITA > Maps** and select the **Allow referenced submaps to be edited** option *(on page 271)*.

**Related Information:**

Managing DITA Maps *(on page 2916)*

**Creating a Bookmap in DITA**

If you want to create a traditional book in DITA, you can use a **bookmap** *(on page 3216)* to organize your topics into a book. A DITA **bookmap** is a specialized type of map, intended for creating output that is structured like a book. A **bookmap** allows you to add book-specific elements such as `<frontmatter>`, `<part>`, `<chapter>`, `<appendix>`, and `<backmatter>` to the map. How these book-specific elements are processed for publication is up to the processing script for each media. See the **DITA documentation** for details.

You can find additional support for creating books in DITA in the DITA for Publishers plugin, which is included with Oxygen XML Editor.

To create a book in DITA using a **bookmap**, follow these steps:
1. Create a new **bookmap (on page 2915)** (File > New > Framework templates > DITA Map > map > Bookmap). If you want the bookmap to be a submap, you can create it the same way by right-clicking the place in the current map where you want to add it (in the DITA Maps Manager (on page 2897)) and selecting New from the Append Child, Insert Before, or Insert After submenus.

2. Create the structure of your book by adding the appropriate book sections and defining containers for chapters and any appendices. To add sections to a bookmap, or children to a section, right-click the bookmap or section icon and choose any of the reference actions in the Append child menu. The selections offered in the menu will adjust depending on the element they are applied to. Consult the DITA documentation to fully understand the structure of a DITA bookmap and where to create each element.

3. Create special elements such as an **index (on page 2939)** and table of contents (on page 2939). The index and table of contents will be generated by the build process, based on the content of the map and the topics it points to.

4. **Add topics (on page 2917)** to your chapters to add content to your book. You may find it easier to manage if you use submaps (on page 2914) to create the content of your chapters. This keeps your bookmap from becoming long and difficult to manage.

### Managing DITA Maps

You may want to manage your **DITA maps (on page 3218)** in a variety of ways, including:

- Change the order and nesting of topics in a map.
- Add topics to a map.
- Insert various types of references in a map.
- Find, move, or rename resources in a map.
- Change other properties of the items in a map.
- Use the Edit Properties dialog box to manage attributes, keys, metadata, or add profiling to any section of a map.

This section includes various topics that describe how you can manage DITA maps and resources.

For more information about the DITA Maps Manager view and many of its features, watch our video demonstration:

https://www.youtube.com/embed/ozFZz6YZMCY

### Change the Order of Topics in DITA Maps

You can change the order and nesting of the topics in a map in several ways:
• By dragging and dropping topics within the DITA Maps Manager (on page 2897).
• By highlighting a topic in the DITA Maps Manager (on page 2897), holding down the Alt key, and pressing the arrow keys.
• By showing the extended DITA Maps Manager (on page 2897) toolbar (click the Settings icon on the DITA Maps Manager toolbar and select the extended toolbar) and then using the node moving buttons (↑↓←→) on the toolbar to move topics around in the map.

To understand how to organize topics in a DITA map using the DITA Maps Manager (on page 2897), you can examine and experiment with the sample map called flowers.ditamap, located in the {OXYGEN_INSTALL_DIR}/samples/dita folder.

### Adding Topics to a DITA Map

When you are working in DITA, there are several approaches that you can use to create topics and maps. You can start by first creating topics and then assembling your finished topics into one or more documents by creating one or more maps, or you can start by creating a map and then adding new topics to it as you work.

The topics-first approach is generally more appropriate if you intend to do a lot of content reuse, as it encourages you to think of each topic as an independent unit that can be combined with other topics in various ways. The map-first approach will be more familiar to you if you are used to creating books or manuals as a whole. Oxygen XML Editor supports both approaches.

A DITA map (on page 3218) organizes content hierarchically, so you can add a topic as a child of the map root element or as a child or sibling of any item already in the map. Therefore, the first step to adding a topic to a map is always to choose the place it will be inserted into the map.

### Adding Existing Topics to a Map

At the XML-level, a topic is added to a map by adding a reference to the map that points to the topic. There are a variety of reference types that you can use. The default type is the `<topicref>` element. See the DITA documentation for the full range of reference elements and their uses. Oxygen XML Editor provides several tools for inserting reference elements into a map:

#### Using the Insert Reference Dialog Box

The Insert Reference dialog box (on page 2923) allows you to create various reference types and configure the most commonly used attributes. You can open the Insert Reference dialog box with any of the following methods:

- Right-click an item in the current map where you want to add the reference, select Append Child, Insert Before, or Insert After and select the type of reference to enter.
- If the topic you want to add is currently open in the editor, you can right-click an item in the current map where you want to add the reference and select Reference to the currently edited file.
• Selecting an item in the map and click the Insert Reference button from the DITA Maps Manager (on page 2897) toolbar.
• Select Insert Reference from the DITA Maps menu.

Dragging and Dropping a File into the DITA Maps Manager

You can add a topic to a DITA map by dragging and dropping the file into the DITA Maps Manager (on page 2897). You can drag and drop files from any of the following:

• Your OS file system explorer.
• The Project view (on page 402).
• The Open/Find Resource view (on page 422).

Adding topics this way will not open the Insert Reference dialog box, but you can adjust all the same properties by invoking the contextual menu from the topic and selecting Edit Properties.

Adding a New Topic to a Map

To add a new topic to a map, follow these steps:

1. In the DITA Maps Manager (on page 2897), right-click the node in the current map where you want to add the new topic.

2. Select one of the following actions:
   ◦ Append Child > New - Select this action to insert the new topic as a child of the selected node. This action opens a New file dialog box (on page 2962) that allows you to select the type of document and assists you with naming it. After you have configured your new topic, click Create.
   ◦ Insert Before > New - Select this action to insert the new topic as a sibling to the current node, before it. This action opens a New file dialog box (on page 2962) that allows you to select the type of document and assists you with naming it. After you have configured your new topic, click Create.
   ◦ Insert After > New - Select this action to insert the new topic as a sibling to the current node, after it. This action opens a New file dialog box (on page 2962) that allows you to select the type of document and assists you with naming it. After you have configured your new topic, click Create.
   ◦ Duplicate - Select this action to create a copy of the selected topic and insert it as a sibling. This action opens a dialog box that allows you to choose the file name and location for the newly created copy of the topic. After you have selected the name and path for your new topic, click OK.

Note:
The value of the root ID is generated taking the Use the file name as the value of the root ID attribute option from the DITA > Topics preferences page (on page 273) into account. When the option is deselected, a unique ID is generated.
**Step Result:** The new topic is now referenced (as a `<topicref>`) in the DITA map at the location where you inserted it and the new topic is opened in the editor.

3. Save the DITA map.

### Adding Multiple Skeleton Topics at Once

Oxygen XML Editor includes a feature in the DITA Maps Manager (on page 2897) that allows you to quickly create multiple skeleton topics at once and you can specify their hierarchical structure within the DITA map (on page 3218).

To access this feature, right-click a node in the DITA Maps Manager where you want the new topics to be inserted and select Fast Create Topics. This opens the Fast Create Topics dialog box where you can configure the structure for the new topics.

For more information, see Fast Create Multiple DITA Topics (on page 2964).

### Adding Multiple References to the Same Topic in a Map

Oxygen XML Editor allows you to reuse entire topics by adding multiple references to the same topic in a DITA map. Whenever multiple references to the same topic are detected in the context of the current map in the DITA Maps Manager (on page 2897), an indicator will appear in the top-right corner of the Author mode editor that shows the number of times the topic is referenced in the DITA map. It also includes navigation arrows that allow you to jump to the next or previous reference.

### Remove Topics from a Map

You can remove topics from a map in a number of ways. Some ways to remove a topic from a map include:

- Highlight the topic and press the Delete or Backspace key on your keyboard.
- Highlight the topic and click the>Delete</a> button on the DITA Maps Manager (on page 2897) extended toolbar.

### Moving and Renaming Resources

You can move or rename resources referenced in your DITA project on disk directly from Oxygen XML Editor and you have the option of updating all the references to the moved or renamed resources. If the resources are referenced in the DITA map, you can do this from the DITA Maps Manager view (on page 2897). You can also move and rename resources (DITA topics, maps, or other resources such as folders, images, HTML files, audio, video, text files, Markdown documents) from the Project view (on page 402). If you have enabled
Main Files support (on page 3181), you will also have the option to update all the references to the moved or renamed resource.

Moving or Renaming DITA Resources (Topics or Maps)

To move or rename resources (such as topics, maps, or other referenced non-DITA resources) referenced directly in the DITA map, use one of the following actions available in the Refactoring submenu of the contextual menu when invoked on the resource in the DITA Maps Manager view (on page 2897):

**Refactoring > Move resource**

This action allows you to change the location of a resource linked in the edited DITA map using the Move resource dialog box. This dialog box contains the following options:

- **Destination** - Specifies the target location of the edited resource.
- **File name** - Allows you to change the name of the edited resource.
- **Update references** - Select this checkbox to update all references of the file in the edited DITA map and in the files referenced from the DITA map (preserving its completeness).
- **Preview** - Select this button to display a preview of the changes Oxygen XML Editor is about to make.
- **Move** - Moves the edited resource in the target location on disk.
- **Cancel** - Cancels the Move resource operation. No changes are applied.

**Refactoring > Rename resource**

This action allows you to change the name of a resource linked in the edited DITA map (on page 3218) using the Rename resource dialog box. This dialog box contains the following options:

- **New name** - Presents the current name and allows you to change it.
- **Update references** - Select this checkbox to update all references of the file in the edited DITA map and in the files referenced from the DITA map (preserving its completeness).
- **Preview** - Select this button to display a preview of the changes Oxygen XML Editor is about to make.
- **Rename** - Executes the Rename resource operation.
- **Cancel** - Cancels the Rename resource operation. No changes are applied.

**Note:**

If a root DITA map (on page 3223) is not defined, the move and rename actions are executed in the context of the current DITA map.

Moving or Renaming Resources and Updating the References to Them Using the Project View

To move or rename DITA (topics, maps) or non-DITA resources (such as folders, images, HTML files, audio, video, text files, Markdown documents), you can simply follow the procedures described in Moving/Renaming.
Resources in the Project View (on page 412). However, this approach will not give you the option to update the references to the moved or renamed resources.

To perform move or rename operation on resources while also updating all the references to them, use the following sets of procedures:

1. Enable Main Files support and add your root DITA map (on page 3223) to the Main Files folder by following the procedure found here: How to Enable Main Files Support in DITA (on page 3181).
2. Move or rename resources and update the references to them by following the procedure found here: Moving or Renaming Non-DITA Resources and Updating the References to Them (on page 3182).

Finding Resources Not Referenced in DITA Maps

Over the course of time, large projects can accumulate a vast amount of resources from a variety of sources. Especially in organizations with a large number of content authors or complex project structures, organizing the project resources can become a challenge. Over time a variety of actions can cause resources to become orphaned from DITA maps (on page 3218). To assist you with organizing project resources, Oxygen XML Editor includes the Find Unreferenced Resources action, that searches for such resources.

To perform this search, open the DITA map in the DITA Maps Manager (on page 2897), invoke the contextual menu on the map, and select the Find Unreferenced Resources action. It can also be selected from the DITA Maps menu. This action opens the Find Unreferenced Resources dialog box, shown below.
The **Find Unreferenced Resources** dialog box includes the following options:

- **DITA Maps** - Provides a list of DITA maps to be included in the search and allows you to **Add** maps to the list or **Remove** them.
- **Folders** - Provides a list of folders to be included in the search and allows you to **Add** or **Remove** specific folders. All files from this list of folders that are not referenced from the maps specified in the **DITA Maps** list will be reported.
- **Filters** - Provides three combo boxes that allow you to filter the search to include or exclude certain files or folders:
  - **Include files** - Allows you to filter specific files to include in the search.
  - **Exclude files** - Allows you to filter specific files to exclude from the search.
  - **Exclude folders** - Allows you to filter specific folders to exclude from the search.

**Note:**
In any of the filter combo boxes you can enter multiple filters by separating them with a comma and you can use the ? and * wildcards. Use the drop-down arrow to select a previously used filter pattern.

When you click the **Find** button, if the search operation finds unreferenced resources, they are displayed in the **Results** panel at the bottom of the editor. If you want to delete an unreferenced resource, you can right-click
its result and select **Remove from Disk**. If you want to see the resource before deciding what to do with it, you can right-click its result and select **Show in Explorer**.

### Inserting References in DITA Maps

A *DITA map (on page 3218)* may contain various types of references. The targets of the references can be a variety of references, such as chapters, maps, topics, topic sets, or key definitions. You can insert references to such targets with the **Insert Reference dialog box (on page 2923)**.

This section explains how to insert and configure references (such as topic references, topic groups, topic headings, and key definitions) in a *DITA map*.

### Insert Reference Dialog Box

The **Insert Reference** dialog box allows you to insert and configure references in *DITA maps (on page 3218)*. There are numerous types of references that can be inserted into maps. They include references to topics, other maps, glossary terms, and keys. You can also use this dialog box to configure the attributes of a reference, add profiling or metadata, and define keys.

To open the **Insert Reference** dialog box, use one of the following methods:

- Select **Reference, Reference to the currently edited file**, or any of the other specific reference actions that are available from the Append Child, Insert Before, and Insert After submenus when invoking the contextual menu in the **DITA Maps Manager (on page 2897)**.
  - To insert the reference as a child of the current node, select the reference from the Append Child submenu.
  - To insert the reference as a sibling of the current node, below the current node in the map, select the reference from the Insert After submenu.
  - To insert the reference as a sibling of the current node, above the current node in the map, select the reference from the Insert Before submenu.

  **Note:**
  The content of these submenus depends on the node that is selected in the *DITA map* tree when the contextual menu is invoked. For example, if the selected node is a topic reference (*<topicref>*), its possible child nodes include the following elements: *<anchorref>, <chapter>, <keydef>, <mapref>, <topicgroup>, <topichead>, <topicref>, <topicset>, and <topicsetref>*.

- Click the **Insert Reference** button on the **DITA Maps Manager** extended toolbar. This action will insert the reference as a sibling of the current node (below the current node in the map).

- Select **Insert Reference** from the **DITA Maps** menu. This action will insert the reference as a sibling of the current node (below the current node in the map).

For the **Reference** or **Reference to the currently edited file** actions, a **Reference type** drop-down list is displayed at the top of the **Insert Reference** dialog box and you can select the type of reference you want
to insert. Depending on the place where the reference will be inserted, Oxygen XML Editor will propose only valid reference types. When you change the reference type, the fields in the various tabs of the dialog box are reconfigured depending upon the availability of the associated attributes. For the other reference actions in the Append Child, Insert Before, and Insert After submenus, the reference type is automatically chosen based upon the invoked action and you cannot change it.

The main section of the dialog box includes the following tabs: Target, Keys, Attributes, Metadata, and Profiling.

**Target Tab**

Figure 736. Insert Reference Dialog Box - Target Tab

The Target tab of the Insert Reference dialog box allows you to specify information about the target reference. It includes the following sections and fields:

**Choose a file location section**

You can browse for and select the source target file by using the Look in drop-down list, browsing buttons, or file window in this section. You can use the Files of type drop-down menu to narrow the list of possible file types that will be displayed.

**URL**

Displays the path to the target and allows you to select or change it by using the combo box or browsing buttons.
ID

The drop-down list displays all of the target elements that are available for the selected target URL.

Href

The selected target automatically modifies this value to point to the corresponding `@href` attribute of the target element.

Note:
If the Reference type is a Navigation Reference, the Href field is changed to Mapref, since a `<navref>` element requires a `@mapref` attribute instead.

Keys Tab

Figure 737. Insert Reference Dialog Box - Keys Tab

The Keys tab allows you to use and define keys (on page 2930) for indirect referencing. For more information, see Working with Keys in DITA (on page 3026). This tab includes the following:

Define keys

Use this text field to define the `@keys` attribute for the target.

Key scopes

Use this text field to define or edit the value of a `@keyscope` attribute. Key scopes allow you to specify different sets of key definitions for different map branches.

Key reference

Instead of using the Target tab to select a file that contains the target reference, you can reference a key definition by using this text field. Use the Choose key reference button to access the list of keys that are already defined in the current root map (on page 3223).
Attributes Tab

The **Attributes** tab of the **Insert Reference** dialog box allows you to insert and edit attribute values for the target reference. This tab includes the following sections and actions:

**Navigation title**

This text field allows you to specify a custom navigation title for the target reference. If you want this attribute to always be populated with a detected value (based on the specifications for the target file), select the **Navigation title** checkbox for the **Always fill values for attributes** option in the **DITA preferences page (on page 271)**. For references to DITA resources, you can enforce the use of the specified title by selecting the **Lock** checkbox (otherwise, the topic `<title>` takes precedence).

**Collection type**

This drop-down list allows you to select the `@collection-type` attribute to create hierarchical linking between topics in a **DITA map** (for example, `unordered`, `sequence`, `choice`, `family`, `-dita-use-conref-target`).

**Type**

Allows you to select a `@type` attribute (such as `topic`, `task`, `concept`, etc.) for the target element. If you want this attribute to always be populated with a detected value (based on the specifications for the target file), select the **Type** checkbox for the **Always fill values for attributes** option in the **DITA preferences page (on page 271)**.

**Scope**

This property corresponds to the `@scope` attribute of the target element. It is populated automatically, based on the selected file type, unless its value for the selected target file is the same as the default attribute value. If you want this attribute to always be populated with a detected value based on the specifications (regardless of the default value), select the **Scope**
checkbox for the **Always fill values for attributes** option in the DITA preferences page (on page 271).

**Format**

This property corresponds to the `@format` attribute of the target element. It is populated automatically, based on the selected file type, unless its value for the selected target file is the same as the default attribute value. If you want this attribute to always be populated with a detected value based on the specifications (regardless of the default value), select the **Format** checkbox for the **Always fill values for attributes** option in the DITA preferences page (on page 271).

**Processing Role**

This drop-down list allows you to set the `@processing-role` attribute to one of the allowed values for DITA reference elements (for example, resource-only, normal, -dita-use-conref-target).

**Other attributes table**

This table contains the attributes that are available for the selected reference. You can use this table to insert or edit the values of any of the listed attributes. Clicking a cell in the **Value** column allows you to use the combo box to enter, edit, or select attribute values.

**Metadata Tab**

![Figure 739. Insert Reference Dialog Box - Metadata Tab](image)

The **Metadata** tab allows you to add metadata elements to the target reference. Use the buttons on the right side of the tab to insert specific metadata elements (you can add the following metadata elements: `<navtitle>`, `<linktext>`, `<shortdesc>`, `<keyword>`, `<indexterm>`). The metadata elements are inserted inside a `<topicmeta>` element. The editing window allows you to easily insert and modify the content of the metadata that will be inserted.
Profiling Tab

Figure 740. Insert Reference Dialog Box - Profiling Tab

The **Profiling** tab allows you to select or change profiling attributes for the selected reference. This tab displays profiling attributes and their values as determined by the following:

- If your root map (on page 3223) references a DITA subject scheme map (on page 3224) that defines values for the profiling attributes, those values are used.
- If your project defines project-level (on page 3222) configuration values for the profiling attributes (on page 187), those values are used.
- If Oxygen XML Editor defines global-level (on page 3219) configuration values for the profiling attributes (on page 187), they are used.
- Otherwise, a basic default set of profiling attributes and values are used.

When you modify a selection of values in this tab, the change will also automatically be reflected in the Attributes tab. For more information, see DITA Profiling / Conditional Text (on page 3133).

Finalizing Your Insert Reference Configuration

Once you click **Insert** or **Insert and close**, the configured reference is added in the map.

**Tip:**

You can easily insert multiple references by keeping the **Insert Reference** dialog box opened, using the **Insert** button.

Related Information:

- DITA Profiling / Conditional Text (on page 3133)
- Working with Keys in DITA (on page 3026)
Inserting Topic Headings

The `<topichead>` element provides a title-only entry in a navigation map, as an alternative to the fully-linked title provided by the `<topicref>` element.

You can insert a topic heading by doing the following:

- Select **Topic Head** from the **Append Child, Insert Before,** or **Insert After** submenus when invoking the contextual menu in the **DITA Maps Manager** view (on page 2897).
- Open the **DITA map** in the **XML editor** (on page 2896) and select the **Insert Topic Heading** action from the main toolbar (or from the **Insert** submenu of the contextual menu).

Those actions open the **Insert Topic Head** dialog box (on page 2923) that allows you to easily insert a `<topichead>` element. A **Navigation title** (```@navtitle``` attribute) is required but other attributes can also be specified from this dialog box (such as **Type, Scope, Format,** etc.)

![Figure 741. Insert Topic Heading Dialog Box](image)

Related Information:

- **Insert Reference Dialog Box** (on page 2923)

Inserting Topic Groups

The `<topicgroup>` element identifies a group of topics (such as a concepts, tasks, or references) or other resources. A `<topicgroup>` can contain other `<topicgroup>` elements, allowing you to express navigation or table-of-contents hierarchies, as well as implying relationships between the containing `<topicgroup>` and its children.

You can set the collection-type of a container `<topicgroup>` to determine how its children are related to each other. Relationships end up expressed as links in the output (with each participant in a relationship having links to the other participants by default).
You can insert a topic group by doing the following:

- Select **Topic Group** from the **Append Child**, **Insert Before**, or **Insert After** submenus when invoking the contextual menu in the **DITA Maps Manager** view (on page 2897).
- Open the **DITA map** in the XML editor (on page 2896) and select the [Insert Topic Group action](on page 2896) from the main toolbar (or from the Insert submenu of the contextual menu).

Those actions open the **Insert Topic Group** dialog box (on page 2923) that allows you to easily insert a `<topicgroup>` element and various attributes can be specified (such as **Collection type**, **Type**, **Scope**, **Format**, etc.)

**Figure 742. Insert Topic Group Dialog Box**

![Insert Topic Group dialog box](image)

**Related Information:**

- **Insert Reference Dialog Box (on page 2923)**

**Defining Keys in DITA Maps**

DITA uses **keys (on page 3026)** to insert content that may have different values in various circumstances. **Keys provide the means for indirect referencing in DITA.** This can make it easier to manage and to reuse content. In DITA, keys are defined in maps and can then be reused and referenced throughout the whole structure of the map. It is considered best practice to create a separate submap that contains all of the key definitions and reference that submap in the **main (root) map (on page 3223)**. This makes it easier to manage since they're all in one location.

There are two types of key definitions that can be created in a map:

- Key with a value inside a **keyword**.
- Key with a target (for example, to target a resource such as an image or external link).
The following example is a DITA map (on page 3218) (a key definition submap) that contains some key definitions with various values for the product key and some targets to external URLs:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE map PUBLIC "-//OASIS//DTD DITA Map//EN" "map.dtd">
<map id="keydefs">
  <!-- product name -->
  <title>Key Definitions</title>
  <keydef keys="product" product="basic">
    <topicmeta>
      <keywords>
        <keyword>Basic Widget</keyword>
      </keywords>
    </topicmeta>
  </keydef>
  <keydef keys="product" product="pro">
    <topicmeta>
      <keywords>
        <keyword>Professional Widget</keyword>
      </keywords>
    </topicmeta>
  </keydef>
  <keydef keys="url_eula" href="https://www.example.com/eula.html" format="html" scope="external"/>
  <keydef keys="url_eula2" href="https://www.example.com/eula2.html" format="html" scope="external"/>
</map>
```

Note:
The profiling of the names is now contained in the map, where it only has to occur once to reuse throughout the whole map structure.

Key Definition with a Keyword Value

To define a key with a value inside a keyword, follow these steps:

1. [Optional but Recommended] Create a submap (on page 2914) that will contain all of your key definitions and reference the submap in your main (root) map (if you don't already have one created).
2. Open that map in the DITA Maps Manager (on page 2897).
3. Right-click the map or an item in the map where you want to add the reference and select Key Definition with Keyword from the Append Child, Insert Before, or Insert After submenu (depending on where you want to insert the key definition). This opens an Insert Key Definition dialog box.
4. Enter the name of the key in the Key field.
5. Enter the key's value in the **Keyword** field.
6. Click **Insert and close**.

**Tip:**
If you need to profile the key or add other attributes, you can right-click the key definition in the **DITA Maps Manager**, select **Edit properties**, and configure them in the **Profiling** tab or **Attributes** tab, respectively.

### Key Definition with a Target

To insert a targeted key definition (for example, to target a resource such as an image or external link), follow these steps:

1. **[Optional but Recommended]** Create a submap ([on page 2914](#)) that will contain all of your key definitions and reference the submap in your main (root) map (if you don't already have one created).
2. Open that map in the **DITA Maps Manager** ([on page 2897](#)).
3. Right-click the map or an item in the map where you want to add the reference and select **Key Definition** from the **Append Child**, **Insert Before**, or **Insert After** submenu (depending on where you want to insert the key definition in the **DITA map**). This opens an **Insert Key Definition** dialog box.
4. Go to the **Keys** tab and enter the name of the key in the **Define keys** field.
5. Go to the **Target** tab and select a target resource (such as an image or external link).

**Tip:**
You can profile the key by using the **Profiling** tab and other attributes can also be defined in the **Attributes** tab.

6. Once you are done configuring the targeted key definition, click **Insert and close**.

**Related Information:**
- Working with Variable Text in DITA ([on page 3056](#))
- Working with Keys in DITA ([on page 3026](#))
- DITA 1.3 Specification: Indirect Key-based Addressing

### Edit Properties Dialog Box

The **DITA Maps Manager** view ([on page 2897](#)) includes a feature that allows you to view and edit the properties of a selected node. The **Edit properties** action is available on both the **DITA Maps Manager** toolbar and in the contextual menu. This action is also available in the contextual menu when you edit a **DITA map** ([on page 3218](#)) document in **Author** mode. The action opens the **Edit Properties** dialog box and it includes several tabs with various functions and fields that are initialized with values based upon the node where the action was invoked.
Note:
If you select multiple sibling nodes and invoke the *Edit properties* action, only the *Profiling* tab will be available and your modifications in that tab will be applied to all the selected nodes. If you select multiple nodes that are not on the same hierarchical level, the other tabs will also be available and your modifications will be applied to the parent node (the child nodes will inherit the attributes of the parent node).

You can use the *Edit Properties* dialog box to modify or define attributes, metadata, profiling, or keys in *DITA maps* or topics. You can also use it to modify the title of *root maps (on page 3223)*.

At the top of the *Edit Properties* dialog box, the *Reference type* drop-down list displays the type of the selected node and it depends on the node where the action was invoked.

The main section of the dialog box includes the following tabs: *Target*, *Keys*, *Attributes*, *Metadata*, and *Profiling*. The availability of the tabs and their functions depend on the selected node. For example, if you invoke the action on a *root map (on page 3223)*, only the *Attributes*, *Metadata*, and *Profiling* tabs are accessible and the *Title* property can be configured. Also, if you select multiple nodes, only the *Profiling* tab is available.

**Target Tab**

**Figure 743. Edit Properties Dialog Box - Target Tab**

The *Target* tab of the *Edit Properties* dialog box displays information about the target node on which the action was invoked and allows you to change the target. It includes the following sections and fields:

- **Choose a file location section**
You can browse for and select the source target file by using the Look in drop-down list, browsing buttons, or file window in this section. You can use the Files of type drop-down menu to narrow the list of possible file types that will be displayed.

**URL**

Displays the path to the target and allows you to select or change it by using the combo box or browsing buttons.

**ID**

The drop-down list displays all of the target elements that are available for the selected target URL.

**Href**

The selected target automatically modifies this value to point to the corresponding @href attribute of the target element.

*Note:*

If the Reference type is a Navigation Reference, the Href field is changed to Mapref, since a <navref> element requires a @mapref attribute instead.

**Keys Tab**

The Keys tab allows you to use and define keys (on page 2930) for indirect referencing. For more information, see Working with Keys in DITA (on page 3026). This tab includes the following:

**Define keys**

Use this text field to define the @keys attribute for the target.

**Key scopes**
Use this text field to define or edit the value of a @keyscope attribute. Key scopes allow you to specify different sets of key definitions for different map branches.

**Key reference**

Use this combo box (or the Choose key reference button) to select a key that is already defined in the root map (on page 3223).

### Attributes Tab

**Figure 745. Edit Properties Dialog Box - Attributes Tab**

The Attributes tab of the Edit Properties dialog box allows you to insert and edit attribute values for the target node where the action was invoked.

If the target is a root map (on page 3223), the tab displays the title of the map. You can change it in the Title text field and assign it to an Attribute, Element, or All. However, if the title of the map contains elements other than plain text, the title is not editable and cannot be changed using this dialog box (you would need to open the DITA map in the main editor to edit the title).

**Figure 746. Attributes Tab for a Root Map**

For other types of targets, the tab includes the following sections and fields that can be used to edit the attributes of the target:

**Navigation title**
This text field allows you to specify a custom navigation title for the target reference. If you want this attribute to always be populated with a detected value (based on the specifications for the target file), select the **Navigation title** checkbox for the **Always fill values for attributes** option in the **DITA preferences page** *(on page 271)*. For references to DITA resources, you can enforce the use of the specified title by selecting the **Lock** checkbox (otherwise, the topic `<title>` takes precedence).

**Tip:**
You can also select the **Prefer navigation title for topicref rendering** option in the **DITA preferences page** *(on page 271)* to always enforce the use of the `@navtitle` value rather than selecting this **Lock** option on individual topics.

**Collection type**

This drop-down list allows you to select the `@collection-type` attribute to create hierarchical linking between topics in a **DITA map** *(for example, unordered, sequence, choice, family, -dita-use-conref-target)*.

**Type**

Allows you to select a `@type` attribute (such as `topic`, `task`, `concept`, etc.) for the target element. If you want this attribute to always be populated with a detected value (based on the specifications for the target file), select the **Type** checkbox for the **Always fill values for attributes** option in the **DITA preferences page** *(on page 271)*.

**Scope**

This property corresponds to the `@scope` attribute of the target element. It is populated automatically, based on the selected file type, unless its value for the selected target file is the same as the default attribute value. If you want this attribute to always be populated with a detected value based on the specifications (regardless of the default value), select the **Scope** checkbox for the **Always fill values for attributes** option in the **DITA preferences page** *(on page 271)*.

**Format**

This property corresponds to the `@format` attribute of the target element. It is populated automatically, based on the selected file type, unless its value for the selected target file is the same as the default attribute value. If you want this attribute to always be populated with a detected value based on the specifications (regardless of the default value), select the **Format** checkbox for the **Always fill values for attributes** option in the **DITA preferences page** *(on page 271)*.

**Processing Role**

This drop-down list allows you to set the `@processing-role` attribute to one of the allowed values for DITA reference elements (for example, `resource-only, normal, -dita-use-conref-target`).

**Other attributes table**
This table contains the attributes that are available for the selected reference. You can use this table to insert or edit the values of any of the listed attributes. Clicking a cell in the Value column allows you to use the combo box to enter, edit, or select attribute values.

Metadata Tab

Figure 747. Edit Properties Dialog Box - Metadata Tab

The Metadata tab allows you to add metadata elements to the target node. Use the buttons on the right side of the tab to insert specific metadata elements (you can add the following metadata elements: \texttt{<navtitle>}, \texttt{<linktext>}, \texttt{<shortdesc>}, \texttt{<keyword>}, \texttt{<indexterm>}). The metadata elements are inserted inside a \texttt{<topicmeta>} element. The editing window allows you to easily insert and modify the content of the metadata that will be inserted.

Profiling Tab

Figure 748. Edit Properties Dialog Box - Profiling Tab

The Profiling tab allows you to select or change profiling attributes for the selected target nodes. This tab displays profiling attributes and their values as determined by the following:
If your root map (on page 3223) references a DITA subject scheme map (on page 3224) that defines values for the profiling attributes, those values are used.

- If your project defines project-level (on page 3222) configuration values for the profiling attributes (on page 187), those values are used.
- If Oxygen XML Editor defines global-level (on page 3219) configuration values for the profiling attributes (on page 187), they are used.
- Otherwise, a basic default set of profiling attributes and values are used.

If you have a large list of profiling attributes, you can use the text filter field to search for attributes or values, and you can expand or collapse attributes by using the Expand All/Collapse All buttons to the right of the text filter or the arrow button to the left of the profiling attribute name.

When you modify a selection of values in this tab, the change will also automatically be reflected in the Attributes tab. For more information, see DITA Profiling / Conditional Text (on page 3133).

Note:
If you invoke the Edit properties action on a selection of multiple nodes that have different values for the same profiling attribute, a conflict panel will be displayed in the Profiling tab and you can choose between the following actions for resolving it:

- **Keep** - Preserves the current attribute values.
- **Change Now** - Allows you to edit the selection of values in this Profiling tab and the changes will be applied to all the selected nodes.

![Figure 749. Profiling Conflict Panel](image)

Finalizing Your Modifications

Once you click OK, all your changes are applied to the target node.

Related Information:

- DITA Profiling / Conditional Text (on page 3133)
- Working with Keys in DITA (on page 3026)
Generating a Table of Contents in DITA

In DITA, the order and hierarchy of the table of contents of a document is based directly on the DITA map that defines the document (on page 2895). In most cases, the processor generates a table of contents (TOC) based on the hierarchy of the topics in a DITA map. By default, each `<topicref>` element in a map represents a node in the TOC.

It is also possible to instruct DITA where the table of contents should occur (or other content lists, such as a list of figures or tables). If you want to instruct the processor to generate a table of contents at a particular location within your DITA map structure, you can use the `<toc>` element in a bookmap (on page 2915) (as in the example below). For more information about the `<toc>` element, see https://docs.oasis-open.org/dita/v1.2/os/spec/langref/toc.html.

Example:

```xml
<bookmap>

.....

<frontmatter>

<booklists>

<toc href="chapter1.dita"/>
</booklists>
</frontmatter>

.....
```

Creating an Index in DITA

In DITA, indexes are created from `<indexterm>` elements. You can insert index term elements in the following:

- **The header of a topic:** In paginated media, such as a printed book or a PDF, this results in an index entry that points to the page where the topic starts, even if it is not the page in which the indexed term occurs.

- **In the `<topicref>` element in a map that references the topic:** This applies those index terms to that topic only when used in that map, allowing you to index topics differently in various publications. In paginated media, index entries point to the page where the topic starts.

- **In the body of a topic:** In paginated media, this results in an index entry that points to the page where the `<indexterm>` element occurs, even if that is not the page where the topic starts.

To add index terms to the text of a topic or the topic header, create the elements as you normally would in Oxygen XML Editor (on page 2967). To add index terms to a map, open the map in the editor and add the elements, as you normally would, in a topic.

In some media, indexes will be generated automatically when index entries are found in the source. For other media, such as books, you may need to tell DITA where to place the index. For instance, to add an index to a bookmap (on page 3216), you need to add an `<indexlist>` element to the `<backmatter>` of the book.
1. Open your bookmap (on page 2915) in the DITA Maps Manager (on page 2897).
2. Right-click the bookmap and select Append Child > Backmatter.
   The Insert Reference dialog box (on page 2923) appears.
3. Click Insert and Close to insert the <backmatter> element.
4. Right-click the <backmatter> element and create a <booklists> element using Append Child > Book Lists.
5. Use the same steps to create an <indexlist> element.

**CAUTION:**
Adding index entries and an <indexlist> to your project creates an instruction to the DITA publishing routines to create an index. There is no guarantee that all DITA output types or third-party customizations obey that instruction or create the index the way you want it. Modifying the output may be necessary to get the result you want.

### Resolving Topic References Through an XML Catalog

There are situations where you want to resolve references with an XML Catalog (on page 3225):

- You customized your DITA map (on page 3218) to reference topics using URIs instead of local paths.
- You have URI content references in your DITA topic files and you want to map them to local files when the map is transformed.

In such situations, you have to add the catalog to Oxygen XML Editor. The DITA Maps Manager view (on page 2897) will solve the displayed topic refs through the added XML catalog URI mappings. The resolution through the XML catalog URI mappings are done only for reference values starting with the urn: prefix.

To add an XML catalog to the DITA framework (on page 3219), follow these steps:

1. Create an XML catalog using the guidelines described in Working with XML Catalogs (on page 826).
2. Open the Preferences dialog box (Options > Preferences) (on page 124) and go to Document Type Association.
3. Select the DITA document type and use the Edit, Duplicate, or Extend button to open a Document type configuration dialog box (on page 140).
4. Go to the Catalogs tab (on page 164).
5. Click on the Add button to open a dialog box that allows you to add your created XML Catalog to the list.
6. After adding your catalog, click OK. You may need to reopen any currently edited files that use the new catalog or run a manual Validate action (on page 775) for the changes to take effect.

**Note:**
You could also add your created catalog to the list of global catalogs in the XML Catalog preferences (on page 235) page.
Adding a Custom URI Resolver to Oxygen XML Editor

You can use the `XMLUtilAccess.addPriorityURIResolver(URIResolver)` API to add your own priority URI resolver from a Workspace Access plugin (on page 2465), allowing you to take control over how topic references in a DITA map are located or how references in DITA topics are resolved.

Publishing a DITA Map with References Resolved Through the XML Catalog

If you are publishing a DITA map that contains references to topics that need to be resolved through the XML catalog support in Oxygen XML Editor, you must enable the `fix.external.refs.com.oxygenxml` parameter in the Parameters tab of the transformation scenario configuration dialog box.

Chunking DITA Topics

By default, when a DITA map (on page 3218) is published to an online format, each topic becomes a separate page in the output. In some cases, you may want to combine multiple source topics into one output page. For instance, you may want to combine several types of information into a single page, or you may have chosen to create many small DITA topics for reuse purposes but feel they are too small to be useful to a reader by themselves. This is referred to as chunking.

To chunk DITA topics, you set the chunking attribute on the `<topicref>` that contains the sub-topics in a DITA map. There are several values that you can set on the chunking attribute (for example, by-topic or to-content). See the DITA documentation for full details. To achieve the effects you want in your topics and table of contents, you may also need to set the `@toc` and `@collection-type` attributes on the sub-topics or container topic to suitable values. See the DITA documentation for details.

You can set the `@collection-type` attribute on your topics using the Edit Properties action in the DITA Maps Manager (on page 2897). To set the `@toc` and `@chunk` attributes, you must open the map file in the editor and add or edit the attributes directly (double-click the map icon in the DITA Maps Manager to open the map in the editor).

DITA Map Validation and Completeness Check

You should validate your DITA maps (on page 3218) regularly to make sure that your maps and topics are valid, and all of the relationships between them are working. Changing one topic, image, or piece of metadata may create errors in references that rely on them. You may not discover these problems all at once. Validate your map to catch all of these kinds of problems. The longer you wait between validating your maps, the more difficult it may be to detect and correct any errors you find.

Validating a DITA Map

To validate a DITA, follow these steps:
1. In the **DITA Maps Manager** view (on page 2897), make sure that the tab that holds your **root map** (on page 3223) is selected and that the **Context** selection is set either to the name of your **root map** or to `<current_map>`.

2. It is a good practice to refresh your **DITA map** before running the validation process. To do so, select the **DITA map** in the **DITA Maps Manager** view and click **Reload (F5)**.

3. Click the **Validate and Check for Completeness** button from the **Validation** drop-down menu on the **DITA Maps Manager** toolbar to open the **DITA Map Completeness Check** dialog box (on page 2942).

4. If you are using profiling, check the **Use DITAVAL filters** box and select the appropriate option.

5. Select any other options you want to check.

6. Click **Check** to run the validation process.

**Result:** A dialog box is displayed showing the progress of the operation. You can click the **Run in Background** button to close this dialog box so that you can continue working while the operation continues in the background and the progress would continue in the information ribbon at the bottom of the application.

**Validation Process**

The validation process of a **DITA map** includes the following:

- Verifies that the file paths of the topic references are valid. For example, if an `@href` attribute points to an invalid file path, it is reported as an error in the message panel at the bottom of the editor.

- Validates each referenced topic and map. Each topic file is opened and validated against the appropriate DITA DTD. If another **DITA map** is referenced in the main one, the referenced **DITA map** is verified recursively, applying the same algorithm as for the main map.

- If errors or warnings are found, they are displayed in a separate message pane at the bottom of the editor and clicking them takes you to the location of the error or warning in the file where it was found.

**DITA Map Completeness Check Dialog Box**

The **DITA Map Completeness Check** dialog box allows you to configure the **DITA map** validation.
You can configure the validation process with the following options that are available in the DITA Map Completeness Check dialog box:

**Batch validate referenced DITA resources**

This option specifies the level of validation that applies to referenced DITA files:

- If the checkbox is left unchecked (default setting), the DITA files will be validated using the rules defined in the DTD or XML Schema declared in the document.
- If the checkbox is selected, the DITA files will be validated using rules defined in their associated validation scenario (on page 787).
Note: If you want your DITA topics to be validated against a Relax NG schema, this checkbox must be selected.

Check the existence of non-DITA references resources

Extends the validation of referenced resources to non-DITA files.

Include remote resources

Select this option if you want to check that remote referenced binary resources (such as images, movie clips, ZIP archives) exist at the specified location.

Use DITAVAL filters

The content of the map is filtered by applying a profiling condition set (on page 3133) before validation. You can choose between the following options:

- **From the current condition set** - The map is filtered using the condition set currently applied in the DITA Maps Manager view (on page 2897). Clicking the 📄 Details icon opens a topic in the Oxygen XML Editor User Guide that explains how to create a profiling condition set.
- **From all available condition sets** - For each available condition set, the map content is filtered using that set before validation.
- **From the associated transformation scenario** - The filtering condition set is specified explicitly as a DITAVAL file in the current transformation scenario associated with the DITA map.
- **Other DITAVAL files** - For each DITAVAL file from this list, the map content is filtered using the DITAVAL file before validation. Use the Add or Remove buttons to configure the list. The Add button opens a dialog box that allows you to select a local or remote path to a DITAVAL file. You can specify the path by using the text field, its history drop-down, the 🗂 Insert Editor Variables (on page 323) button, or the browsing actions in the 🗄 Browse drop-down list.

Report references to resources outside of the DITA map folder

If selected, it will report any references to DITA resources that are located outside the main DITA map (on page 3223) folder.

Report links to topics not referenced in DITA maps

Checks that all the topics referenced by other topics are also linked in the DITA map.

Report multiple references to the same topic

If selected, it will report warnings when a topic is referenced multiple times in the DITA map, unless a unique @copy-to attribute is used on the <topicref> element for any topic that is referenced multiple times.
For example, it will not report a warning if there is a topic referenced twice, but the second
  \texttt{<topicref>} has a \texttt{@copy-to} attribute set:

  \begin{verbatim}
  \begin{verbatim}
  \texttt{<topicref href="topic.dita"/>}
  ...
  \texttt{<topicref href="topic.dita" copy-to="topic2.dita"/>}
  \end{verbatim}
  \end{verbatim}

On the other hand, it will report a warning if there is a topic referenced twice and none of the
reference-type elements has a \texttt{@copy-to} attribute set or both of them have the \texttt{@copy-to} attribute
set to the same value:

  \begin{verbatim}
  \begin{verbatim}
  \texttt{<topicref href="topic.dita" copy-to="topic2.dita"/>}
  ...
  \texttt{<topicref href="topic.dita" copy-to="topic2.dita"/>}
  \end{verbatim}
  \end{verbatim}

Check for duplicate topic IDs within the DITA map context

Checks for multiple topics with the same ID in the context of the entire map.

Report duplicate key definitions

Checks the \textit{DITA map} for multiple key references with the same key defined for them. This is
helpful because if you have two different resources with the same value for the \texttt{@keys} attribute, all
references will point to the first one encountered and the other will be ignored.

\begin{itemize}
  \item \textbf{Note:}
  \begin{itemize}
    \item This option takes \textit{key scopes (on page 3058)} into account. For example, if you have
    something like this:
    \begin{verbatim}
    \begin{verbatim}
    \texttt{<topicref href="t2.dita" keys="k2"/>}
    \texttt{<topicgroup keyscope="ks"} \begin{verbatim}
    \begin{verbatim}
    \texttt{<topicref href="t2.dita" keys="k2"/>}
    \end{verbatim}
    \end{verbatim}\end{verbatim}
    \end{verbatim}
    \end{verbatim}
    \end{verbatim}
    \end{verbatim}
    \end{verbatim}
    \end{verbatim}
    \end{verbatim}
  \end{itemize}
\end{itemize}

it will not report the “k2” key as a duplicate because it is defined in a \textit{key scope (on page 3058)} on the second occurrence.

Report unreferenced key definitions

Checks the entire \textit{DITA map} and reports any key definitions that are not referenced anywhere.
Note that if the \textbf{Use DITAVAL filters} option is selected, this check will search for unreferenced
key definitions based upon your selected filter.

Report unreferenced reusable elements

Checks the entire \textit{DITA map} and reports any detected reusable elements that are not referenced
anywhere. It looks for elements that have an \textit{ID} specified in the following types of topic
references:
Any `<topicref>` that contains a `@processing-role` attribute set to `resource-only`.
Any other referenced topic that contains elements that are reused elsewhere through a `@conref` or `@conkeyref`.

**Report table layout problems**

Looks for table layout problems. The types of errors that may be reported include:

- If a row has fewer cells than the number of columns detected.
- For a CALS table, if a cell has a vertical span greater than the available rows count.
- For a CALS table, if the number of `<colspecs>` is different than the number of columns detected from the table `@cols` attribute.
- For a CALS table, if the number of columns detected from the table `@cols` attribute is different than the number of columns detected in the table structure.
- For a CALS table, if the value of the `@cols`, `@rowsep`, or `@colsep` attributes are not numeric.
- For a CALS table, if the `@names`, `@nameend`, or `@colname` attributes point to an incorrect column name.

**Identify possible conflicts in profile attribute values**

When the profiling attributes of a topic contain values that are not found in parent topic profiling attributes, the content of the topic is overshadowed when generating profiled output. This option reports these possible conflicts.

**Report attributes and values that conflict with profiling preferences**

Looks for profiling attributes and values that are not defined in the Profiling / Conditional Text preferences page (on page 187) (you can click the Profiling Preferences button to open this preferences page). It also checks if profiling attributes defined as `single-value` have multiple values set in the searched topics.

**Additional Schematron checks**

Allows you to select a Schematron file that Oxygen XML Editor will use for the validation of DITA resources. You can specify the path by using the text field, its history drop-down, the Insert Editor Variables button, or the browsing actions in the Browse drop-down list.

**Export settings**

Allows you to export the settings assigned in this dialog box to an XML file that you can share with other users or use on other systems.

**Import settings**

Allows you to import settings for this dialog box from an XML file that was created by the Export settings action.
Use the **Check** button to begin the validation process. The options that you choose in this dialog box are preserved between sessions.

**Tip:**
This function can be executed from an automated command-line script, for more details, see [Scripting Oxygen](on page 3196).

**Related information**
- [DITA Maps Manager](on page 2897)

**DITA Map Author Mode Actions**
A variety of actions are available for DITA map documents that can be found in **DITA** menu, toolbar, contextual menu, and the [Content Completion Assistant](on page 3217).

**DITA Map Toolbar and Menu Actions**
When a **DITA map** is opened in **Author** mode, the following default actions are available on the DITA Map toolbar (by default, they are also available in the **DITA** menu and in various submenus of the contextual menu):

- **Insert New DITA Resource**
  
  Opens a **New DITA file dialog box** *(on page 2961)* where you can choose the type of DITA document to create and inserts a reference to it at the current position within the map.

- **Insert Topic Reference**
  
  Opens the **Insert Reference dialog box** *(on page 2923)* where you can configure a topic reference and inserts it at the current position within the map.

- **Insert Key Definition with Keyword**
  
  Opens a dialog box where you can choose the name of a key and its keyword value and inserts the key definition at the current position within the map.

- **Reuse Content**
  
  Opens the **Reuse Content dialog box** *(on page 3043)* that allows you to insert and configure a content reference (@conref), or a content key reference (@conkeyref) at the cursor position.

- **Insert Topic Heading**
  
  Opens the **Insert Reference dialog box** *(on page 2923)* that allows you to insert a topic heading at the cursor position.

- **Insert Topic Group**
  
  Opens the **Insert Reference dialog box** *(on page 2923)* that allows you to insert a topic group at the cursor position.
Insert Relationship Table

Opens a dialog box that allows you to configure the relationship table to be inserted. The dialog box allows you to configure the number of rows and columns of the relationship table, if the header will be generated and if the title will be added.

Relationship Table Properties

Allows you to change the properties of rows in relationship tables.

Insert Relationship Row

Inserts a new table row with empty cells. The action is available when the cursor position is inside a table.

Insert Relationship Column

Inserts a new table column with empty cells after the current column. The action is available when the cursor position is inside a table.

Delete Relationship Column

Deletes the table column where the cursor is located.

Delete Relationship Row

Deletes the table row where the cursor is located.

Move Up

Moves the selected node up one position on its same level.

Move Down

Moves the selected node down one position on its same level.

Promote (Alt + LeftArrow)

Moves the selected node up one level to the level of its parent node.

Demote (Alt + RightArrow)

Moves the selected node down one level to the level of its child nodes.

DITA Map Contextual Menu Actions

The following actions are available in the contextual menu when editing in Author mode (most of them are also available in the DITA menu at the top of the interface):

Add File to Review Task

This action can be used to add the current document to a task in the Content Fusion Tasks Manager view. Oxygen Content Fusion is a flexible, intuitive collaboration platform designed to adapt to any type of documentation review workflow. This functionality is available through a pre-installed connector add-on (on page 2527). To fully take advantage of all of the benefits and
features of Content Fusion, your organization will need an Oxygen Content Fusion Enterprise Server. For more information, see the Oxygen Content Fusion website.

Edit Properties

Opens the Edit Properties dialog box that allows you to configure the properties of a selected node. For more details about this dialog box, see Edit Properties Dialog Box (on page 2932).

Cut (Ctrl + X (Command + X on macOS))

Removes the currently selected content from the document and places it in the clipboard.

Copy (Ctrl + C (Command + C on macOS))

Places a copy of the currently selected content in the clipboard.

Paste (Ctrl + V (Command + V on macOS))

Inserts the current clipboard content into the document at the cursor position.

Paste special submenu

This submenu includes the following special paste actions that are specific to the DITA framework:

Paste as content reference

Inserts a content reference (a DITA element with a @conref attribute) to the DITA XML element from the clipboard. An entire DITA XML element with an ID attribute must be present in the clipboard when the action is invoked. The conref attribute will point to this ID value.

Paste as content key reference

Allows you to indirectly reference content using the @conkeyref attribute. When the DITA content is processed, the key references are resolved using key definitions from DITA maps (on page 3218). To use this action, you must first do the following:

1. Make sure the DITA element that contains the copied content has an ID attribute assigned to it.
2. In the DITA Maps Manager view, make sure that the Context combo box (on page 2901) points to the correct map that stores the keys.
3. Make sure the topic that contains the content you want to reference has a key assigned to it. To assign a key, right-click the topic with its parent map opened in the DITA Maps Manager, select Edit Properties, and enter a value in the Keys field.

Paste as link

Looks for the first element with an ID value in the clipboard and inserts an <xref> that points to that element. If no elements with an ID value are found, a message
will appear that informs you that to use this action, the clipboard contents must include at least one element with a declared ID.

**Paste as link (keyref)**

Inserts a link to the element that you want to reference. To use this action, you must first do the following:

1. Make sure the DITA element that contains the copied content has an ID attribute assigned to it.
2. In the DITA Maps Manager view, make sure that the Context combo box (on page 2901) points to the correct map that stores the keys.
3. Make sure the topic that contains the content you want to reference has a key assigned to it. To assign a key, right-click the topic with its parent map opened in the DITA Maps Manager, select Edit Properties, and enter a value in the Keys field.

**Insert submenu**

This submenu includes the following insert actions that are specific to the DITA Map framework:

- **Insert New DITA Resource**
  
  Opens a New DITA file dialog box (on page 2961) where you can choose the type of DITA document to create and inserts a reference to it at the current position within the map.

- **Insert Topic Reference**
  
  Opens the Insert Reference dialog box (on page 2923) where you can configure a topic reference and inserts it at the current position within the map.

- **Insert Key Definition with Keyword**
  
  Opens a dialog box where you can choose the name of a key and its keyword value and inserts the key definition at the current position within the map.

- **Reuse Content**
  
  Opens the Reuse Content dialog box (on page 3043) that allows you to insert and configure a content reference (@conref), or a content key reference (@conkeyref) at the cursor position.

- **Insert Topic Heading**
  
  Opens the Insert Reference dialog box (on page 2923) that allows you to insert a topic heading at the cursor position.

- **Insert Topic Group**
Opens the **Insert Reference** dialog box ([on page 2923](#)) that allows you to insert a topic group at the cursor position.

**Insert Entity**

Allows you to insert a predefined entity or character entity. Surrogate character entities (range \#x10000 to \#x10FFFF) are also accepted. Character entities can be entered in one of the following forms:

- `#<decimal value>` - e.g. #65
- `&#<decimal value>` - e.g. &#65
- `#x<hexadecimal value>` - e.g. #x41
- `&#x<hexadecimal value>` - e.g. &#x41

**Relationship Table > Insert Relationship Table**

Opens a dialog box that allows you to configure the relationship table to be inserted. The dialog box allows you to configure the number of rows and columns of the relationship table, if the header will be generated and if the title will be added.

**Generate IDs**

Oxygen XML Editor generates unique IDs for the current element (or elements), depending on how the action is invoked:

- When invoked on a single selection, an ID is generated for the selected element at the cursor position.
- When invoked on a block of selected content, IDs are generated for all top-level elements and elements listed in the **ID Options** dialog box that are found in the current selection.

**Note:**

The **Generate IDs** action does not overwrite existing ID values. It only affects elements that do not already have an `@id` attribute.

**Search References**

Finds the references to the `@href` or `@keys` attribute value of the topic/map reference element at the current cursor position, in all the topics from the current DITA map ([opened in the DITA Maps Manager view ([on page 2897](#))](#)). The current topic/map reference element must have an `@href` or `@keys` attribute defined to complete the search.

**Show Key Definition**

Available for elements that have a `@conkeyref` or `@keyref` attribute set (or elements with an ancestor element that has a `@conkeyref` or `@keyref` attribute). It computes the key name and opens the **DITA map ([on page 3218](#))** that contains the definition of the key with the element that defines that key selected.

**Select submenu**
This submenu allows you to select the following:

**Element**

Selects the entire element at the current cursor position.

**Content**

Selects the entire content of the element at the current cursor position, excluding the start and end tag. Performing this action repeatedly will result in the selection of the content of the ancestor of the currently selected element content.

**Parent**

Selects the entire parent element at the current cursor position.

**Text submenu**

This submenu contains the following actions:

**To Lower Case**

Converts the selected content to lower case characters.

**To Upper Case**

Converts the selected content to upper case characters.

**Capitalize Sentences**

Converts to upper case the first character of every selected sentence.

**Capitalize Words**

Converts to upper case the first character of every selected word.

**Count Words**

Counts the number of words and characters (no spaces) in the entire document or in the selection for regular content and read-only content.

---

**Note:**

The content marked as deleted with change tracking (on page 3224) is ignored when counting words.

**Convert Hexadecimal Sequence to Character (Ctrl + Shift + X (Command + Shift + X on macOS))**

Converts a sequence of hexadecimal characters to the corresponding Unicode character (on page 463). The action can be invoked if there is a selection containing a valid hexadecimal sequence or if the cursor is placed at the right side of a valid hexadecimal sequence. A valid hexadecimal sequence can be composed of 2 to 4 hexadecimal characters and may or may not be preceded by the 0x or 0X prefix. Examples of valid sequences and the characters they will be converted to:
- 0x0045 will be converted to  ב
- 0x0125 to ง
- 265 to ɥ
- 2190 to ←

**Note:**
For more information about finding the hexadecimal value of a character, see Finding the Decimal, Hexadecimal, or Character Entity Equivalent (on page 466).

**Refactoring submenu**

Contains a series of actions designed to alter the XML structure of the document:

- **Toggle Comment**
  Encloses the currently selected text in an XML comment, or removes the comment if it is commented.

- **Move Up (Alt + UpArrow)**
  Moves the current node or selected nodes in front of the previous node.

- **Move Down (Alt + DownArrow)**
  Moves the current node or selected nodes after the subsequent node.

- **Split Element (Alt + Shift + D (Ctrl + Alt + D on macOS))**
  Splits the content of the closest element that contains the position of the cursor. Thus, if the cursor is positioned at the beginning or at the end of the element, the newly created sibling will be empty.

- **Join Elements**
  Joins two adjacent block elements (on page 3216) that have the same name. The action is available only when the cursor position is between the two adjacent block elements. Also, joining two block elements can be done by pressing the Delete or Backspace keys and the cursor is positioned between the boundaries of these two elements.

- **Surround with Tags (Ctrl + E (Command + E on macOS))**
  Allows you to choose a tag to enclose a selected portion of content. If there is no selection, the start and end tags are inserted at the cursor position.
• If the Position cursor between tags option (on page 212) is selected in the 
  Content Completion preferences page, the cursor is placed between the 
  start and end tag.
• If the Position cursor between tags option (on page 212) is not selected in 
  the Content Completion preferences page, the cursor is placed at the end of 
  the start tag, in an insert-attribute position.

🔗 Surround with [tag] (Ctrl + ForwardSlash (Command + ForwardSlash on macOS))

  Surround the selected content with the last tag used.

🔗 Rename Element

  The element from the cursor position, and any elements with the same name, can 
  be renamed according with the options from the Rename dialog box.

🔗 Delete Element Tags

  Deletes the tags of the closest element that contains the position of the cursor. 
  This operation is also executed if the start or end tags of an element are deleted by 
  pressing the Delete or Backspace keys.

🔗 Remove All Markup

  Removes all the XML markup inside the selected block of content and keeps only 
  the text content.

🔗 Remove Text

  Removes the text content of the selected block of content and keeps the markup 
  intact with empty elements.

Attributes Refactoring Actions

  Contains built-in XML refactoring operations that pertain to attributes with some of 
  the information preconfigured based upon the current context.

  Add/Change attribute

    Allows you to change the value of an attribute or insert a new one.

  Convert attribute to element

    Allows you to change an attribute into an element.

  Delete attribute

    Allows you to remove one or more attributes.

  Rename attribute

    Allows you to rename an attribute.

  Replace in attribute value
Allows you to search for a text fragment inside an attribute value and change the fragment to a new value.

**Comments Refactoring Actions**

Contains built-in XML refactoring operations that pertain to comments with some of the information preconfigured based upon the current context.

**Delete comments**

Allows you to delete comments found inside one or more elements.

**Elements Refactoring Actions**

Contains built-in XML refactoring operations that pertain to elements with some of the information preconfigured based upon the current context.

**Delete element**

Allows you to delete elements.

**Delete element content**

Allows you to delete the content of elements.

**Insert element**

Allows you to insert new elements.

**Rename element**

Allows you to rename elements.

**Unwrap element**

Allows you to remove the surrounding tags of elements, while keeping the content unchanged.

**Wrap element**

Allows you to surround elements with element tags.

**Wrap element content**

Allows you to surround the content of elements with element tags.

**Fragments Refactoring Actions**

Contains built-in XML refactoring operations that pertain to XML fragments with some of the information preconfigured based upon the current context.

**Insert XML fragment**

Allows you to insert an XML fragment.

**Replace element content with XML fragment**

Allows you to replace the content of elements with an XML fragment.

**Replace element with XML fragment**
Allows you to replace elements with an XML fragment.

**Review submenu**

This submenu includes the following actions:

- **Track Changes**
  Enables or disables the *Track Changes (on page 3224)* support for the current document.

- **Accept Change(s) and Move to Next**
  Accepts the *Track Change (on page 3224)* located at the cursor position or all of the changes in a selection and then moves to the next change. If you select a part of a *deletion or insertion* change, only the selected content is accepted.

- **Accept All Changes**
  Accepts all *Tracked Changes (on page 3224)* in the current document.

- **Reject Change(s) and Move to Next**
  Rejects the *Track Change (on page 3224)* located at the cursor position or all of the changes in a selection and then moves to the next change. If you select a part of a *deletion or insertion* change, only the selected content is rejected.

- **Reject All Changes**
  Rejects all *Tracked Changes (on page 3224)* in the current document.

- **Comment Change**
  Opens a dialog box that allows you to add a comment to an existing *Tracked Change (on page 3224)*. The comment will appear in a callout and a tooltip when hovering over the change. If the action is selected on an existing commented change, the dialog box will allow you to edit the comment.

- **Highlight**
  Enables the highlighting tool that allows you to mark text in your document.

**Colors**

Allows you to select the color for highlighting text.

**Stop highlighting**

Use this action to deactivate the highlighting tool.

**Remove highlight(s)**

Use this action to remove highlighting from the document.

- **Add Comment**
  Inserts a comment at the cursor position. The comment appears in a callout box and a tooltip (when hovering over the change).
Show/Edit Comment

Opens a dialog box that displays the discussion thread and allows the current user to edit comments that do not have replies. If you are not the author who inserted the original comment, the dialog box just displays the comment without the possibility of editing it.

Remove Comment

Removes a selected comment. If you remove a comment that contains replies, all of the replies will also be removed.

Manage Reviews

Opens the Review view (on page 664).

Folding submenu

This submenu includes the following actions:

- **Toggle Fold**
  Toggles the state of the current fold.
- **Collapse Other Folds (Ctrl + NumPad/ (Command + NumPad/ on macOS))**
  Folds all the elements except the current element.
- **Collapse Child Folds (Ctrl + NumPad. (Command + NumPad. on macOS))**
  Folds the elements indented with one level inside the current element.
- **Expand Child Folds**
  Unfolds all child elements of the currently selected element.
- **Expand All (Ctrl + NumPad* (Command + NumPad* on macOS))**
  Unfolds all elements in the current document.

About Element > Go to Definition

Moves the cursor to the definition of the current element.

Inspect Styles

Opens the CSS Inspector view (on page 640) that allows you to examine the CSS rules that match the currently selected element.

Options

Opens the Author mode preferences page (on page 176) where you can configure various options with regard to the Author editing mode.
Floating Contextual Toolbar for DITA

Oxygen XML Editor includes a dynamic feature where certain editing contexts will trigger a floating toolbar with common actions that are available in the current editing context.

The floating contextual toolbar is automatically displayed when editing DITA map documents when a `<topicref>` element is selected and it includes actions for moving the topic reference node up or down (or promoting/demoting the node).

DITA Map Drag/Drop Actions

Dragging a file from the Project view (on page 402) or DITA Maps Manager view (on page 2897) and dropping it into a DITA map document that is edited in Author mode creates a link to the dragged file (a `<topicref>` element, `<chapter>`, `<part>`, etc.) at the drop location.

Opening a Topic from a DITA Map in Author Mode

If a DITA map is open in the Author visual editing mode, you can open a referenced topic by clicking the icon to the left of the particular topic. The source topic is opened in a new tab in the main editor.

Tip:
For information about customizing Author mode actions for a particular framework (on page 3219) (document type), see the Customizing the Author Mode Editing Experience for a Framework (on page 2145) section.

Related Information:
Customizing the Author Mode Editing Experience for a Framework (on page 2145)

Opening a DITA Map With Topic Content Resolved

It is possible to open a DITA map in the main editor with all the content from the referenced topics resolved and presented in one document. To do this, select the DITA map in the DITA Maps Manager view and click the Open Map in Editor with Resolved Topics toolbar button. This opens the DITA map in the main editor area with content from all topic references expanded in-place.

Content from the resolved topics that is referenced using a `@conref` or `@conkeyref` attribute is presented as read-only by default. To edit it, you must use the Edit Reference contextual menu action to open the source topic that contains the referenced content.

Editing Referenced Content Directly

If you want to edit the referenced content directly without having to open the source document, go to Options > Preferences > Editor > Edit Modes > Author and select the Allow referenced content to be edited option (on page 179). The referenced content becomes editable in-place and saving the document will save all other modified topics.
Things to be Aware of When Enabling This Option:

- The references become editable only if the referenced topics are the root elements. If, for example, in the DITA map, there are references directly to subtopics embedded in a larger topic, those references will not be editable.
- If the content is stored in a CMS, you need to deselect the **Local files only** option (on page 179) to edit such remote referenced topics directly but this feature might not function properly with remote resources (it depends on the capabilities of the CMS connector).
- Since a single topic may be referenced in multiple places in the DITA map, be careful not to make conflicting changes to that topic.
- When modified topics are saved, the **Only modified content** option in the **Preferences > Editor > Edit Modes > Author > Serialization** page (on page 196) is ignored.
- The toolbar has two DITA map-specific actions for inserting topic references and all DITA topic-specific actions that can be used to make changes in the referenced DITA topics.
- The content completion and schema-aware insertion strategies work in each referenced topic according to their respective schema.
- The contextual menu presents the relevant actions in each referenced topic.
- Validation works for each individual referenced topic but only if it contains modifications.

Working with DITA Topics

DITA is a structured writing format. Structure can have several meanings, all of which are relevant to DITA. This section includes information about working with DITA topics and the structure.

Information Types

The structure of a piece of content refers to how the words and images are selected and organized to convey information. One approach to structured writing is to divide content into discrete blocks that contain various types of information, and then to combine those blocks to form publications. DITA is based on this approach, and encourages the author to write in discrete blocks called topics. DITA provides three base topic types (concept, task, and reference), a number of extended topic types, and the capability to create new topic types through specialization.

Text Structure

Every piece of text is made up of certain text structures, such as paragraphs, lists, and tables. DITA supports text structures through XML elements such as `<p>`, `<ol>`, and `<simpletable>`. The DITA markup specifies the text structures, but not how they will be published in various types of media. The formatting of text structures is determined by the output transformations and may be customized to meet the needs of various organizations and type of media.
Semantic Structure

Semantic structure is structure that shows the meaning of things. For example:

- A `<task>` element specifies that a block of content contains the description of a task.
- A `<codeblock>` element specifies that a block of text consists of programming code.
- A `<uicontrol>` element specifies that a word is the name of a control in a computer GUI.
- The `@platform` profiling attribute specifies that a particular piece of content applies only to certain computing platforms.

Semantic structure is important in a structured writing system because it allows both authors and readers to find content, and it allows processing scripts to process various pieces of content differently, based on their role or meaning. This can be used to do things such as filtering content related to a specific product so that you can produce documentation on many products from the same source.

There can be many forms of semantics captured in a document set. DITA captures some of these in topics and some of them in maps. If you are using a CMS, it may capture additional semantics.

Document Semantics

Documents consist of elements that may be made up of the same basic text structures as the rest of the text, but have a special function within the structure of the document. For instance, both tables of contents and indexes are lists, but they play a special role in the document. Chapters and sections are just sequences of paragraphs and other text structures, yet they are meaningful in the structure of the document. In some cases, such as indexes and tables of contents, these structures can be generated from semantic information embedded in the source. For instance, a table of contents can be built by reading the titles of chapters and sections. DITA provides elements to describe common document semantics.

Subject Matter Semantics

In some cases, the semantics of the content relate directly to the subject matter that the content describes. For instance, DITA supports tags that allow you to mark a piece of text as the name of a window in a software application (`<wintitle>`), or to mark a piece of text as applying only to a particular product.

Audience Semantics

In some cases, the semantics of the content relate to the audience that it is addressed to. For instance, a topic might be addressed to a particular role, or to a person with a particular level of experience. DITA provides an `<audience>` element to capture audience metadata.

Creating Topic Structures

Oxygen XML Editor provides a number of tools to help you create topic structures:

- **Content Completion Assistant (on page 3217)** - Shows you which elements can be created at the current position.
- **Model view (on page 545)** - Shows you the complete structure supported by the current element.
• **Outline view (on page 539)** - Shows you the current structure of your document.
• **DITA toolbar** - Helps you to easily insert many common structures.

**Resources**

For more information about getting started with DITA and how to work with DITA in Oxygen XML Editor, see our compiled collection of DITA-related webinars that are meant to help you with your journey into working with DITA: [Webinars: Working with DITA in Oxygen](#).

**Related information**

- [Getting Started with DITA (on page 2887)](#)
- [DITA Topics Document Type (Framework) (on page 1337)](#)

**Creating a New DITA Topic**

The basic building block for DITA information is the DITA topic. DITA provides a variety of specialized topic types, the most common of which are:

- **Topic** - The base topic type from which all other topic types are specialized. Typically, it is used when a more specialized topic type is inappropriate.
- **Task** - For procedural information such as how to use a dialog box.
- **Concept** - For general, conceptual information such as a description of a product or feature.
- **Reference** - For reference information.

Oxygen XML Editor also supports numerous other specialized topic types that you will find templates for in the various folders in the [New DITA file dialog box (on page 2962)](#). They include DITA 1.3 specializations, Lightweight DITA templates, MathML composites, Markdown documents, and other DITA specialized topic and [DITA map (on page 3218)](#) types such as **Glossentry**, **Troubleshooting**, **Bookmap (on page 3216)**, and **Subject Scheme Map (on page 3224)**.

To create a new DITA topic and add a reference to it in your [DITA map (on page 3218)], follow these steps:

1. In the [DITA Maps Manager (on page 2897)](#), right-click the node in the current map where you want to add the new topic.

2. Select one of the following actions:
   - **Append Child > New** - Select this action to insert the new topic as a child of the selected node. This action opens a [New file dialog box (on page 2962)](#) that allows you to select the type of document and assists you with naming it. After you have configured your new topic, click **Create**.
   - **Insert Before > New** - Select this action to insert the new topic as a sibling to the current node, before it. This action opens a [New file dialog box (on page 2962)](#) that allows you to select the type of document and assists you with naming it. After you have configured your new topic, click **Create**.
Insert After > New - Select this action to insert the new topic as a sibling to the current node, after it. This action opens a New file dialog box (on page 2962) that allows you to select the type of document and assists you with naming it. After you have configured your new topic, click Create.

Duplicate - Select this action to create a copy of the selected topic and insert it as a sibling. This action opens a dialog box that allows you to choose the file name and location for the newly created copy of the topic. After you have selected the name and path for your new topic, click OK.

Note:
The value of the root ID is generated taking the Use the file name as the value of the root ID attribute option from the DITA > Topics preferences page (on page 273) into account. When the option is deselected, a unique ID is generated.

Step Result: The new topic is now referenced (as a <topicref>) in the DITA map at the location where you inserted it and the new topic is opened in the editor.

3. Save the DITA map.

New DITA File Dialog Box

The New DITA file dialog box allows you to create a new DITA topic using various types of DITA file templates and provides some options that help you to configure the new topic.
Figure 751. New DITA File Dialog Box

The templates that appear in this dialog box include all templates that have an associated .properties file and the type property is set to dita, as well as templates that do not have an associated properties file or the type property is not defined. It will also include custom templates that you create using the procedures presented in Creating New Document Templates (on page 375).

The New DITA file dialog box includes the following features and options:

Choose a file template

Use the template preview pane to select the appropriate type of DITA file you want to create. Once you select a template, the other options will appear below the preview pane.

Tip:
You can use the text filter field at the top of the dialog box to search for a specific template.
Title

Depending on the selected file template, the value of the Title field is set in:

- The `<title>` element of a DITA topic file. The `<title>` element needs to be the first child of the root element.
- The `<glossterm>` element of a Glossentry file.

New Topics Preferences

Pressing this button opens the DITA New Topics preference page (on page 273).

Save as

Use this option to specify a file name and path for the new file. You can specify the path by using the text field, the history drop-down, or the browsing actions in the Browse drop-down menu.

Create

When you click this button, a reference (`<topicref>`) to the new topic is added to the current DITA map and the new topic is opened in the editor.

Related Information:

- Getting Started with DITA (on page 2887)
- Adding Topics to a DITA Map (on page 2917)
- Working with Markdown Documents in DITA (on page 3024)
- Fast Create Multiple DITA Topics (on page 2964)

Fast Create Multiple DITA Topics

The DITA Maps Manager (on page 2897) includes a feature that allows you to quickly create multiple skeleton topics at once and you can specify their hierarchical structure within the DITA map (on page 3218). A common use-case for using this feature is when you need to insert a new chapter or section that will include multiple topics and you have the structure and titles planned out in advance.

Note:

The Fast Create Topics feature works for the following types of local and remote resource protocols: file, http, https, ftp, ftps.

To access this feature, right-click a node in the DITA Maps Manager where you want the new topics to be inserted and select Fast Create Topics. This opens the Fast Create Topics dialog box where you can configure the structure for the new topics.
The Fast Create Topics dialog box includes the following features and options:

**Hierarchy Text Pane**

Use this text area to enter the titles for your new topics, one per line, and specify the hierarchy by using indents (Tab or Space). Topic references will be created in the DITA map according to the hierarchy you enter in this section.

**File name generation rules**

The titles that you enter in the text pane will not only be used for the topic titles but also to generate their file names and you can click the File name generation rules link to configure the rules (on page 273) for how those file names will be generated.

**Tip:**

If you have added a file name prefix or suffix to the properties file (on page 378) for DITA document templates, the generated file name will include that prefix or suffix.
**File name prefix**

Use this option to add a specified prefix to the file name. If you have added a file name prefix to the properties file (on page 378) for DITA document templates, the prefix you enter here will override the one from the properties file.

**File name suffix**

Use this option to add a specified suffix to the end of the file name. If you have added a file name suffix to the properties file (on page 378) for DITA document templates, the suffix you enter here will override the one from the properties file.

**Topic type**

All of the topics that will be created will have the same DITA topic type, which is detected from the most recently created topic. You can click the Change button to select a different type from a list of possible DITA templates.

Tip:

You can convert any of these new files to a different DITA topic type at a later time by using another feature that allows you to easily convert DITA documents to other types (on page 2970).

**Add created topic hierarchy relative to the selection as**

By default, the hierarchy of topics will be added to the DITA map as the First Child of the node where the action was invoked. You can change this to Last Child, Preceding Sibling, or Following Sibling if the selected node allows topics to be inserted as such.

**Create**

When you click Create, the specified hierarchy is added as topic references in the DITA map. The new documents are created as bare skeleton topics with only the topic title and possibly the root ID populated.

Tip:

You can easily change the order of the topics in the DITA map (on page 2916) at a later time,

**Related Information:**

- Adding Topics to a DITA Map (on page 2917)
- Converting DITA Topics to Another Type (on page 2970)
Editing DITA Topics

Oxygen XML Editor provides a number of features to help you edit DITA topics. A DITA topic is an XML document, thus all the editing features that Oxygen XML Editor provides for editing XML documents also apply to DITA topics. Oxygen XML Editor also provides extensive additional support specifically for DITA.

Opening a DITA Topic

There are several ways to open a DITA topic in the XML editor. Use any of the following methods to open a topic:

- Double-click the topic in the DITA Maps Manager (on page 2897) (or right-click the topic and select Open).
- Double-click the file in the Project view (on page 402) (or right-click the file and select Open).
- If you have a DITA map (on page 3218) opened in the XML editor, you can click the icon to the left of the topic.
- Drag a DITA file from your system browser and drop it in the XML editor.

Visual Editing in Author Mode

DITA is an XML format (on page 34), although you do not have to write raw XML to create and edit DITA topics. Oxygen XML Editor provides a graphical view of your topics in Author mode (on page 354). Your topics will likely open in Author mode by default, so this is the first view you will see when you open or edit a DITA topic. If your topic does not open in Author mode, just click Author at the bottom left of the editor window to switch to this mode.

Author mode presents a graphical view of the document you are editing, similar to the view you would see in a word processor. However, there are some differences, including:

- **Author** mode is not a WYSIWYG view. It does not show you exactly what your content will look like when printed or displayed on-screen. The appearance of your output is determined by the DITA publishing process, and your organization may have modified that process to change how the output is displayed. Oxygen XML Editor has no way of determining what your final output will look like or where line breaks or page breaks will fall. Treat Author mode as a friendly visual editing environment, not a faithful preview of your output.
- Your document is still an XML document. Author mode creates a visual representation of your document by applying a CSS stylesheet to the XML. You can see the XML at any time by switching to Text mode (on page 353). You, or someone in your organization, can change how the Author view looks by changing the CSS stylesheet or providing an alternate stylesheet.
- Your aim in editing a DITA document is not to make it look right, but to create a complete and correct DITA XML document. Author mode keeps you informed of the correctness of your content by highlighting XML errors in the text and showing you the current status in a box at the top right of the editor window. Green means that your document is valid, yellow means valid with warnings, and red means invalid. Warnings and errors are displayed when you place the cursor on the error location.
Your XML elements may have attributes set on them. Conventionally, attributes are used to contain metadata that is not displayed to the reader. By default, attributes are not displayed in the Author view (though there are some exceptions) and cannot be edited directly in the Author view (though in some cases the CSS that drives the display may use form controls to let you edit attributes directly). To edit the attributes of an element, place your cursor on the element and press Alt+Enter to bring up the attribute editor. Alternatively, you can use the Attributes view (on page 627) to edit attributes.

**Tip:**
You can select Hints from the Styles drop-down menu (available on the Author Styles toolbar) to display tooltips throughout the DITA document that offers additional information to help you with the DITA structure. For more information, see the Selecting and Combining Multiple CSS Styles (on page 2145) section.

### Content Completion Assistance

Since it is a structured format, DITA only allows certain elements in certain places. The set of elements allowed differs from one DITA topic type to another (this is what makes one topic type different from another). To help you figure out which elements you can add in any given place and help you understand what they mean, Oxygen XML Editor has a number of Content Completion Assistant (on page 3217) features.

- **The Enter key:** In Author mode, the Enter key does not create line breaks, it brings up the Content Completion Assistant to help you enter a new element. In XML, you do not use line breaks to separate paragraphs. You create paragraphs by creating paragraph elements (element `p` in DITA) and tools insert the line breaks in the output and on-screen.

### Figure 753. Content Completion Assistant

The Content Completion Assistant not only suggests new elements you can add. If you press Enter at the end of a block element (on page 3216) (such as a paragraph) it suggests creating a new element of the same type. If you press Enter in the middle of a block element, it suggests splitting that element into two elements.

A useful consequence of this behavior is that you can create a new paragraph simply by hitting Enter twice (just as you might in a text editor).

As you highlight an element name, a basic description of the element is displayed. Select the desired element and press Enter to create it.
To wrap an element around an existing element or piece of text, simply select it and press **Enter** and use the *Content Completion Assistant* to choose the wrapper element.

- **The Model view:** You can see the entire model of the current element by opening the *Model view (on page 545)* (Window > Show View > Model, if the view is not already open). The *Model* view shows you what type of content the current element can contain, all the child elements it can contain, all its permitted attributes, and their types.

---

**Tip:**
You can also select **Inline actions** from the Styles drop-down menu (available on the *Author Styles* toolbar) to display possible elements that are allowed to be inserted at various locations throughout the DITA document. For more information, see the *Selecting and Combining Multiple CSS Styles (on page 2145)* section.

---

**DITA Editing Actions**

A variety of actions are available in the DITA *framework (on page 3219)* to specifically assist you with editing DITA documents. These various actions are available in the contextual menu, the DITA menu, the DITA (Author Custom Actions) toolbar, or the *Content Completion Assistant*.

The DITA toolbar contains buttons for inserting a number of common DITA elements (elements that are found in most DITA topic types).

If the DITA toolbar is not displayed, right-click anywhere on the toolbar area, select **Configure Toolbars**, and select it from the displayed dialog box.

---

**Note:**
The DITA toolbar contains a list of the most common elements and actions for DITA, such as inserting an image, creating a link, inserting a content reference, or creating a table. It does not contain a button for inserting every possible DITA element. For a complete list of elements that you can insert at the current location in your document, press **Enter** to open the *Content Completion Assistant*.

Whenever the current document in the editor is a DITA document, the DITA menu is displayed in the menu bar. It contains a large number of actions for inserting elements, creating content references and keys, editing DITA documents, and controlling the display. These actions are specific to DITA and supplement the general editing commands available for all document types. Many of these actions are also conveniently available in the contextual menu. In addition to the DITA framework-specific actions (on page 3001), the contextual menu also includes various *general Author mode contextual menu actions (on page 759).*
Converting DITA Topics to Another Type

Oxygen XML Editor includes a feature that allows you to convert an existing DITA document to a different topic type. For example, if you want to convert a DITA Task to a DITA Topic, or vice versa. There are several ways to access these refactoring actions and you can choose a scope for the operation and some filtering options.

**DITA Conversion Refactoring Operations for DITA**

The following conversion operations are available:

- **Convert Nested Topics to New Topics (Available from the contextual menu of editable maps/nodes in the DITA Maps Manager (on page 2897))**
  
  Use this operation on topics that contain nested `<topic>` elements to convert each nested topic to a new topic. Also, the new topics are added in the DITA Maps Manager as the first child topics of the original topic.

- **Convert Sections to New Topics (Available from the contextual menu of editable maps/nodes in the DITA Maps Manager (on page 2897))**
  
  Use this operation on topics that contain multiple sections to convert each section to a new topic. Also, the new topics are added in the DITA Maps Manager as the first child topics of the original topic.

- **Convert to Concept**
  
  Use this operation to convert a DITA topic (of any type) to a DITA Concept topic type (for example, Topic to Concept).

- **Convert to General Task**
  
  Use this operation to convert a DITA topic (of any type) to a DITA General Task topic type (for example, Task to General Task). A DITA General Task is a less restrictive alternative to the Strict Task information type.

- **Convert to Reference**
  
  Use this operation to convert a DITA topic (of any type) to a DITA Reference topic type (for example, Topic to Reference).

- **Convert to Task**
  
  Use this operation to convert a DITA topic (of any type) to a DITA Task topic type (for example, Topic to Task).

- **Convert to Topic**
Use this operation to convert a DITA topic (of any type) to a DITA Topic (for example, Task to Topic).

**Convert to Troubleshooting**

Use this operation to convert a DITA topic (of any type) to a DITA Troubleshooting topic type (for example, Topic to Troubleshooting).

### Methods for Accessing the DITA Conversion Refactoring Operations

To access the conversion operations, use one of the following methods:

**Single Document Method**

With the document opened in the editor, right-click anywhere in the main editing pane (or right-click the topic reference in the [DITA Maps Manager](on page 2897)), go to the **Refactoring** submenu, and choose whichever operation is appropriate for your needs.

**Multiple Documents At Once Method**

Select **XML Refactoring** from the **Tools** menu (or from the **Refactoring** submenu when you right-click one or more documents in the [Project view](on page 402) or the [DITA Maps Manager view](on page 2897)). Then select whichever operation is appropriate for your needs.

### XML Refactoring Wizard Dialog Box

When you select any of the operations, Oxygen XML Editor proceeds to the **XML Refactoring Wizard**. If you used the **Multiple Documents At Once Method** (on page 2971), the wizard page allows you to choose a scope for the operation and some filtering options:

- **Scope** - Select from a variety of options to define the scope that will have resources affected by the operation. For example, you can choose to affect all resources in the Project, All opened files, Current DITA map hierarchy, Selected reference, and others depending on the context.

- **Filters** section
  - **Include files** - Specifies files to be excluded from the operation. You can specify multiple files by separating them with commas and the patterns can include wildcards (such as * or ?).
  - **Restrict to known XML file types only** - Excludes non-XML file types from the operation.
  - **Look inside archives** - If this option is selected, the scope of the operation will include files inside archives.

If you used the **Single Document Method** (on page 2971), the scope will be the current file so the scope and filtering options are not displayed.

You can then use one of the following buttons to proceed with the operation:

- **Preview**
  
  You can use the Preview button to open a comparison panel where you can review all the changes that will be made by the refactoring operation before applying the changes.
Warning:

It is always recommended to use the Preview button to make sure the operation is not going to do something unexpected and after you click the Finish button, any Undo action will only revert changes on the current document.

Finish

When you use the Finish button, behind the scenes Oxygen XML Editor maps the structure of the previous DITA document type to a structure that fits the new type. In some cases, especially when the previous structure was very complex, the conversion might result in an invalid structure and some manual adjustments might be required.

Handling Special Characters When Generating New File Names

For refactoring operations that generate a new file, if special characters are detected in an element that will be used to generate the new file name, the special characters will automatically be replaced with their ASCII equivalents (for example, Ä is changed to AE). If an ASCII equivalent does not exist, it will be replaced with an underline character (_). The purpose of this functionality is to avoid generating invalid file names.

It is possible to customize the list of replaceable symbols by editing the following XSLT character map file:

```
{OXYGEN_INSTALL_DIR}/frameworks/dita/refactoring/utils/character-map.xsl
```

Converting To and From DITA Specialization Document Types

If you use your own DITA specialization document type (on page 3176), you can modify mappings for the predefined conversion operations to work with your specialization.

To use the conversion operations with your DITA specialization, follow these steps:

1. Locate the conversion stylesheets in the following directory (and its subdirectories):

   `{OXYGEN_INSTALL_DIR}/frameworks/dita/refactoring/`

   **Note:**
   
   The stylesheets for converting entire files (from one type to another) are located in the `dita-files-conversion-stylesheets` folder. Each of these conversion operations has a stylesheet with the word `entrypoint` at the end of its name. Edit the appropriate `*-entrypoint.xsl` file (for example, to modify the Convert to Task operation, edit the `convert-resource-to-task-entrypoint.xsl` file).

2. Depending on whether you use a DTD, XML Schema, or Relax NG-based specialization, you can:
a. Modify the values of the declared root-element, public-literal-target, and system-literal-target variables to match your specialization's DTD information.
b. Modify the value of the declared schema-location variable to match the location of your specialization's XML schema.
c. Modify the value of the declared xml-model-location variable to match your Relax NG specialization.

3. For the Convert Nested Topics to New Topics and Convert Sections to New Topics operations, if your DITA specialization uses your own custom URN or DOCTYPE, you can replace the default mappings in the \[OXYGEN_INSTALL_DIR\]/frameworks/dita/refactoring/utils/dita-formats.xsl stylesheet with your own values for the DOCTYPE or xml-model.

4. If you want to change the name of the operation that will be displayed in Oxygen XML Editor, follow these substeps:
   a. Locate the resource XML file for the same conversion operation in the following directory:
      \[OXYGEN_INSTALL_DIR\]/frameworks/dita/refactoring/ (for example, for the Convert to Task operation, it is convertResource2Task.xml).
   b. Edit that XML file and change the name attribute to match whatever you want to be displayed for that operation (for example, name="Convert to My DocType").

5. Save your changes to all modified files.
6. Restart Oxygen XML Editor

Result: You should now see your changes when accessing the conversion operations (on page 2971).

Tip:
You can also create your own customized refactoring operations. For more information, see Custom Refactoring Operations (on page 856).

Related Information:
Editing DITA Topics (on page 2967)
Refactoring XML Documents (on page 840)

Working with Images in DITA Topics

There are several ways to add images to a DITA topic, depending on if you want to create a figure element (with a title and caption), just insert an image inline, or if you want to use multiple versions of a graphic depending on the situation. For instance, you might want to use a specific image for each different product version or output media.

Adding an Image Inline with the Insert Image Dialog Box

Use the following procedure to add an image inline:
1. Place the cursor in the position you want the graphic to be inserted.

2. Select the **Insert Image** action. The **Insert Image** dialog box appears.

![Figure 754. Insert Image Dialog Box](image)

3. Configure the options in this dialog box and click **Insert**.

The **Insert Image** dialog box includes the following options and features for inserting images into a DITA document:

**Location**

Use this option to specify a URL for the image as the value of an `@href` attribute inside the `<image>` element. You can type the URL of the image you want to insert or use browsing actions in the **Browse** drop-down menu (there is also a history drop-down).

**Key**

Use this option to insert the selected key as the value of a `@keyref` attribute inside the `<image>` element. All keys that are presented in the dialog box are gathered from the root map (on page 3223) of the current DITA map. You can use the **Choose Key Reference** button to open the **Choose Key** dialog box that presents the list of keys available in the selected root map.

**Note:**

If your defined keys are not listed in this dialog box, it is most likely trying to gather keys from the wrong root map. You can change the root map by using the
Change Root Map link in the Choose Key dialog box or change it in the Context option in the toolbar of the DITA Maps Manager.

Figure title

Use this text box to insert a `<title>` and `<image>` element inside a `<fig>`.

Alternate text

Use this text box to insert an `<alt>` element inside the `<image>`.

Size

Use this section to configure the **Width** and **Height** of the image, or **Scale** the image. Specifying a value in these options inserts a `@width`, `@height`, and `@scale` attribute, respectively.

Layout

Use the options in this section to insert `@placement` and `@align` attributes into the `<image>` element.

Preview

The **Preview** box shows a thumbnail of the selected image so that you can see a preview of the image before clicking **Insert**.

Adding an Image Inline with Drag/Drop (or Copy/Paste) Actions

You can drag images from your system explorer or the **Project view** (on page 402) and drop them into a DITA document (or copy and paste). This will insert the path of the image file as the value of the `@href` attribute in a DITA `<image>` element:

```xml
<image href="../images/image_file.png"/>
```

**Tip:**

To replace an image, just drag and drop a new image over the existing one. Oxygen XML Editor will automatically update the reference to the new image.

Adding an Image in a Figure Element

To add an image in a figure:

1. Add a `<fig>` element to your document at the appropriate place.
2. Add a `<title>` and/or `<desc>` element to `<fig>`, according to your needs.
3. Add an `<image>` element (on page 2973) to the `<fig>` element.

**Note:**

The `<fig>` element has a number of other child elements that may be appropriate to your content. See the DITA documentation for complete information about `<fig>`.
Note:
The order that the content of the `<image>`, `<title>`, and `<desc>` elements will appear in the output is determined by the output transformation. If you want to change how they appear, you may have to modify the output transformation, rather than your source content.

Floating Images in DITA Topics for PDF or XHTML Output

Oxygen XML Editor provides the possibility of floating an image to the left or right of blocks of content in DITA topics, for both PDF and XHTML output.

To float an image, you simply need to set the `@outputclass` attribute on the `<image>` element. The possible values are:

- `float-left`
- `float-right`

For example, the following DITA structure will present the image to the right of the paragraph content in the output:

```dita
<p>
  <image href="../../images/Lilac.jpg" scale="45" outputclass="float-right"/>
  <b>Lilac</b> (<b>Syringa</b>) is a genus of about 20–25 species of flowering plants in the olive family (<b>Oleaceae</b>), native to Europe and Asia.
</p>
```

Figure 755. Image Floated to the Right

Searching for References to Images

You can search for all references to an image by selecting `Search References` from the contextual menu. The result depends on how the image is defined, as follows:

- If the action is invoked on an `<image>` element that contains an `@href` attribute but does not include an `@id` attribute, all direct references to the image are reported. If the `<image>` element does have an `@id` attribute, all links to the specified ID are also reported.
- If the action is invoked on an `<image>` element that contains a `@keyref` attribute but does not include an `@id` attribute, all direct references to the image are reported along with all instances where the key is used. If the `<image>` element does have an `@id` attribute, all links to the specified ID are also reported.
Adding Video, Audio, and Embedded HTML Resources in DITA Topics

You can insert references to media resources (such as videos, audio clips, or embedded HTML frames) in your DITA topics. The media resources can be played directly in Author mode and in all HTML5-based outputs. There is a toolbar button that allows you to insert and configure a reference to the media resource. You can also drag media files from your system explorer or the Project view and drop them into your documents (or copy and paste them).

**Table 53. Supported Media Types**

<table>
<thead>
<tr>
<th>Media</th>
<th>Description</th>
<th>Type</th>
<th>Supported Size Properties</th>
</tr>
</thead>
<tbody>
<tr>
<td>mp3</td>
<td>Moving Picture Experts Group Layer-3 Audio</td>
<td>audio</td>
<td>Width</td>
</tr>
<tr>
<td>wav</td>
<td>Windows Wave</td>
<td>audio</td>
<td>Width</td>
</tr>
<tr>
<td>pcm</td>
<td>Pulse Code Modulation</td>
<td>audio</td>
<td>Width</td>
</tr>
<tr>
<td>m4a</td>
<td>Moving Picture Experts Group Layer-4 Audio</td>
<td>audio</td>
<td>Width</td>
</tr>
<tr>
<td>aif</td>
<td>Audio Interchange Format</td>
<td>audio</td>
<td>Width</td>
</tr>
<tr>
<td>mp4</td>
<td>Moving Picture Experts Group Layer-4 Video</td>
<td>video</td>
<td>Width &amp; Height</td>
</tr>
<tr>
<td>m4v</td>
<td>Itunes Video File</td>
<td>video</td>
<td>Width &amp; Height</td>
</tr>
<tr>
<td>avi</td>
<td>Audio Video Interleaved</td>
<td>video</td>
<td>Width &amp; Height</td>
</tr>
<tr>
<td>embedded video (such as YouTube or Vimeo)</td>
<td>Embedded Iframe Code</td>
<td>iframe</td>
<td>Width &amp; Height</td>
</tr>
</tbody>
</table>

**Adding a Media Resource**

To insert a media resource in a DITA document, use the following procedure:
1. Place the cursor at the location where you want the media resource.

2. Select the **Insert Media Resource** action from the toolbar. The **Insert Media** dialog box appears.

   ![Insert Media Dialog Box](image)

   **Note:**
   You can also drag media files from your system explorer or the [Project view](on page 402) and drop them into your documents (or copy and paste them). Note that this method will bypass the **Insert Media** dialog box, so if you need to adjust the size you will need to adjust the @width or @height attributes manually.

3. Configure the options in this dialog box and click **Insert**.

   The **Insert Media** dialog box includes the following options:

   **Location**
   
   Use this option to specify a URL for the media resource as the value of a @data attribute inside the `<object>` element. You can type the URL of the resource you want to insert or use browsing actions in the [Browse drop-down menu](on page 402) (there is also a history drop-down).

   **Key**
   
   Use this option to insert the selected key as the value of a @datakeyref attribute inside the `<object>` element. All keys that are presented in the dialog box are gathered from the root map (on page 3223) of the current DITA map. You can use the [Choose Key Reference](on page 3223) button to open the Choose Key dialog box that presents the list of keys available in the selected root map.
Note:
If your defined keys are not listed in this dialog box, it is most likely trying to
gather keys from the wrong root map. You can change the root map by using the
Change Root Map link in the Choose Key dialog box or change it in the Context
option in the toolbar of the DITA Maps Manager.

Type
Oxygen XML Editor detects and automatically selects the media type based upon the
specified resource in the URL field (on page 2978). You can manually change the type,
but keep in mind that in the publishing stage the object element is converted to an HTML5
element (on page 2979) based upon the type selected here. You can choose between:
audio, video, or iframe.

Size
Use this section to configure the Width and Height of the frame for the media resource.
Specifying a value in these options inserts a @width and @height attribute, respectively. For
audio clips, only the Width can be adjusted.

Result in Author Mode: A reference to the specified video, audio, or embedded HTML frame is inserted in an
<object> element and it is rendered in Author mode so that it can be played directly from there.

Attention:
• On Ubuntu 17.10, if you receive an error when trying to play videos in Author mode, you need to
install the libavformat57 library.

Result in Output: In the publishing stage, the object element is converted to an HTML5 element so that it can
be rendered properly and played in all HTML5-based outputs.

• Videos - The object element is converted to an HTML5 <video> element.
• Audio Clips - The object element is converted to an HTML5 <audio> element.
• Embedded HTML Frames - The object element is converted to an HTML5 <iframe> element.

Tip:
There is an even faster way of inserting an embedded video (such as a YouTube or Vimeo). If you
copy the embed code from the source (for example, you can right-click on a YouTube video and select
Copy embed code), you can then paste the contents of the clipboard in the URL field (on page 2978)
and the Type (on page 2979) will automatically be set on iframe, while the Width and Height (on
page 2979) will be populated according to the detected size, and an allowfullscreen parameter will
automatically be added (set the value of this parameter to true to allow videos to play in full screen mode once the document is converted to XHTML output).

Inserting Media in HTML Outputs That Do Not Support Embedded Media

For certain types of HTML output (for example, CHM) that do not support embedded media (such as videos or audio files), Oxygen XML Editor provides a parameter that can be set in the transformation scenario to present the media object as a plain link in the output.

This can be achieved by following these steps:

1. Edit the DITA transformation scenario (on page 3106) for the output type that does not support embedded objects (for example, DITA Map CHM).
2. Go to the Parameters tab (on page 3112) and click the New button to add a new parameter.
3. For the Name, enter: com.oxygenxml.xhtml.linkToMediaResources.
4. For the Value, enter: true.
5. Click OK and continue on to run the transformation.

Result: The media object will appear in the output as a plain link instead of an embedded object.

For more information, see the following video demonstration:

https://www.youtube.com/embed/llX11gS4WaU

Related Information:

Working with Images in DITA Topics (on page 2973)

How to Add Video and Audio Objects in DITA WebHelp Output (on page 1674)

Image Maps in DITA

Oxygen XML Editor includes support for image maps in DITA documents through the use of the <imagemap> element. This feature provides an easy way to create hyperlinks in various areas within an image without having to divide the image into separate image files. The visual Author editing mode includes an Image Map Editor that helps you to easily create and configure image maps.
Image Map Editor Interface in DITA

The interface of the **Image Map Editor** consists of the following sections and actions:

**Toolbar**

- **New Rectangle**
  
  Use this button to draw a rectangular shape over an area in the image. You can drag any of the four points to adjust the size and shape of the rectangle.

- **New Circle**
  
  Use this button to draw a circle over an area in the image. You can drag any of the four points to adjust the size of the circle.

- **New Polygon**
  
  Use this button to draw a polygon shape over an area in the image. This action opens a dialog box that allows you to select the number of points for the polygon. You can drag any of the points to adjust the size and shape of the polygon.

- **New Free Form Shape**
  
  Use this button to draw a free form shape over an area in the image. After selecting this button, left-click anywhere in the image to place the first point of your shape.
Then move the cursor to the location of the next desired point and left-click to place the next point, and so on. To complete the shape (area), click the first point again and a line will automatically be added from the last point that was added, or simply double-click the last point to automatically add the line from the last point back to the first.

**Duplicate**

Use this button to create a duplicate of the currently selected shape.

**Delete**

Use this button to delete the currently selected shape.

**Undo**

Use this button to undo the last action.

**Redo**

Use this button to redo the last action that was undone.

**Show/Hide Numbers**

Use this button to toggle between showing or hiding the numbers for the shapes.

**Bring Shape to Front**

Use this button to bring the currently selected shape forward to the top layer.

**Bring Shape Forward**

Use this button to bring the currently selected shape forward one layer.

**Send Shape Backward**

Use this button to send the currently selected shape back one layer.

**Send Shape to Back**

Use this button to send the currently selected shape back to the bottom layer.

**Color Chooser**

Use this drop-down menu to select a color scheme for the lines and numbers of the shapes.

**Zoom Slider**

Use this slider to zoom the image in or out in the main image pane.

**Image Pane**

This main Image Pane is where you work with shapes to add hyperlinks to multiple areas within an image. The editing mechanisms that are supported in the Image Pane include the following:
Mouse Controls and Keyboard Shortcuts

• Use the mouse to select and move shapes around in the image pane. It is easy to see which shape is selected in this image pane because the border of the selected shape changes from a solid line to a dotted one.
• You can also drag any of the points of a selected shape to adjust its size and shape.
• You can hold down the Ctrl key to select multiple shapes and then move them simultaneously.
• You can also move shapes by using the arrow keys on your keyboard. In addition, you can hold down Shift while using the arrow keys to move the shape further or Alt to move it 1 pixel at a time.
• To zoom in or out, you can use the NumPad + or NumPad - keys respectively. Use Ctrl + NumPad 0 to reset the zoom level to its default value.
• You can use Ctrl + Z to undo an action or Ctrl + Y to redo the last action that was undone.

Contextual Menu Actions Available in the Image Pane

You can right-click the shapes, points, or anywhere in the Image Pane to invoke the contextual menu where the following actions are available:

+ Add Point
  Adds a point to Polygon or Free Form shapes.

✗ Remove Point
  Removes the current point from Polygon or Free Form shapes.

☑ Duplicate
  Create a duplicate of the currently selected shape.

✗ Delete
  Delete the currently selected shape.

☑ New Rectangle
  Creates a rectangular shape over an area in the image. You can drag any of the four points to adjust the size and shape of the rectangle.

☑ New Circle
  Creates a circle over an area in the image. You can drag any of the four points to adjust the size of the circle.

☑ New Polygon
Creates a polygon shape over an area in the image. This action opens a dialog box that allows you to select the number of points for the polygon. You can drag any of the points to adjust the size and shape of the polygon.

Undo

Use this action to undo the last action.

Redo

Use this action to redo the last action that was undone.

Shape Table

The table at the right of the Image Pane is a sequential list of all the areas (shapes) that have been added in the image. It shows their number, type, and description (if one has been added). If you select one of the entries in the table, the corresponding shape will be selected in the Image Pane.

Properties

Type

Displays information about the selected coordinate.

Target

Allows you to choose the target resource that you want the selected area (shape) to be linked to. Select a target by using the Link drop-down menu to the right of the text field. You can choose between the following types of links: Cross Reference, File Reference, or Web Link. All three types will open a dialog box that allows you to define the target resource. This linking process is similar to the normal process of inserting links in DITA (on page 3073) by using the identical Link drop-down menu from the main toolbar.

When you click OK to finalize your changes in the Image Map Editor, an `<xref>` element will be inserted with either an `@href` attribute or a `@keyref` attribute. Additional attributes may also be inserted and their values depend on the target and the type of link. For details about the three types of links and their dialog boxes, see Inserting a Link in Oxygen XML Editor (on page 3073).

Description

You can enter an optional description for the selected area (shape) that will be displayed in the Image Map Details section (on page 2985) in Author mode and as a tooltip message when the end-user hovers over the hyperlink in the output.

How to Create an Image Map in DITA

To create an image map on an existing image in a DITA document, follow these steps:
1. Right-click the image and select **Image Map Editor**.

   **Step Result:** This action will apply an *image map* to the current image and open the **Image Map Editor** dialog box.

2. Add hyperlinks to the image by selecting one of the shape buttons ( □ New Rectangle, ○ New Circle, or ○ New Polygon).

3. Move the shape to the desired area in the image and drag any of the points on the shape to adjust its size or form. You can use the other buttons on the toolbar (on page 2981) to adjust its layer and color, or to perform other editing actions.

   **Tip:**
   You can right-click any of the points, shapes, or anywhere in the Image Pane to access various helpful contextual menu actions (on page 2983). For example, the easiest way to remove a point is to right-click the point and select × Remove Point.

4. With the shape selected, use one of the linking options (on page 2984) in the Link drop-down menu to select a target resource (or enter its path in the Target (on page 2984) text field).

5. (Optional) Enter a Description (on page 2984) for the selected area (shape).

6. If you want to add more hyperlinks to the image, select a shape button again and repeat the appropriate steps.

7. When you are finished creating hyperlinks, click OK to process your changes.

   **Result:** The *image map* is applied on the image and the appropriate elements and attributes are automatically added. In Author mode, the image map is now rendered over the image. If the image includes an `<alt>` element, its value will be displayed under the image. The following two buttons will also now be available under the image in Author mode:

   - **Image Map Editor** - Click this button to open the **Image Map Editor**.
   - **Image Map Details** - Click this button to expand a section that displays the details of the image map and allows you to change the shape and coordinates of the hyperlinked areas. Keep in mind that if you change the shape in this section, you also need to add or remove coordinates to match the requirements of the new shape.

**Figure 758. Image Map Details**
How to Edit an Existing Image Map in DITA

To edit an existing image map, use any of the following methods:

- Simply double-click the image.
- Right-click the image and select Image Map Editor.
- Click the Image Map Editor button below the image.

All three methods open the Image Map Editor where you can make changes to the image map using the various features described above. You can also make changes to the XML structure of the image map in the Text editing mode.

You can also click the Image Map Details button below the image to expand a section that displays the details of the image map and allows you to change the shape and coordinates of the hyperlinked areas. Keep in mind that if you change the shape in this section, you also need to add or remove coordinates to match the requirements of the new shape.

Overlapping Areas

If shapes overlap one another in the Image Map Editor, the one on the top layer takes precedence. The number shown inside each shape represents its layer (if the numbers are not displayed, click the Show/Hide Numbers button on the Image Map Editor toolbar (on page 2981)). To change the layer order for a shape, use the layer buttons on the Image Map Editor toolbar (on page 2981) (↑, ↓, ⏹, ⏹).

If you insert a shape and all of its coordinates are completely inside another shape, the Image Map Editor will display a warning to let you know that the shape is entirely covered by a bigger shape. Keep in mind that if a shape is completely inside another shape, its hyperlink will only be accessible if its layer is on top of the bigger shape.

Related Information:
DITA 'imagemap' Element Specifications
Working with Images in DITA Topics (on page 2973)

Adding Tables in DITA Topics

You can use the Insert Table action on the toolbar or from the contextual menu to add a table in a DITA topic. By default, DITA supports four types of tables:

- DITA Simple table model (on page 2987) - This is the most commonly used model for basic tables.
- CALS table model (OASIS Exchange Table Model) (on page 2988) - This is used for more advanced functionality.
• **DITA Choice table model (on page 2991)** - This is used within a `<step>` element in a DITA Task document to describe a series of optional choices that a user must make before proceeding.

• **DITA Properties table model (on page 2993)** - This is used in DITA Reference documents to describe a property (for example, its type, value, and description).

If you are using a specialized DITA vocabulary, it may contain specialized versions of these table models.

Since DITA is a structured format, you can only insert a table in places in the structure of a topic where tables are allowed. The Oxygen XML Editor toolbar provides support for entering and editing tables. It also helps to indicate where you are allowed to insert a table or its components by disabling the appropriate buttons.

### Inserting a Simple Table Model

To insert a Simple DITA table, select the **Insert Table** action on the toolbar or from the contextual menu (or the **Table** submenu from the **DITA** menu). The **Insert Table** dialog box appears. Select **Simple** for the table Model.

![Figure 759. Insert Table Dialog Box - Simple Model](image)

The dialog box allows you to configure the following options when you select the **Simple** table model:

**Title**

If this checkbox is selected, you can specify a title for your table in the adjacent text box.

**Generate table header**

If selected, an extra row will be inserted at the top of the table to be used as the table header.

**Column widths**
Allows you to specify the type of properties for column widths (@colwidth attribute). You can choose one of the following properties for the column width:

- **proportional** - The width is specified in proportional (relative) units of measure. The proportion of the column is specified in a @relcolwidth attribute with the values listed as the number of shares followed by an asterisk. The value of the shares is totaled and rendered as a percent. For example, relcolwidth="1* 2* 3*" causes widths of 16.7%, 33.3%, and 66.7%. When entering content into a cell in one column, the width proportions of the other columns are maintained. If you change the width by dragging a column in Author mode, the values of the @relcolwidth attribute are automatically changed accordingly. By default, when you insert, drag and drop, or copy/paste a column, the value of the @relcolwidth attribute is 1*.

- **dynamic** - If you choose this option, the columns are created without a specified width. Entering content into a cell changes the rendered width dynamically. If you change the width by dragging a column in Author mode, a dialog box will be displayed that asks you if you want to switch to proportional or fixed column widths.

### Frame

Allows you to specify a value for the @frame attribute. It is used to specify where a border should appear in the table. The allowed values are as follows:

- **none** - No border will be added.
- **all** - A border will be added to all frames.
- **top** - A border will be added to the top frame.
- **topbot** - A border will be added to the top and bottom frames.
- **bottom** - A border will be added to the bottom frame.
- **sides** - A border will be added to the side frames.
- **-dita-use-conref-target** - Normally, when using a @conref, the values of attributes specified locally are preserved. You can choose this option to override this behavior and pull the value of this particular attribute from the @conref target. For more information, see [https://www.oxygenxml.com/dita/1.3/specs/langRef/attributes/ditauseconreftarget.html](https://www.oxygenxml.com/dita/1.3/specs/langRef/attributes/ditauseconreftarget.html).

**Note:**

The options in the Insert Table dialog box for DITA documents are persistent, so changes made in one session will carry over to another.

When you click **Insert**, a simple table is inserted into your document at the current cursor position.

## Inserting a CALS Table Model (OASIS Exchange Table)

To insert an OASIS Exchange Table (CALS), select the **Insert Table** action on the toolbar or from the contextual menu (or the Table submenu from the DITA menu). The Insert Table dialog box appears. Select **CALS** for the table **Model**. This model allows you to configure more properties than the **Simple** model.
The dialog box allows you to configure the following options when you select the CALS table model:

**Title**

If this checkbox is selected, you can specify a title for your table in the adjacent text box.

**Table Size**

Allows you to choose the number of Rows and Columns for the table.

**Generate table header**

If selected, an extra row will be inserted at the top of the table to be used as the table header.

**Column widths**

Allows you to specify the type of properties for column widths (@colwidth attribute). You can choose one of the following properties for the column width:

- **proportional** - The width is specified in proportional (relative) units of measure. The proportion of the column is specified in a @colwidth attribute with the values listed as the number of shares followed by an asterisk. The value of the shares is totaled and rendered as a percent. For example, \texttt{colwidth="1* 2* 3*"} causes widths of 16.7%, 33.3%, and 66.7%. When entering content into a cell in one column, the width proportions of the other columns are maintained. If you change the width by dragging a column in Author mode, the values of the @colwidth attribute are automatically changed accordingly. By default, when you insert, drag and drop, or copy/paste a column, the value of the @colwidth attribute is `1*`.

- **dynamic** - If you choose this option, the columns are created without a specified width (@colwidth attribute). Entering content into a cell changes the rendered width dynamically.
If you change the width by dragging a column in Author mode, a dialog box will be displayed that asks you if you want to switch to proportional or fixed column widths.

- **fixed** - The width is specified in fixed units. By default, the \texttt{pt} unit is inserted, but you can change the units in the \texttt{colspecs} (column specifications) section above the table or in Text mode. The following units are allowed: \texttt{pt} (points), \texttt{cm} (centimeters), \texttt{mm} (millimeters), \texttt{pi} (picas), \texttt{in} (inches).

**Frame**

Allows you to specify a value for the \texttt{@frame} attribute. It is used to specify where a border should appear in the table. The allowed values are as follows:

- **none** - No border will be added.
- **all** - A border will be added to all frames.
- **top** - A border will be added to the top frame.
- **topbot** - A border will be added to the top and bottom frames.
- **bottom** - A border will be added to the bottom frame.
- **sides** - A border will be added to the side frames.
- **-dita-use-conref-target** - Normally, when using a \texttt{@conref}, the values of attributes specified locally are preserved. You can choose this option to override this behavior and pull the value of this particular attribute from the \texttt{@conref} target. For more information, see https://www.oxygenxml.com/dita/1.3/specs/langRef/attributes/ditauseconreftarget.html.

**Row separator**

Specifies whether or not to include row separators (\texttt{@rowsep} attribute). The allowed values are: 0 (no separator) and 1 (include separators).

**Column separator**

Specifies whether or not to include column separators (\texttt{@colsep} attribute). The allowed values are: 0 (no separator) and 1 (include separators).

**Alignment**

Specifies the alignment of the text within the table (\texttt{@align} attribute). The allowed values are:

- **left** - Aligns the text to a left position.
- **right** - Aligns the text to a right position.
- **center** - Aligns the text to a centered position.
- **justify** - Stretches the line of text so that it has equal width.

**Note:**

The \texttt{justify} value cannot be rendered in Author mode, so you will only see it in the output.
• **char** - Aligns text to the leftmost occurrence of the value specified on the `@char` attribute for alignment.

• **-dita-use-conref-target** - Normally, when using a `@conref`, the values of attributes specified locally are preserved. You can choose this option to override this behavior and pull the value of this particular attribute from the `@conref` target. For more information, see [https://www.oxygenxml.com/dita/1.3/specs/langRef/attributes/ditauseconreftarget.html](https://www.oxygenxml.com/dita/1.3/specs/langRef/attributes/ditauseconreftarget.html).

**Note:**

The options in the **Insert Table** dialog box for DITA documents are persistent, so changes made in one session will carry over to another.

When you click **Insert**, a CALS table is inserted into your document at the current cursor position.

When you insert a CALS table, you see a link for setting the `colspecs` (column specifications) of your table. Click the link to open the controls that allow you to adjust various column properties. Although they appear as part of the **Author mode (on page 354)**, the `colspecs` link and its controls will not appear in your output. They are just there to make it easier to adjust how the columns of your table are formatted.

![Sample CALS Table with Fixed Width](image)

**Inserting a Choice Table Model**

To insert a **Choice** table within a `<step>` element in a DITA Task document, select the **Insert Table** action on the toolbar or in the **Insert** submenu from the contextual menu (or the **Table** submenu from the DITA menu), or select `choicetable` from the **Content Completion Assistant (on page 3217)**. The **Insert Table** dialog box appears. Select **Simple** for the table **Model**.
The dialog box allows you to configure the following options when you insert a *Choice* table model within a DITA Task:

**Table Size**

Allows you to choose the number of **Rows** and **Columns** for the table.

**Generate table header**

If selected, an extra row will be inserted at the top of the table to be used as the table header.

**Column widths**

Allows you to specify the type of properties for column widths (**@colwidth** attribute). You can choose one of the following properties for the column width:

- **proportional** - The width is specified in proportional (relative) units of measure. The proportion of the column is specified in a **@relcolwidth** attribute with the values listed as the number of shares followed by an asterisk. The value of the shares is totaled and rendered as a percent. For example, **relcolwidth="1* 2* 3*"** causes widths of 16.7%, 33.3%, and 66.7%. When entering content into a cell in one column, the width proportions of the other columns are maintained. If you change the width by dragging a column in **Author** mode, the values of the **@relcolwidth** attribute are automatically changed accordingly. By default, when you insert, drag and drop, or copy/paste a column, the value of the **@relcolwidth** attribute is **1***.

- **dynamic** - If you choose this option, the columns are created without a specified width. Entering content into a cell changes the rendered width dynamically. If you change the width by dragging a column in **Author** mode, a dialog box will be displayed that asks you if you want to switch to proportional or fixed column widths.

**Frame**

Allows you to specify a value for the **@frame** attribute. It is used to specify where a border should appear in the table. The allowed values are as follows:

- **none** - No border will be added.
- **all** - A border will be added to all frames.
• **top** - A border will be added to the top frame.
• **topbot** - A border will be added to the top and bottom frames.
• **bottom** - A border will be added to the bottom frame.
• **sides** - A border will be added to the side frames.
• **-dita-use-conref-target** - Normally, when using a `@conref`, the values of attributes specified locally are preserved. You can choose this option to override this behavior and pull the value of this particular attribute from the `@conref` target. For more information, see https://www.oxygenxml.com/dita/1.3/specs/langRef/attributes/ditauseconreftarget.html.

When you click **Insert**, a *Choice* table is inserted into your DITA Task document at the current cursor position (within a `<step>` element).

**Inserting a Properties Table Model**

To insert a *Properties* table within a `<refbody>` element in a DITA Reference document, select the **Insert Table** action on the toolbar or in the **Insert** submenu from the contextual menu (or the **Table** submenu from the DITA menu), or select **properties(wizard)** from the **Content Completion Assistant** (on page 3217). The **Insert Table** dialog box appears. Select **Properties** for the table **Model**.

![Insert Table Dialog Box - Properties Model](image)

The dialog box allows you to configure the following options when you insert a *Properties* table model within a DITA Reference:

**Table Size**

Allows you to choose the number of **Rows** and **Columns** for the table.

**Generate table header**
If selected, an extra row will be inserted at the top of the table to be used as the table header.

**Frame**

Allows you to specify a value for the `@frame` attribute. It is used to specify where a border should appear in the table. The allowed values are as follows:

- **none** - No border will be added.
- **all** - A border will be added to all frames.
- **top** - A border will be added to the top frame.
- **topbot** - A border will be added to the top and bottom frames.
- **bottom** - A border will be added to the bottom frame.
- **sides** - A border will be added to the side frames.
- `-dita-use-conref-target` - Normally, when using a `@conref`, the values of attributes specified locally are preserved. You can choose this option to override this behavior and pull the value of this particular attribute from the `@conref` target. For more information, see https://www.oxygenxml.com/dita/1.3/specs/langRef/attributes/ditauseconreftarget.html.

When you click **Insert**, a **Properties** table is inserted into your DITA Reference document at the current cursor position (within a `<refbody>` element).

**Editing an Existing Table**

You can edit the structure of an existing table using the table buttons on the toolbar (or in the contextual menu) to add or remove cells, rows, or columns, and to set basic table properties. Additional attributes can be used to fine-tune the formatting of your tables by using the **Attributes** view ([on page 627](on page 627)) ([Window > Show View > Attributes](on page 627)). See the **DITA documentation** for a full explanation of these attributes.

You can also use the ![Table Properties (Ctrl + T (Command + T on macOS))](on page 2997) action from the toolbar or contextual menu (or **DITA** menu) to **modify many of the properties of the table** ([on page 2997](on page 2997)).

Also, remember that underneath the visual representation, both table models are really just XML. If necessary, you can edit the XML directly by switching to **Text mode** ([on page 353](on page 353)).

You can use normal copy/paste shortcuts to move content between cells. Oxygen XML Editor includes a **Smart Paste feature ([on page 613](on page 613))** that preserves certain style and structure information when pasting content.

**Tip:**

When copying a multiple selection of table cells and pasting them outside the table, a new table will be created. When pasting into space-preserved elements, the cell content will be pasted as plain text.

**Related Information:**

**Editing Tables in Author Mode** ([on page 683](on page 683))
DITA Table Layouts

Depending on the context, DITA accepts the following table layouts:

- **CALS table model (on page 2995)**
- **Simple table model (on page 2995)**
- **Choice table model (on page 2996)**
- **Properties table model (on page 2996)**

**CALS Table Model Layout**

The **CALS** table model allows for more flexibility and table customization than other models. When choosing a **CALS** table model from the **Insert Table** dialog box, you have access to more configurable properties. The layout of a **CALS** table includes a **colspecs** section that allows you to easily configure some properties without opening the **Table Properties** dialog box. For example, you can change the value of column widths (**@colwidth** attribute) or the text alignment (**@align** attribute). Although they appear as part of the **Author mode (on page 354)**, the **colspecs** link and its controls will not appear in your output. They are just there to make it easier to adjust how the columns of your table are formatted.

![CALS Table in DITA](image)

**Simple Table Model Layout**

When choosing a **Simple** table model from the **Insert Table** dialog box, you only have access to configure a few properties. For example, you can choose the number of rows and columns, specify values for frames, and choose from a few types of properties for the column width. The layout of this type of table is very simple, as the name suggests.

![DITA Simple Table](image)
**Choice Table Model Layout**

A **Choice** table model is used within a `<step>` element in a DITA Task document to describe a series of optional choices that a user must make before proceeding. The `<choicetables>` element is a useful device for documenting options within a single step of a task. You can insert Choice tables in DITA Task documents either by selecting `choicetable` from the *Content Completion Assistant* (on page 3217) (within a `<step>` element) or by using the **Insert Table** action on the toolbar or from the contextual menu). The options and layout of a Choice table is similar to the Simple table model.

![Figure 766. DITA Choice Table](image)

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Opt A</td>
<td></td>
</tr>
<tr>
<td>Opt B</td>
<td></td>
</tr>
<tr>
<td>Opt C</td>
<td></td>
</tr>
</tbody>
</table>

**Properties Table Model Layout**

A **Properties** table model is used within a `<refbody>` element in a DITA Reference document to describe a property (for example, its type, value, and description). You can insert Properties tables in DITA Reference documents either by selecting `properties(wizard)` from the *Content Completion Assistant* (on page 3217) (within a `<refbody>` element) or by using the **Insert Table** action on the toolbar (or from the contextual menu) and selecting Properties for the Model. The layout of a Properties table is very simple. It allows for a maximum of 3 columns (typically for property type, value, and description) and the only options available are for whether or not you want a header row and for specifying frames (borders).

![Figure 767. DITA Properties Table](image)

<table>
<thead>
<tr>
<th>Property Type</th>
<th>Property Value</th>
<th>Property Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>B</td>
<td>C</td>
</tr>
</tbody>
</table>

**Table Validation in DITA**

Oxygen XML Editor reports table layout problems that are detected in manual or automatic validations. When you validate a DITA map (on page 3218) with the **Validate and Check for Completeness** action, if the **Report table layout problems** option (on page 2946) is selected in the DITA Map Completeness Check dialog box, table layout problems will be reported in the validation results. The types of errors that may be reported for DITA table layout problems include:

- **CALS Tables**
  - A row has fewer cells than the number of columns detected from the table `@cols` attribute.
  - A row has more cells than the number of columns detected from the table `@cols` attribute.
  - A cell has a vertical span greater than the available rows count.
• The number of `<colspecs>` is different than the number of columns detected from the table `@cols` attribute.
• The number of columns detected from the table `@cols` attribute is different than the number of columns detected in the table structure.
• The value of the `@cols`, `@rowsep`, or `@colsep` attributes are not numeric.
• The `@namest`, `@nameend`, or `@colname` attributes point to an incorrect column name.

Simple or Choice Tables

A row has fewer cells than the number of table columns.

Editing Table Properties in DITA

To customize the look of a table in DITA, place the cursor anywhere in a table and invoke the Table Properties (Ctrl + T (Command + T on macOS)) action from the toolbar or the Table submenu of the contextual menu (or DITA menu). This opens the Table properties dialog box.

The Table properties dialog box allows you to set specific properties to the table elements. The options that are available depend on the context and location within the table where the action was invoked.

Note:
Some properties allow the following special values, depending on the context and the current properties or values:

• `<not set>` - Use this value if you want to remove a property.
• `<preserve>` - If you select multiple elements that have the same property set to different values, you can choose this value to keep the values that are already set. In some cases it can also be used to keep the current non-standard value for a particular property.

Edit Table Properties for a CALS Table Model

For a CALS table model, the Table properties dialog box includes four tabs of options:

• Table tab - The options in this tab apply to the entire table.
• Row tab - The options in this tab apply to the current row or selection of multiple rows. A message at the bottom of the tab tells you how many rows will be affected.
• Column tab - The options in this tab apply to the current column or selection of multiple columns. A message at the bottom of the tab tells you how many columns will be affected.
• Cell tab - The options in this tab apply to the current cell or selection of multiple cells. A message at the bottom of the tab tells you how many cells will be affected.

The options in four tabs include a Preview pane that shows a representation of the modification.
The options in the four tabs include the following:

**Horizontal alignment (Available in the Table, Column, and Cell tabs)**

Specifies the horizontal alignment of text within the current table/column/cell or selection of multiple columns/cells (@align attribute). The allowed values are as follows:

- **left** - Aligns the text to a left position.
- **right** - Aligns the text to a right position.
- **center** - Aligns the text to a centered position.
- **justify** - Stretches the line of text so that it has equal width.

**Note:**

The justify value cannot be rendered in **Author** mode, so you will only see it in the output.

- **char** - Aligns text to the leftmost occurrence of the value specified on the @char attribute for alignment.
- **dita-use-conref-target** - Normally, when using a @conref, the values of attributes specified locally are preserved. You can choose this option to override this behavior and pull the value of this particular attribute from the @conref target. For more information, see [https://www.oxygenxml.com/dita/1.3/specs/langRef/attributes/ditauseconreftarget.html](https://www.oxygenxml.com/dita/1.3/specs/langRef/attributes/ditauseconreftarget.html).

**Vertical alignment (Available in the Row and Cell tabs)**
Specifies the vertical alignment of text within the current row/cell or selection of multiple rows/cells (@valign attribute). The allowed values are as follows:

- **top** - Aligns the text at the top of the cell.
- **middle** - Aligns the text in a vertically centered position.
- **bottom** - Aligns the text at the bottom of the cell.
- **-dita-use-conref-target** - Normally, when using a @conref, the values of attributes specified locally are preserved. You can choose this option to override this behavior and pull the value of this particular attribute from the @conref target. For more information, see https://www.oxygenxml.com/dita/1.3/specs/langRef/attributes/ditauseconreftarget.html.

**Column separator (Available in the Table, Column, and Cell tabs)**

Specifies whether or not to include column separators (borders/grid lines) in the form of the @colsep attribute. The allowed values are: 0 (no separator) and 1 (include separators).

**Row separator (Available in all four tabs)**

Specifies whether or not to include row separators (borders/grid lines) in the form of the @rowsep attribute. The allowed values are: 0 (no separator) and 1 (include separators).

**Frame (Available only in the Table tab)**

Allows you to specify a value for the @frame attribute. It is used to specify where a border should appear in the table. The allowed values are as follows:

- **none** - No border will be added.
- **all** - A border will be added to all frames.
- **top** - A border will be added to the top frame.
- **topbot** - A border will be added to the top and bottom frames.
- **bottom** - A border will be added to the bottom frame.
- **sides** - A border will be added to the side frames.
- **-dita-use-conref-target** - Normally, when using a @conref, the values of attributes specified locally are preserved. You can choose this option to override this behavior and pull the value of this particular attribute from the @conref target. For more information, see https://www.oxygenxml.com/dita/1.3/specs/langRef/attributes/ditauseconreftarget.html.

**Edit Table Properties for a Simple, Choice, or Properties Table Model**

For a Simple, Choice, Properties table model, the Table properties dialog box only allows you to edit a few options.

**Table tab**

**Frame**

Allows you to specify a value for the @frame attribute. It is used to specify where a border should appear in the table. The allowed values are as follows:
- **none** - No border will be added.
- **all** - A border will be added to all frames.
- **top** - A border will be added to the top frame.
- **topbot** - A border will be added to the top and bottom frames.
- **bottom** - A border will be added to the bottom frame.
- **sides** - A border will be added to the side frames.
- **-dita-use-conref-target** - Normally, when using a `@conref`, the values of attributes specified locally are preserved. You can choose this option to override this behavior and pull the value of this particular attribute from the `@conref` target. For more information, see [https://www.oxygenxml.com/dita/1.3/specs/langRef/attributes/ditauseconreftarget.html](https://www.oxygenxml.com/dita/1.3/specs/langRef/attributes/ditauseconreftarget.html).

**Row tab (not available for Properties tables)**

**Row type**

Allows you to change the row to a body or header type of row.

---

**Related Information:**

* Adding Tables in DITA Topics *(on page 2986)*
* Editing Tables in Author Mode *(on page 683)*

---

**Adding MathML Equations in DITA Topics**

You can add MathML equations in a DITA document using one of the following methods:

- **Embed MathML directly into a DITA topic.** You can use **Insert > Insert Equation** from the contextual menu to insert a MathML equation. Clicking on the equation will open a **MathML Editor** where you can edit the code.
- **Reference an external MathML file as an image,** using the **Insert Image** action that is available on the DITA toolbar (or from the **DITA > Insert** menu).

---

**Publishing Notes:**

- MathML equations contained in DITA topics can be published out-of-the-box in PDF using the **DITA PDF** transformation scenario.
- The **DITA Map PDF - based on HTML5 & CSS** transformation scenario *(on page 1768)* support MathML equations *(on page 1947)*.
- For details about HTML output, see **How to View MathML Equations in HTML Output** *(on page 1709)*.
- For other publishing formats, you might need to employ additional customizations for handling MathML content.
Adding LaTeX Equations in DITA Topics

LaTeX is a high-quality typesetting system that includes features designed for the production of technical and scientific documentation. LaTeX can also be used to express mathematical formulas in a textual format. By default, web browsers and PDF readers do not have support to show mathematical equations written in LaTeX, but there are open-source projects that can read LaTeX and convert it to other image types.

Adding support for writing LaTeX equations in a DITA topic implies three stages:

1. Find a way to write the equation in the DITA XML content. You can either create a DITA DTD specialization and add a new element (for example, called `<latex>` and it extends the DITA `<foreign>` element). Alternatively, you can directly use the DITA `<foreign>` element with a specific `@outputclass` attribute value:

```xml
<!DOCTYPE topic PUBLIC "-//OASIS//DTD DITA Topic//EN" "topic.dtd">
<topic id="testEquation">
  <title>Test equation</title>
  <body>
    <p><foreign outputclass="embed-latex">L' = \(L\sqrt{1-\frac{v^2}{c^2}}\)</foreign></p>
  </body>
</topic>
```

2. If you want Oxygen XML Editor to properly present the LaTeX equation when editing in the Author visual mode, you need a plugin that converts the equation content to an image. There is a sample plugin that does that here: https://github.com/oxygenxml/wsaccess-javascript-sample-plugins/tree/master/latex-images-support. You can download and copy the plugin folder `latex-images-support` to the Oxygen XML Editor plugins folder, then restart Oxygen XML Editor.

3. The final stage would be to publish the content to HTML-based or PDF output. The following DITA Open Toolkit plugin automatically converts LaTeX images to SVG when publishing: https://github.com/oxygenxml/dita-latex.

DITA Topic Author Mode Actions

A variety of actions are available for DITA documents that can be found in DITA menu, toolbar, contextual menu, and the Content Completion Assistant (on page 3217).

DITA Toolbar Actions

The following default actions are available on the DITA toolbar when editing in Author mode (by default, most of them are also available in the DITA menu and in various submenus of the contextual menu):

- **Bold**
  Surrounds the selected text with a `<b>` tag. You can use this action on multiple non-contiguous selections.

- **Italic**
Surrounds the selected text with an `<i>` tag. You can use this action on multiple non-contiguous selections.

<i> Underline

Surrounds the selected text with a `<u>` tag. You can use this action on multiple non-contiguous selections.

🔗 **Link Actions Drop-Down Menu**

The following link actions are available from this menu:

- **Cross Reference**
  
  Opens the Cross Reference (xref) dialog box (on page 3073) that allows you to insert a link to a target DITA resource at the current location within a document. The target resource can be the location of a file or a key that is already defined in your DITA map (on page 3218) structure. Once the target resource has been selected, you can also target specific elements within that resource. For more information, see Linking in DITA Topics (on page 3072).

- **File Reference**
  
  Opens the File Reference dialog box (on page 3073) that allows you to insert a link to a target non-DITA file resource at the current location within a document. The target resource can be the location of a file or a key that is already defined in your DITA map structure. For more information, see Linking in DITA Topics (on page 3072).

- **Web Link**
  
  Opens the Web Link dialog box (on page 3073) that allows you to insert a link to a target web-related resource at the current location within a document. The target resource can be a URL or a key that is already defined in your DITA map structure. For more information, see Linking in DITA Topics (on page 3072).

- **Related Link to Topic**
  
  Opens the Cross Reference (xref) dialog box (on page 3074) that allows you to insert a link to a target DITA resource in a related links section at the bottom of the current document. The target resource can be the location of a file or a key that is already defined in your DITA map structure. Once the target resource has been selected, you can also target specific elements within that resource. If a related links section does not already exist, this action creates one. For more information, see Linking in DITA Topics (on page 3072).

**Tip:**

You can use the Find Similar Topics action (available in the contextual menu or DITA menu) to quickly find related topics that can be added as related links. It opens the Open/Find Resource view and performs a search.
using text content from the `<title>`, `<shortdesc>`, `<keyword>`, and `<indexterm>` elements.

**Related Link to File**

Opens the **File Reference** dialog box *(on page 3074)* that allows you to insert a link to a target non-DITA file resource in a related links section at the bottom of the current document. The target resource can be the location of a file or a key that is already defined in your **DITA map** structure. If a related links section does not already exist, this action creates one. For more information, see **Linking in DITA Topics** *(on page 3072)*.

**Related Link to Web Page**

Opens the **Web Link** dialog box *(on page 3074)* that allows you to insert a link to a target web-related resource in a related links section at the bottom of the current document. The target resource can be a URL or a key that is already defined in your **DITA map** structure. If a related links section does not already exist, this action creates one. For more information, see **Linking in DITA Topics** *(on page 3072)*.

**Insert Image**

Opens the **Insert Image** dialog box *(on page 2973)* that allows you to configure the properties of an image to be inserted into a DITA document at the cursor position.

**Insert Media Resource**

Opens the **Insert Media** dialog box *(on page 2977)* that allows you to select and configure the properties of a media object to be inserted into a DITA document at the cursor position. The result will be that a reference to the specified video, audio, or embedded HTML frame is inserted in an `<object>` element and it is rendered in **Author** mode so that it can be played directly from there.

**Insert Section Drop-Down Menu**

The following insert actions are available from this menu:

- **Insert Section**

  Inserts a new `<section>` element in the document, depending on the current context.

- **Insert Concept**

  Inserts a new `<concept>` element, depending on the current context. Concepts provide background information that users must know before they can successfully work with a product or interface.

- **Insert Task**

  Inserts a new `<task>` element, depending on the current context. Tasks are the main building blocks for task-oriented user assistance. They generally provide step-by-step instructions that will help a user to perform a task.
**Insert Topic**

Inserts a new `<topic>` element, depending on the current context. Topics are the basic units of DITA content and are usually organized around a single subject.

**Insert Reference**

Inserts a new `<reference>` element, depending on the current context. A reference is a top-level container for a reference topic.

**Insert Paragraph**

Inserts a new paragraph at current cursor position.

**Reuse Content**

This action provides a mechanism for reusing content fragments. It opens the Reuse Content dialog box (on page 3043) that allows you to insert several types of references to reusable content at the cursor position. The types of references that you can insert using this dialog box include content references (`@conref`) (on page 3044), content key references (`@conkeyref`) (on page 3046), or key references to metadata (`@keyref`) (on page 3049).

**Insert step or list item**

Inserts a new list or step item in the current list type.

**Insert Unordered List**

Inserts an unordered list at the cursor position. A child list item is also automatically inserted by default. You can also use this action to convert selected paragraphs or other types of lists to an unordered list.

**Insert Ordered List**

Inserts an ordered list at the cursor position. A child list item is also automatically inserted by default. You can also use this action to convert selected paragraphs or other types of lists to an ordered list.

**Sort**

Sorts cells or list items in a table.

**Insert Table**

Opens a dialog box that allows you to configure and insert a table. You can generate a header and footer, set the number of rows and columns of the table and decide how the table is framed. You can also use this action to convert selected paragraphs, lists, and inline content (mixed content, text plus markup, that is rendered inside a block element (on page 3216)) into a table, with the selected content inserted in the first column, starting from the first row after the header (if a header is inserted).
Note:
If the selection contains a mixture of elements that cannot be converted, you will receive an error message saying that **Only lists, paragraphs, or inline content can be converted to tables**.

- **Insert Row**
  Inserts a new table row with empty cells below the current row. This action is available when the cursor is positioned inside a table.

- **Delete Row(s)**
  Deletes the table row located at the cursor position or multiple rows in a selection.

- **Insert Column**
  Inserts a new table column with empty cells after the current column. This action is available when the cursor is positioned inside a table.

- **Delete Column(s)**
  Deletes the table column located at the cursor position or multiple columns in a selection.

- **Table Properties**
  Opens the Table properties dialog box that allows you to configure properties of a table (such as frame borders).

- **Join Cells**
  Joins the content of the selected cells (both horizontally and vertically).

- **Split Cell**
  Splits the cell at the cursor location. If Oxygen XML Editor detects more than one option to split the cell, a dialog box will be displayed that allows you to select the number of rows or columns to split the cell into.

**DITA Contextual Menu Actions**

The following actions are available in the contextual menu when editing in Author mode (most of them are also available in the DITA menu at the top of the interface):

- **Add File to Review Task**
  This action can be used to add the current document to a task in the Content Fusion Tasks Manager view. Oxygen Content Fusion is a flexible, intuitive collaboration platform designed to adapt to any type of documentation review workflow. This functionality is available through a pre-installed connector add-on (on page 2527). To fully take advantage of all of the benefits and features of Content Fusion, your organization will need an Oxygen Content Fusion Enterprise Server. For more information, see the Oxygen Content Fusion website.
**Edit Attributes**

Displays an in-place attributes editor (on page 630) that allows you to manage the attributes of an element.

**Edit Profiling Attributes**

Allows you to change the profiling attributes (on page 668) defined on all selected elements.

**Cut (Ctrl + X (Command + X on macOS))**

Removes the currently selected content from the document and places it in the clipboard.

**Copy (Ctrl + C (Command + C on macOS))**

Places a copy of the currently selected content in the clipboard.

**Paste (Ctrl + V (Command + V on macOS))**

Inserts the current clipboard content into the document at the cursor position.

**Paste special submenu**

This submenu includes the following special paste actions that are specific to the DITA framework:

**Paste as content reference**

 Inserts a content reference (a DITA element with a `@conref` attribute) to the DITA XML element from the clipboard. An entire DITA XML element with an ID attribute must be present in the clipboard when the action is invoked. The `conref` attribute will point to this ID value.

**Paste as content key reference**

Allows you to indirectly reference content using the `@conkeyref` attribute. When the DITA content is processed, the key references are resolved using key definitions from DITA maps (on page 3218). To use this action, you must first do the following:

1. Make sure the DITA element that contains the copied content has an ID attribute assigned to it.
2. In the DITA Maps Manager view, make sure that the Context combo box (on page 2901) points to the correct map that stores the keys.
3. Make sure the topic that contains the content you want to reference has a key assigned to it. To assign a key, right-click the topic with its parent map opened in the DITA Maps Manager, select Edit Properties, and enter a value in the Keys field.

**Paste as link**

Looks for the first element with an ID value in the clipboard and inserts an `<xref>` that points to that element. If no elements with an ID value are found, a message...
will appear that informs you that to use this action, the clipboard contents must include at least one element with a declared ID.

**Paste as link (keyref)**

Inserts a link to the element that you want to reference. To use this action, you must first do the following:

1. Make sure the DITA element that contains the copied content has an ID attribute assigned to it.
2. In the **DITA Maps Manager** view, make sure that the **Context combo box** *(on page 2901)* points to the correct map that stores the keys.
3. Make sure the topic that contains the content you want to reference has a key assigned to it. To assign a key, right-click the topic with its parent map opened in the **DITA Maps Manager**, select **Edit Properties**, and enter a value in the **Keys** field.

**Insert submenu**

This submenu includes the following insert actions that are specific to the DITA framework:

**Insert Table**

Opens a dialog box that allows you to configure and insert a table. You can generate a header and footer, set the number of rows and columns of the table and decide how the table is framed. You can also use this action to convert selected paragraphs, lists, and inline content (mixed content, text plus markup, that is rendered inside a **block element** *(on page 3216)*) into a table, with the selected content inserted in the first column, starting from the first row after the header (if a header is inserted).

**Note:**

If the selection contains a mixture of elements that cannot be converted, you will receive an error message saying that **Only lists, paragraphs, or inline content can be converted to tables**.

**Insert Image**

Inserts an image reference *(on page 719)* at the cursor position. Depending on the current location, an image-type element is inserted.

**Insert Media Resource**

Opens a **Choose Media** dialog box *(on page 747)* that allows you to select the URL of a media object to be inserted into a document at the cursor position. The result will be that a reference to the specified video, audio, or embedded HTML frame is inserted and rendered in **Author** mode so that it can be played directly from there.
∑ Insert Equation

Opens the XML Fragment Editor that allows you to insert and edit MathML notations (on page 749).

⪪ Insert Note

Inserts a new `<note>` element at the current cursor position.

{ } Insert Code Block

Inserts a new `<codeblock>` element at current cursor position.

✎ Insert Menu Cascade

Inserts a new `<menucascade>` element at current cursor position.

¶ Insert Paragraph

Inserts a new `<p>` (paragraph) element at current cursor position.

§ Insert Section

Inserts a new `<section>` element in the document, depending on the current context.

❉ Insert Topic

Inserts a new `<topic>` element, depending on the current context. Topics are the basic units of DITA content and are usually organized around a single subject.

Insert Entity

Allows you to insert a predefined entity or character entity. Surrogate character entities (range #x10000 to #x10FFFF) are also accepted. Character entities can be entered in one of the following forms:

• `<decimal value>` - e.g. #65
• `&<decimal value>` - e.g. &#65
• `<hexadecimal value>` - e.g. #x41
• `&<hexadecimal value>` - e.g. &x41

Style submenu

This submenu includes the following text styling actions:

B Bold

Emphasizes the selected text by surrounding it with a `<b>` (bold) tag. You can use this action on multiple non-contiguous selections.

I Italic

Emphasizes the selected text by surrounding it with an `<i>` (italic) tag. You can use this action on multiple non-contiguous selections.

U Underline
Emphasizes the selected text by surrounding it with a `<u>` (underline) tag. You can use this action on multiple non-contiguous selections.

**Subscript**

Surrounds the selected text with a `<sub>` (subscript) tag, used for inserting a character (number, letter, or symbol) that will appear slightly below the baseline and slightly smaller than the rest of the text.

**Superscript**

Surrounds the selected text with a `<sup>` (superscript) tag, used for inserting a character (number, letter, or symbol) that will appear slightly above the baseline and slightly smaller than the rest of the text.

**Code**

Surrounds the selected text with a `<codeph>` tag.

**UI Control**

Surrounds the selected text with a `<uicontrol>` tag, used to mark up names of buttons, entry fields, menu items, or other interface objects.

**Filepath**

Surrounds the selected text with a `<filepath>` tag, used to indicate the name, and optionally the location of a referenced file. You can specify the directory that contains the file and other directories that may precede it in the system hierarchy.

**Image Map Editor**

This action is available in the contextual menu when it is invoked on an image. This action applies an *image map* to the current image (if one does not already exist) and opens the *Image Map Editor* dialog box. This feature allows you to create hyperlinks in specific areas of an image that will link to various destinations.

**Table Actions**

A variety of table editing actions are available in the contextual menu when it is invoked on a table (depending on the context, the table-related actions are promoted to the top level of the contextual menu and the Other Actions submenu provides access to the other actions):

**Insert Rows**

Opens a dialog box that allows you to insert any number of rows and specify the position where they will be inserted (Above or Below the current row).

**Delete Row(s)**

Deletes the table row located at the cursor position or multiple rows in a selection.

**Insert Columns**
Opens a dialog box that allows you to insert any number of columns and specify the position where they will be inserted (Above or Below the current column).

**Delete Column(s)**

Deletes the table column located at the cursor position or multiple columns in a selection.

**Join Cells**

Joins the content of the selected cells (both horizontally and vertically).

**Split Cell**

Splits the cell at the cursor location. If Oxygen XML Editor detects more than one option to split the cell, a dialog box will be displayed that allows you to select the number of rows or columns to split the cell into.

**Sort**

Sorts cells or list items in a table.

**Table Properties**

Opens the Table properties dialog box that allows you to configure properties of a table (such as frame borders).

**Other Actions submenu**

This submenu gives you access to all the usual contextual menu actions.

**Link submenu**

The following link actions are available from this submenu:

**Cross Reference**

Opens the Cross Reference (xref) dialog box (on page 3073) that allows you to insert a link to a target DITA resource at the current location within a document. The target resource can be the location of a file or a key that is already defined in your DITA map (on page 3218) structure. Once the target resource has been selected, you can also target specific elements within that resource. For more information, see Linking in DITA Topics (on page 3072).

**File Reference**

Opens the File Reference dialog box (on page 3073) that allows you to insert a link to a target non-DITA file resource at the current location within a document. The target resource can be the location of a file or a key that is already defined in your DITA map structure. For more information, see Linking in DITA Topics (on page 3072).

**Web Link**
Opens the **Web Link** dialog box *(on page 3073)* that allows you to insert a link to a target web-related resource at the current location within a document. The target resource can be a URL or a key that is already defined in your *DITA map* structure. For more information, see Linking in DITA Topics *(on page 3072)*.

**Related Link to Topic**

Opens the **Cross Reference (xref)** dialog box *(on page 3074)* that allows you to insert a link to a target DITA resource in a related links section at the bottom of the current document. The target resource can be the location of a file or a key that is already defined in your *DITA map* structure. Once the target resource has been selected, you can also target specific elements within that resource. If a related links section does not already exist, this action creates one. For more information, see Linking in DITA Topics *(on page 3072)*.

**Tip:**

You can use the **Find Similar Topics** action (available in the contextual menu or *DITA* menu) to quickly find related topics that can be added as related links. It opens the **Open/Find Resource** view and performs a search using text content from the *<title>, <shortdesc>, <keyword>, and <indexterm>* elements.

**Related Link to File**

Opens the **File Reference** dialog box *(on page 3074)* that allows you to insert a link to a target non-DITA file resource in a related links section at the bottom of the current document. The target resource can be the location of a file or a key that is already defined in your *DITA map* structure. If a related links section does not already exist, this action creates one. For more information, see Linking in DITA Topics *(on page 3072)*.

**Related Link to Web Page**

Opens the **Web Link** dialog box *(on page 3074)* that allows you to insert a link to a target web-related resource in a related links section at the bottom of the current document. The target resource can be a URL or a key that is already defined in your *DITA map* structure. If a related links section does not already exist, this action creates one. For more information, see Linking in DITA Topics *(on page 3072)*.

**Sort**

Available when invoked on a list, it opens a dialog box where you can configure a sorting operation for an entire list or a selection of list items.

**Generate IDs**

Oxygen XML Editor generates unique IDs for the current element (or elements), depending on how the action is invoked:
• When invoked on a single selection, an ID is generated for the selected element at the cursor position.
• When invoked on a block of selected content, IDs are generated for all top-level elements and elements listed in the **ID Options** dialog box that are found in the current selection.

**Note:**
The **Generate IDs** action does not overwrite existing ID values. It only affects elements that do not already have an `@id` attribute.

### Reuse submenu

This submenu includes the following actions regarding reusing content in DITA:

- **Reuse Content**
  
  This action provides a mechanism for reusing content fragments. It opens the **Reuse Content** dialog box *(on page 3043)* that allows you to insert several types of references to reusable content at the cursor position. The types of references that you can insert using this dialog box include content references *(@conref) *(on page 3044)*, content key references *(@conkeyref) *(on page 3046)*, or key references to metadata *(@keyref) *(on page 3049)*.

- **Push Current Element**
  
  Opens the **Push current element** dialog box *(on page 3052)* that allows content from a source topic to be inserted into another topic without any special coding in the topic where the content will be re-used.

- **Edit Content Reference**
  
  This action is available for elements with a `@conref` or `@conkeyref` attribute. It opens the **Edit Content Reference** dialog box that allows you to edit the source location (or key) and source element of a content reference (or content key reference), and the reference details *(@conref/@conkeyref and @conrefend attributes). For more information, see Reuse Content Dialog Box *(on page 3043).*

- **Replace Reference with Content**
  
  Replaces the referenced fragment *(@conref or @conkeyref)* at the cursor position with its content from its source. This action is useful if you want to make changes to the content in the currently edited document without changing the referenced fragment in its source location. If the source content includes references to other topics/resources *(hrefs)*, the operation also resolves those references relative to the new location. Attributes are preserved according to the following priority:
1. Attributes from the elements in the current document that reference other content are preserved except for attributes with a `-dita-use-conref-target` value.

2. Attributes from the referenced content are brought into the replaced elements in the current document except for `@id` attributes.

**Replace All References with Content**

Replaces all referenced fragments (`@keyref`, `@conref`, or `@conkeyref`) in the current document with the content. Attributes are preserved according to the following priority:

1. Attributes from the elements in the current document that reference other content are preserved except for attributes with a `-dita-use-conref-target` value.

2. Attributes from the referenced content are brought into the replaced elements in the current document except for `@id` attributes.

For keyrefs inside `<xref>` or `<link>` elements, the `@keyref` attribute is changed to an `@href` attribute, while the rest of the content for the keyref is replaced with its source content.

If the source content includes references to other topics/resources (`hrels`), the operation also resolves those references relative to the new location.

**Remove Content Reference**

Removes the content reference (`@conref` or `@conkeyref`) inside the element at the cursor position.

**Create Reusable Component**

Opens a dialog box that helps you to create a reusable component from the current element or selection of elements. If the Replace selection with content reference option is selected in the dialog box, the selection will be replaced with a content reference (`@conref`). If multiple elements are selected (for example, multiple steps or list items), the selection will be replaced with a content reference range (`@conref` and `@conrefend`). For more information, see Creating a Reusable Content Component (on page 3055).

**Insert Reusable Component**

Inserts a reusable component at cursor location. For more information, see Inserting a Reusable Content Component (on page 3056).

**Search References (Ctrl + Shift + G (Command + Shift + G on macOS))**

Finds the references to the `@id` attribute value for the element at the current cursor position, in all the topics contained in the current DITA map (on page 3218) (opened in the DITA Maps...
Manager view (on page 2897)). If no references are found for the current element, a dialog box will be displayed that offers you the option of searching for references to its ancestor elements.

Figure 769. Search References to Ancestors Dialog Box

Tip:
If you are invoking the action on an image, see Searching for References to Images (on page 2976) for details about what will be reported.

Find Similar Topics
Opens the Open/Find Resource view and performs a search using text content from the <title>, <shortdesc>, <keywords>, and <indexterm> elements. It is helpful for quickly finding related topics that can be added as related links.

Show Key Definition
Available for elements that have a @conkeyref or @keyref attribute set (or elements with an ancestor element that has a @conkeyref or @keyref attribute). It computes the key name and opens the DITA map (on page 3218) that contains the definition of the key with the element that defines that key selected.

About Element submenu
This submenu includes the following actions:

Style Guide
Opens the DITA Style Guide Best Practices for Authors in your browser and displays a topic that is relevant to the element at the cursor position. When editing DITA documents, this action is available in the contextual menu of the editing area (under the About Element sub-menu), in the DITA menu, and in some of the documentation tips that are displayed by the Content Completion Assistant (on page 3217).

Browse reference manual
Opens a reference to the documentation of the XML element closest to the cursor position in a web browser.

Go to Definition
Moves the cursor to the definition of the current element.
Select submenu

This submenu allows you to select the following:

**Element**
Selects the entire element at the current cursor position.

**Content**
Selects the entire content of the element at the current cursor position, excluding the start and end tag. Performing this action repeatedly will result in the selection of the content of the ancestor of the currently selected element content.

**Parent**
Selects the entire parent element at the current cursor position.

Text submenu

This submenu contains the following actions:

**To Lower Case**
Converts the selected content to lower case characters.

**To Upper Case**
Converts the selected content to upper case characters.

**Capitalize Sentences**
Converts to upper case the first character of every selected sentence.

**Capitalize Words**
Converts to upper case the first character of every selected word.

**Count Words**
Counts the number of words and characters (no spaces) in the entire document or in the selection for regular content and read-only content.

**Note:**
The content marked as deleted with [change tracking](on page 3224) is ignored when counting words.

**Convert Hexadecimal Sequence to Character (Ctrl + Shift + X (Command + Shift + X on macOS))**

Converts a sequence of hexadecimal characters to the corresponding Unicode character [on page 463]. The action can be invoked if there is a selection containing a valid hexadecimal sequence or if the cursor is placed at the right side of a valid hexadecimal sequence. A valid hexadecimal sequence can be composed
of 2 to 4 hexadecimal characters and may or may not be preceded by the 0x or 0X prefix. Examples of valid sequences and the characters they will be converted to:

- 0x0045 will be converted to E
- 0x0125 to ĥ
- 265 to ũ
- 2190 to ←

Note:
For more information about finding the hexadecimal value of a character, see Finding the Decimal, Hexadecimal, or Character Entity Equivalent (on page 466).

Refactoring submenu

Contains a series of actions designed to alter the XML structure of the document:

- **Toggle Comment**
  Encloses the currently selected text in an XML comment, or removes the comment if it is commented.

- **Move Up (Alt + UpArrow)**
  Moves the current node or selected nodes in front of the previous node.

- **Move Down (Alt + DownArrow)**
  Moves the current node or selected nodes after the subsequent node.

- **Split Element (Alt + Shift + D (Ctrl + Alt + D on macOS))**
  Splits the content of the closest element that contains the position of the cursor. Thus, if the cursor is positioned at the beginning or at the end of the element, the newly created sibling will be empty.

- **Join Elements**
  Joins two adjacent block elements (on page 3216) that have the same name. The action is available only when the cursor position is between the two adjacent block elements. Also, joining two block elements can be done by pressing the Delete or Backspace keys and the cursor is positioned between the boundaries of these two elements.

- **Surround with Tags (Ctrl + E (Command + E on macOS))**
  Allows you to choose a tag to enclose a selected portion of content. If there is no selection, the start and end tags are inserted at the cursor position.
• If the **Position cursor between tags** option *(on page 212)* is selected in the **Content Completion** preferences page, the cursor is placed between the start and end tag.

• If the **Position cursor between tags** option *(on page 212)* is not selected in the **Content Completion** preferences page, the cursor is placed at the end of the start tag, in an insert-attribute position.

**Surround with ‘[tag]’ (Ctrl + ForwardSlash (Command + ForwardSlash on macOS))**

Surround the selected content with the last tag used.

**Rename Element**

The element from the cursor position, and any elements with the same name, can be renamed according with the options from the **Rename** dialog box.

**Delete Element Tags**

Deletes the tags of the closest element that contains the position of the cursor. This operation is also executed if the start or end tags of an element are deleted by pressing the **Delete** or **Backspace** keys.

**Remove All Markup**

Removes all the XML markup inside the selected block of content and keeps only the text content.

**Remove Text**

Removes the text content of the selected block of content and keeps the markup intact with empty elements.

**DITA-related Refactoring Actions**

A variety of built-in XML refactoring operations that pertain to DITA documents with some of the information preconfigured based upon the current context.

**Change Topic ID to File Name**

Use this operation to change the ID of a topic to be the same as its file name.

**Convert CALS Tables to Simple Tables**

Use this operation to convert DITA CALS tables to simple tables. If you invoke this operation from a nested table (a table inside a table), only the nested table will be affected. If it is invoked on a parent table that contains nested tables, all of the contained tables will be converted.

**Convert conrefs to conkeyrefs**
Use this operation to convert `@conref` attributes to `@conkeyref` attributes.

**Convert Simple Tables to CALS Tables**

Use this operation to convert DITA simple tables to CALS tables. If you invoke this operation from a nested table (a table inside a table), only the nested table will be affected. If it is invoked on a parent table that contains nested tables, all of the contained tables will be converted.

**Convert to Concept**

Use this operation to convert a DITA topic (of any type) to a DITA Concept topic type (for example, Topic to Concept).

**Convert to General Task**

Use this operation to convert a DITA topic (of any type) to a DITA General Task topic type (for example, Task to General Task). A DITA General Task is a less restrictive alternative to the Strict Task information type.

**Convert to Reference**

Use this operation to convert a DITA topic (of any type) to a DITA Reference topic type (for example, Topic to Reference).

**Convert to Task**

Use this operation to convert a DITA topic (of any type) to a DITA Task topic type (for example, Topic to Task).

**Convert to Topic**

Use this operation to convert a DITA topic (of any type) to a DITA Topic (for example, Task to Topic).

**Convert to Troubleshooting**

Use this operation to convert a DITA topic (of any type) to a DITA Troubleshooting topic type (for example, Topic to Troubleshooting).

**Rename Key**

Available when invoked on a key, and can be used to quickly rename a key. It also updates all references to it. Note that it does not work on DITA 1.3 key scopes.

**Generate IDs**

Use this operation to automatically generate unique IDs for elements.
Contains built-in XML refactoring operations that pertain to attributes with some of the information preconfigured based upon the current context.

**Add/Change attribute**
Allows you to change the value of an attribute or insert a new one.

**Convert attribute to element**
Allows you to change an attribute into an element.

**Delete attribute**
Allows you to remove one or more attributes.

**Rename attribute**
Allows you to rename an attribute.

**Replace in attribute value**
Allows you to search for a text fragment inside an attribute value and change the fragment to a new value.

**Comments Refactoring Actions**
Contains built-in XML refactoring operations that pertain to comments with some of the information preconfigured based upon the current context.

**Delete comments**
Allows you to delete comments found inside one or more elements.

**Elements Refactoring Actions**
Contains built-in XML refactoring operations that pertain to elements with some of the information preconfigured based upon the current context.

**Delete element**
Allows you to delete elements.

**Delete element content**
Allows you to delete the content of elements.

**Insert element**
Allows you to insert new elements.

**Rename element**
Allows you to rename elements.

**Unwrap element**
Allows you to remove the surrounding tags of elements, while keeping the content unchanged.

**Wrap element**
Allows you to surround elements with element tags.

**Wrap element content**

Allows you to surround the content of elements with element tags.

**Fragments Refactoring Actions**

Contains built-in XML refactoring operations that pertain to XML fragments with some of the information preconfigured based upon the current context.

**Insert XML fragment**

Allows you to insert an XML fragment.

**Replace element content with XML fragment**

Allows you to replace the content of elements with an XML fragment.

**Replace element with XML fragment**

Allows you to replace elements with an XML fragment.

**Review submenu**

This submenu includes the following actions:

- **Track Changes**
  Enables or disables the *Track Changes* support for the current document.

- **Accept Change(s) and Move to Next**
  Accepts the *Tracked Change* located at the cursor position or all of the changes in a selection and then moves to the next change. If you select a part of a deletion or insertion change, only the selected content is accepted.

- **Accept All Changes**
  Accepts all *Tracked Changes* in the current document.

- **Reject Change(s) and Move to Next**
  Rejects the *Tracked Change* located at the cursor position or all of the changes in a selection and then moves to the next change. If you select a part of a deletion or insertion change, only the selected content is rejected.

- **Reject All Changes**
  Rejects all *Tracked Changes* in the current document.

- **Comment Change**
  Opens a dialog box that allows you to add a comment to an existing *Tracked Change*. The comment will appear in a callout and a tooltip when
hovering over the change. If the action is selected on an existing commented change, the dialog box will allow you to edit the comment.

**Highlight**

Enables the highlighting tool that allows you to mark text in your document.

**Colors**

Allows you to select the color for highlighting text.

**Stop highlighting**

Use this action to deactivate the highlighting tool.

**Remove highlight(s)**

Use this action to remove highlighting from the document.

**Add Comment**

Inserts a comment at the cursor position. The comment appears in a callout box and a tooltip (when hovering over the change).

**Show/Edit Comment**

Opens a dialog box that displays the discussion thread and allows the current user to edit comments that do not have replies. If you are not the author who inserted the original comment, the dialog box just displays the comment without the possibility of editing it.

**Remove Comment**

Removes a selected comment. If you remove a comment that contains replies, all of the replies will also be removed.

**Manage Reviews**

Opens the Review view (on page 664).

**Manage IDs submenu**

This submenu is available for topics that have an associated DTD or schema. It includes the following actions:

**Rename in**

Renames the ID and all its occurrences. Selecting this action opens the Rename XML ID dialog box. This dialog box lets you insert the new ID value and choose the scope of the rename operation.

**Search References**

Searches for the references of the ID. By default, the scope of this action is the current project. If you configure a scope using the Select the scope for the Search and Refactor operations (on page 831) dialog box, this scope will be used instead.
Search References in

Searches for the references of the ID. Selecting this action opens the Select the scope for the Search and Refactor operations (on page 831).

Search Occurrences in file

Searches for the occurrences of the ID in the current document.

Folding submenu

This submenu includes the following actions:

- **Toggle Fold**
  Toggles the state of the current fold.

- **Collapse Other Folds** *(Ctrl + NumPad/ (Command + NumPad/ on macOS))*
  Folds all the elements except the current element.

- **Collapse Child Folds** *(Ctrl + NumPad. (Command + NumPad. on macOS))*
  Folds the elements indented with one level inside the current element.

- **Expand Child Folds**
  Unfolds all child elements of the currently selected element.

- **Expand All** *(Ctrl + NumPad* (Command + NumPad* on macOS))*
  Unfolds all elements in the current document.

Inspect Styles

Opens the CSS Inspector view (on page 640) that allows you to examine the CSS rules that match the currently selected element.

Options

Opens the Author mode preferences page (on page 176) where you can configure various options with regard to the Author editing mode.

Floating Contextual Toolbar for DITA

Oxygen XML Editor includes a dynamic feature where certain editing contexts will trigger a floating toolbar with common actions that are available in the current editing context.

**Figure 770. DITA Floating Contextual Toolbar**

The floating contextual toolbar is automatically displayed when editing DITA documents in various situations, including:
• When a <p>, <li>, or <shortdesc> element has a selection inside, the floating toolbar includes actions such as **Bold**, _Italic_, _Underline_, a Link submenu, and more.

• When an <image> or <xref> element is selected:
  ◦ If the element has an @href attribute, the floating toolbar includes a URL chooser where you can select the appropriate target.
  ◦ If the element has a @keyref attribute, the floating toolbar includes a drop-down control where you can select the appropriate target key reference.

• When an <object> element is selected:
  ◦ If the element has a @data attribute, the floating toolbar includes a URL chooser where you can select the appropriate target.
  ◦ If the element has a @datakeyref attribute, the floating toolbar includes a drop-down control where you can select the appropriate target key reference.

• When an element with a @conref attribute is selected, the floating toolbar includes actions for editing, removing, or replacing content references.

• When a <codeblock> element is selected, the floating toolbar includes a drop-down control where you can select the value of the @outputclass attribute.

• When a <ul> element is selected, the floating toolbar includes actions for converting it to an ordered list or sorting the list.

• When an <ol> element is selected, the floating toolbar includes actions for converting it to an unordered list or sorting the list.

• When an <li> or <step> element is selected, the floating toolbar includes actions for moving the item up or down in the list/procedure.

• When a <row> or <strow> element is selected in a table, the floating toolbar includes various table-related actions (such as actions for editing table properties, inserting rows, or deleting rows).

• When an <entry> or <stentry> element is selected in a table, the floating toolbar includes various table-related actions (such as actions for editing table properties, inserting/deleting rows, or inserting/deleting columns).

• When a <table> or <simpletable> element is selected, the floating toolbar includes actions for editing table properties or sorting the table.

**DITA Drag/Drop (or Copy/Paste) Actions**

Dragging a file from the Project view (on page 402) or DITA Maps Manager view (on page 2897) and dropping it into a DITA document that is edited in Author mode, creates a link to the dragged file (the <xref> DITA element with the @href attribute) at the drop location. Copy and paste actions work the same.

You can also drag images or media files from your system explorer or the Project view (on page 402) and drop them into a DITA document (or copy and paste). This will insert the appropriate element at the drop or paste location (for example, dropping/pasting an image will insert the DITA <image> element with an @href attribute).
Tip:
For information about customizing Author mode actions for a particular framework (on page 3219) (document type), see the Customizing the Author Mode Editing Experience for a Framework (on page 2145) section.

Related Information:
Customizing the Author Mode Editing Experience for a Framework (on page 2145)

Working with Markdown Documents in DITA

Oxygen XML Editor includes some unique features that allow you to easily integrate Markdown documents in a DITA project. This is especially helpful for teams that have contributors who are familiar with the Markdown syntax, but they want their output to be generated from DITA projects. The integration between the Markdown editor and DITA includes actions to export or convert Markdown documents to DITA topics and the DITA tab in the Preview pane provides a visualization of how the DITA topic will look after conversion. Likewise, the XDITA tab in the Preview pane provides a visualization of how a Lightweight DITA topic will look after conversion. Keys that are defined in the root map are also resolved in the Preview pane.

Export Markdown as a DITA Topic

The Markdown editor includes an option to quickly convert the current Markdown document into a DITA topic. The Export as DITA Topic action is available in the contextual menu.

The conversion creates a new XML file that is defined as a DITA topic and opens it in the Text editing mode. You can then work with the document as you would with any other DITA topic, although you may need to manually correct some issues where the parser could not properly map Markdown syntax to DITA markup.

Working with Markdown Documents in the DITA Maps Manager

Oxygen XML Editor has some specialized features that allow you to integrate Markdown documents directly into your DITA project using the DITA Maps Manager (on page 2897). The following features are available for Markdown documents in the DITA Maps Manager view:

- **Insert Reference to Markdown Document** - You can use the New, Reference, and Reference to the currently edited file actions from the Append Child, Insert Before, or Insert After submenu when invoking the contextual menu in the DITA Maps Manager to insert a reference to a Markdown document at the selected location in the map. Markdown documents will be inserted as a topic reference (topicref element) with the format attribute set to markdown.

- **Validate Markdown Documents in DITA Maps** - When you use the Validate and Check for Completeness action from the DITA Maps Manager toolbar to check the integrity of the structure of a DITA map, Markdown documents that are referenced in the DITA map will be converted to DITA topics in the background and validated the same as any other DITA topic.
• **Transforming DITA Maps with Markdown Documents** - When transforming DITA maps that have Markdown documents referenced, the transformation will convert the Markdown documents to normal DITA output without you needing to manually convert the Markdown documents to DITA topics.

• **Manually Convert Markdown Documents to DITA Topics** - If you need to use DITA semantics that are not possible in Markdown syntax (such as content references, related links, and other DITA-specific syntax), you can manually convert the Markdown document into a DITA topic. To do so, right-click the Markdown document in the DITA Maps Manager and select Refactoring > Convert Markdown to DITA Topic. This will open a dialog box that allows you to configure options for converting the document to an XML file that is defined as a DITA topic.

![Figure 771. Convert Markdown to DITA Topic Dialog Box](image)

This dialog box includes the following options:

**Destination**

The destination path for the new DITA topic.

**File Name**

Presents the current name and allows you to change it.

**Update references**

Select this option to update all references of the file in the DITA map and in the files referenced from the DITA map.

**Delete Markdown file**

If selected, the Markdown version of the file is deleted when the document is converted into a DITA file. If deselected (default value), when the document is converted into a DITA file, the original Markdown file is also preserved in its current location.

**Preview**

Select this button to display a preview of the changes Oxygen XML Editor is about to make.
Convert

Select this button to perform the conversion. If the Markdown file has `format="markdown"`, it will be converted to a DITA topic. If it has `format="mdita"`, it will be converted to a LightWeight DITA topic.

Tip:
Oxygen XML Editor comes with a sample ditamap project for converting Markdown to DITA. Go to the Project view (on page 402), open the sample.xpr project, and navigate to the dita/markdown-dita folder.

Converting Multiple Markdown Documents to DITA

Oxygen XML Editor offers an add-on that contributes actions in the Tools menu and contextual menu to enable batch conversion between various formats, including Markdown to DITA. For more information and instructions for installing the add-on, see Batch Documents Converter Add-on (on page 2533).

DITA-Related Markdown Syntax

For a list of Markdown rules and syntax examples that are specific to DITA, see the Markdown DITA Syntax Reference.

Related information
- Markdown Editor (on page 1263)
- Actions Available in the Markdown Editor (on page 1266)
- Markdown Editor Syntax Rules and Specifications (on page 1277)
- Automatic Validation in Markdown Documents (on page 1274)
- Markdown DITA Syntax Reference

Working with Keys in DITA

DITA uses keys to insert content that may have different values in particular circumstances. Keys provide a way to reference something indirectly. This can make it easier to manage and to reuse content in a various ways.

You can think of keys as like renting a post office box. Instead of the mail going directly from the sender to your house, it now goes to the post office box. You then go to the post office box and bring the mail back to your house. If you move to a new house, your mail still gets to you because it comes to the same post office box. You do not have to send change of address cards to all the people who send you mail. Your mailbox address is the key that makes sure your mail always reaches you, even if you move.

Similarly, if you use keys in your content to reference other content, you do not have to update the source content to change the value of the key or what it points to. You just change the definition of the key.
Defining Keys in DITA Maps

Keys are defined in maps and can then be reused and referenced throughout the whole structure of the map. It is considered best practice to create a separate submap that contains all of the key definitions and reference that submap in the main (root) map (on page 3223). This makes it easier to manage since they're all in one location.

There are two types of key definitions that can be created in a map:

- Key with a value inside a `<keyword>`.
- Key with a target (for example, to target a resource such as an image or external link).

Using Keys for Values

You can use keys to represent values that may vary depending on the type of output. For instance, you may have several products that share a common feature. When you want to describe that feature, you need a way to insert the name of the product, even though that name is different depending on which product the feature description is being used for. For more information, see Working with Variable Text in DITA (on page 3056).

Assigning Keys to Topics

You can assign a key to a topic and use that key to reference that topic for various purposes, such as reuse or linking. As always, keys are defined in maps, so the key definition is done using the keys attribute of the `<topicref>` element:

```xml
<topicref href="quick-heat.dita" keys="feature.quick-heat"/>
```

The easiest way to assign keys to a topic (and insert the `<topicref>` element in its DITA map (on page 3218)) is to use the Keys tab in the Edit Properties dialog box (on page 2934). In the DITA Maps Manager (on page 2897), invoke the contextual menu on the topic that will have the key assigned and select Edit Properties. Go to the Keys tab and enter the name of the key in the Define keys field.

Once a key is assigned to a topic, you can use it to reference that topic for various purposes:

- You can create a link (on page 3072) to it using `xref keyref="feature.quick-heat">`. This allows you to change the target of the link by changing the topic that is pointed to by the key (for example, by profiling).
- You can use it in a map to create a reference to a topic (on page 2917) by key: `<topicref keyref="feature.quick-heat">`. This allows you to change which topic is inserted in the map by the build, by changing the topic that is pointed to by the key.
- You can use it to insert a content reference (on page 3038). In this case, the content reference uses the key to locate the topic to pull content from. It uses a @conkeyref attribute: `<procedure conkeyref="feature.quick-heat/preheat-procedure">`. In this example, `feature.quick-heat` is the key,
and `preheat-procedure` is the ID of a procedure within the topic for that key. Using this mechanism, you could have multiple versions of the preheat procedure in various topics and control which one is inserted by changing the topic that is pointed to by the key.

**Assigning Keys to Graphics**

You can assign a key to an image (using a map to point to the image file *(on page 2932)*) and then insert the image using the key *(on page 2973)*.

Example of a key definition for a targeted image file:

```xml
<map id="keydefs">
  <!-- product name -->
  <title>Key Definitions</title>
  <keydef keys="image1" href="/img/image1.png" format="png"/>
</map>
```

**Related Information:**
- Defining Keys in DITA Maps *(on page 2930)*
- Creating a DITA Content Key Reference *(on page 3038)*
- Reuse Content Dialog Box *(on page 3043)*
- DITA Reusable Components View *(on page 3061)*
- DITA 1.3 Specification: Indirect Key-based Addressing

---

**Working with a Glossary of Terms in DITA**

There are several ways to manage a Glossary of Terms in DITA, but it is considered best practices to create a separate submap for the glossary and embed that glossary map in the *main (root) map* *(on page 3223)*. The actual glossary terms are small *glossary entry* topics that are referenced in the glossary map. You can add links to the glossary terms *(on page 3029)* in the output and you can even define abbreviated forms *(on page 3030)* for terms that have an acronym or some other type of abbreviation.

**How to Create a Glossary of Terms in Oxygen XML Editor**

Even though there are several ways to create a glossary and reference the glossary terms, the following is the recommended approach:

1. Create a new submap *(on page 2915)* for your glossary and embed it in your main map.

2. Create a glossary entry topic (`<glossentry>`) for each glossary term. The `<glossentry>` element may contain numerous *optional glossentry elements*, but every glossentry topic must contain a `<glossterm>` and `<glossdef>` element. The `<glossterm>` is the name of the term while the `<glossdef>` is its definition.

Here is an simple example:
The easiest way to create a glossentry topic in Oxygen XML Editor:

a. Click the New file wizard button on the toolbar.
b. Type glossentry in the search field at the top of the dialog box.
c. Select the Glossentry DITA topic type, configure the name and optionally the title, and click Create.

3. Reference each glossary entry topic in your glossary submap using the <glossref> element. This element requires a @keys attribute. Please make sure the @print attribute is set to yes to show the glossary also in the PDF output.

   <glossref keys="gloss_ddl" href="ddl.dita" print="yes"/>

The easiest way to reference a glossentry in Oxygen XML Editor:

a. With the glossary entry topic opened in the main editor, open the glossary submap in the DITA Maps Manager, right-click the map node and select Append Child > Reference to the currently edited file (if you already have existing glossentry topics, you can right-click the glossentry where you want to insert the new one and select Insert After > Reference to the currently edited file).

   **Step Result:** This opens the Insert Reference dialog box (on page 2923).

b. Go to the Keys tab and enter a name in the Define keys field.
c. Go to the Attributes tab and select Glossary Reference from the Reference type drop-down list at the top of the dialog box.
d. Click Insert and Close.

   **Tip:**
   You could also group multiple glossentry topics into a single collection by using the <glossgroup> element.

**How to Create Links to Glossary Terms**

To specify that a link is generated in the output from the glossary term to its definition, use the <term> element (or <abbreviated-form> element as described in the next section (on page 3030)) with a @keyref attribute that references the corresponding key specified in the <glossref>. Of course, the <glossref> points to the <glossentry> topic where the glossary term is defined.

   <term keyref="gloss_ddl"/>
In the output, the text specified in the `<glossterm>` element is displayed for the glossary term with a link to its glossentry topic that contains its definition.

The easiest way to add a `<term>` element and reference the glossary term in Oxygen XML Editor:

1. Place the cursor at the location where you want to insert a link to the glossary term.
2. In the DITA Reusable Components view (on page 3061), go to the Keys tab and use the search filter field at the top of the view to find the key for the particular glossary term.
3. Right-click the key and select Insert as Keyref > More > Term.

Using Abbreviated Forms (Acronyms) with Glossary Terms

The `<abbreviated-form>` element can be used for glossary terms that you want to appear in an abbreviated form (such as an acronym). Abbreviated forms are expanded to their full form the first time that they appear in a document, and then all subsequent instances will display the short form (or acronym). You would need to define the long and short forms in the `<glossentry>` and then reference it with the `<abbreviated-form>` element (instead of the `<term>` element).

The recommended best practices for defining the long and short forms would be to use a structure similar to this:

```xml
<glossentry id="ddl">
  <glossterm>Data Definition Language</glossterm>
  <glossBody>
    <glossSurfaceForm>Data Definition Language (DDL)</glossSurfaceForm>
    <glossAlt>
      <glossAcronym>DDL</glossAcronym>
    </glossAlt>
  </glossBody>
</glossentry>
```

The long form is declared using the `<glossSurfaceForm>` element while the short form is declared using the `<glossAcronym>` element.

Then you need to reference the glossentry that contains the long and short forms using the `<abbreviated-form>` element:

```xml
<abbreviated-form keyref="gloss_ddl"/>
```

For more information about the recommended best practices for using abbreviations, including information about using multiple languages, see: http://www.oasis-open.org/committees/download.php/29734/AcronymBestPractice_08112008.doc.

Related information
https://docs.oasis-open.org/dita/v1.2/os/spec/langref/glossentry.html
https://docs.oasis-open.org/dita/v1.2/os/spec/langref/abbreviated-form.html
Reusing DITA Content

Reusing content is one of the key features of DITA and DITA provides several methods for reusing content. Oxygen XML Editor provides support for each of these methods.

Reusing Topics in DITA Maps

A DITA topic does not belong to any one publication. You add a DITA topic to a publication by referencing it in a map. You can reference the same topic in multiple maps (on page 3034).

Reusing Content with References and Keys

DITA allows you to reuse content by referencing it in another topic. DITA provides several mechanisms for including content by reference (on page 3035) (conref, conkeyref, coderref). A conref (content reference) (on page 3036) creates a direct reference to a specific element of another topic. A conkeyref (content key reference) (on page 3038) creates a reference to a key, which then points to a specific element in another topic. The advantage of using a conkeyref is that you can change the element that is included by changing the key reference. For example, since keys are defined in maps, if you include a topic in multiple maps, you can use a different key reference in each map. A coderref references an external file that contains literal code.

Oxygen XML Editor provides support for all of these mechanisms.

While the conref and conkeyref mechanisms can be used to reference any content element, it is considered best practice to only conref or conkeyref content that is specifically set and managed as reusable content. This practice helps reduce expensive errors, such as an author accidentally deleting the source element that other topics are including by the reference. Oxygen XML Editor can help you create a reusable component from your current content.

Reusing Content with Reusable Components

DITA allows you to select content in a topic, create a reusable component (on page 3054) from it and reference that component in other locations. Each reusable component is created as a separate file. Anytime the content needs to be edited, you only need to update it in the component file and all the locations in your topics that reference it will also be updated. This can help you to maintain continuity and accuracy throughout your documents.

Reusing Content with Variables

DITA allows you to replace the content of certain elements with a value that is pointed to by a key. This mechanism effectively means that you can create variables in your content (on page 3056), which you can then create multiple outputs by changing the value that the key points to. This is done by profiling the definition of the key value, or by substituting another map with a different key value.

Reusing Content with DITA 1.3 Concepts

DITA 1.3 allows you to use some advanced concepts to expand content reuse possibilities even further. Key Scopes (or scoped keys) (on page 3058) allow you to reuse topics with variable content depending on the
particular context and it maximizes reuse possibilities for keys. *Branch Filtering (on page 3060)* allows you to reuse the same content that is profiled in multiple ways within the same publication, each time using a different filter.

**DITA Reusable Components View**

If you use a large amount of keys or reusable components in your DITA project, the DITA Reusable Components view (on page 3061) can be quite helpful. It collects all of the keys and reusable components that are defined in the root map (on page 3223) and presents them in a dynamic table where you can easily locate and insert references to them.

**Reuse Actions in Oxygen XML Editor**

Oxygen XML Editor includes some actions that are specifically designed for DITA reusable content. These actions are available in the contextual menu, the DITA menu, and some are available on the toolbar.

**Reuse Content**

This action provides a mechanism for reusing content fragments. It opens the Reuse Content dialog box (on page 3043) that allows you to insert several types of references to reusable content at the cursor position. The types of references that you can insert using this dialog box include content references (@conref) (on page 3044), content key references (@conkeyref) (on page 3046), or key references to metadata (@keyref) (on page 3049).

**Push Current Element**

Opens the Push current element dialog box (on page 3052) that allows content from a source topic to be inserted into another topic without any special coding in the topic where the content will be re-used.

**Edit Content Reference**

This action is available for elements with a @conref or @conkeyref attribute. It opens the Edit Content Reference dialog box that allows you to edit the source location (or key) and source element of a content reference (or content key reference), and the reference details (@conref/@conkeyref and @conrefend attributes). For more information, see Reuse Content Dialog Box (on page 3043).

**Replace Reference with Content**

Replaces the referenced fragment (@conref or @conkeyref) at the cursor position with its content from it source. This action is useful if you want to make changes to the content in the currently edited document without changing the referenced fragment in its source location. If the source content includes references to other topics/resources (hrefs), the operation also resolves those references relative to the new location. Attributes are preserved according to the following priority:
1. Attributes from the elements in the current document that reference other content are preserved except for attributes with a `dita-use-conref-target` value.
2. Attributes from the referenced content are brought into the replaced elements in the current document except for `id` attributes.

**Replace All References with Content**

Replaces all referenced fragments (`@keyref`, `@conref`, or `@conkeyref`) in the current document with the content. Attributes are preserved according to the following priority:

1. Attributes from the elements in the current document that reference other content are preserved except for attributes with a `dita-use-conref-target` value.
2. Attributes from the referenced content are brought into the replaced elements in the current document except for `id` attributes.

For `keyrefs` inside `<xref>` or `<link>` elements, the `@keyref` attribute is changed to an `@href` attribute, while the rest of the content for the `keyref` is replaced with its source content.

If the source content includes references to other topics/resources (`hrefs`), the operation also resolves those references relative to the new location.

**Remove Content Reference**

Removes the content reference (`@conref` or `@conkeyref`) inside the element at the cursor position.

**Create Reusable Component**

Opens a dialog box that helps you to create a reusable component from the current element or selection of elements. If the **Replace selection with content reference** option is selected in the dialog box, the selection will be replaced with a content reference (`@conref`). If multiple elements are selected (for example, multiple steps or list items), the selection will be replaced with a content reference range (`@conref` and `@conrefend`). For more information, see Creating a Reusable Content Component (on page 3055).

**Insert Reusable Component**

Inserts a reusable component at cursor location. For more information, see Inserting a Reusable Content Component (on page 3056).

**Resources**

For more information about reusing strategies in DITA, see the following resources:

- Webinar: Working with DITA in Oxygen - Basic Profiling and Reuse Strategies
- Webinar: Working with DITA in Oxygen - Advanced Profiling and Reuse Strategies

**Related information**

Working with Keys in DITA (on page 3026)
Reusing DITA Topics in Multiple Maps

You can reuse an entire DITA topic simply by referencing it in multiple maps (or multiple locations within the same map (on page 2919)) using one of the following procedures:

Reuse Topics Using the DITA Maps Manager

1. Make sure the DITA map (on page 3218) is opened in the DITA Maps Manager (on page 2897).
2. Add a reference to an existing topic by using one of the following methods (depending on your particular situation):
   a. If the topic already exists in this DITA map, do one of the following:
      • Simply drag the topic and press Ctrl (or Alt on macOS) at the new location within the map (or use the Copy and Paste contextual menu actions).
      • If the topic is the currently open document in the main editor, determine the new location in the map (in the DITA Maps Manager (on page 2897)), right-click a parent or sibling topic, and select Append Child > Reference to the currently edited file or Insert After > Reference to the currently edited file.
   b. If the topic already exists in another DITA map, do one of the following:
      • Open the other map in the DITA Maps Manager (on page 2897), right-click the topic, select Copy, switch back to the original DITA map in the DITA Maps Manager, determine the new location in the map, right-click a parent or sibling topic, and use one of the Paste contextual menu actions (Paste, Paste Before, or Paste After).
      • If the topic is the currently open document in the main editor, determine the new location in the map (in the DITA Maps Manager (on page 2897)), right-click a parent or sibling topic, and select Append Child > Reference to the currently edited file or Insert After > Reference to the currently edited file.
   c. If the topic exists in the project, but has not yet been added to a DITA map, do one of the following:
      • Right-click the topic in the Project view (on page 402) (or the file system), select Copy, switch to the DITA Maps Manager (on page 2897) view, determine the new location in the map, right-click a parent or sibling topic, and use one of the Paste contextual menu actions (Paste, Paste Before, or Paste After).
      • If the topic is the currently open document in the main editor, determine the new location in the map (in the DITA Maps Manager (on page 2897)), right-click a parent or sibling topic, and select Append Child > Reference to the currently edited file or Insert After > Reference to the currently edited file.
3. If your topic uses a key reference (on page 3026), set up the appropriate key definition in your map (on page 2930).
4. If you want to define relationships between topics, other than those defined in the topics themselves, you can add a relationship table to your map (on page 3078).
5. When you have finished adding topics, check that your map is complete and that all topic links and keys resolve correctly. To do this validation, click the Validate and Check for Completeness action (on page 2941) on the toolbar in the DITA Maps Manager.
Reuse Topics Using Author Mode Editor

1. Open the DITA map (on page 2895) in the Author mode editor.
2. Add a reference to an existing topic by dragging it from the Project view (on page 402) (or the file system) and dropping it in the desired location in the DITA map opened in Author mode. You can also accomplish the same thing by using the Copy and Paste contextual menu actions.
3. If your topic uses a key reference (on page 3026), set up the appropriate key definition in your map (on page 2930).
4. If you want to define relationships between topics, other than those defined in the topics themselves, you can add a relationship table to your map (on page 3078).
5. When you have finished adding topics, check that your map is complete and that all topic links and keys resolve correctly. To do this validation, click the Validate and Check for Completeness action (on page 2941) on the toolbar in the DITA Maps Manager.

Displaying Multiple References to the Same Topics

Whenever multiple references to the same topic are detected in the context of the current map in the DITA Maps Manager (on page 2897), an indicator will appear in the top-right corner of the Author mode editor that shows the number of times the current topic is referenced in the DITA map. It also includes navigation arrows that allow you to jump to the next or previous reference in the DITA Maps Manager.

Working with Content References

The DITA content reference feature lets you insert a piece of source content by referencing it from its source. When you need to update that content, you only need to do it in one place. The source content can be referenced using the DITA @conref or @conkeyref attributes.

There are several strategies for managing content references:

- **Reusable components** - With this strategy, you create a new file for each piece of content that you want to reuse and you insert references from the content of the reusable component files. For example, suppose that you have a disclaimer that needs to be included in certain sections of your documentation. You can create a reusable component that contains your disclaimer and reuse it as often as you need to. If the disclaimer ever needed to be updated, you only have to edit it in one file.
- **Single-source content references** - You may prefer to keep many pieces of reusable content in one file. For example, you might want to create a single file that contains all the actions that are available in various menus or toolbars for your software application. Then, wherever you need to describe or display an action in your documentation, you can reuse content from that single file by inserting content references. This strategy requires more setup than reusable components, but might make it easier to centrally managing the reused content and it allows for more flexibility in the XML structure of the reusable content.
• **Arbitrary content references** - Although it is not recommended, you can create content references among topics without storing the reusable content in components or a single file. This strategy might make it difficult to manage content that is reused and to maintain continuity and accuracy, since you may not have any indication that content you are editing is reused elsewhere.

A reference to the external content is created by adding a [@conref](#) or [@conkeyref](#) attribute to an element in the local document. The @conref or @conkeyref attribute defines a link to the referenced content, made up of a path to the file and the topic ID within the file. The path may also reference a specific element ID within the topic. Referenced content is not physically copied to the referencing file. However, by default Oxygen XML Editor displays it in **Author** mode as if it is there in the referencing file. If you do not want referenced content displayed, open the **Preferences** dialog box (**Options > Preferences**) (on page 124), go to **Editor > Edit modes > Author**, and deselect the **Display referenced content** option (on page 179).

**Note:**
A reference also displays [tracked changes](#) and comments that are included in the source fragment. To edit these comments (or accept/reject changes) right-click the comment or tracked change and select **Edit Reference**.

**Tip:**
To search for references made through a direct content reference, use the **Search References** action from the contextual menu.

**Related Information:**
- Working with Reusable Components (on page 3054)
- Working with Keys in DITA (on page 3026)
- Working with the Conref Push Mechanism (on page 3052)
- DITA Reusable Components View (on page 3061)

**Creating a DITA Content Reference**

**DITA Content Reference**

A DITA content reference, or **conref**, is one of the main content reuse features of DITA (on page 3031). It is a mechanism for re-using the same content in multiple topics (or even in multiple locations within the same topic).

For a **conref** to be created, the source content must have an **id** attribute that the **conref** can reference. Therefore, creating a **conref** requires that you add an **id** to the content to be reused before inserting a **conref** into the topic that reuses the referenced content.
Assigning an ID to the Referenced Content

To add an id to a DITA element in a topic, place the cursor on the element and select Edit Attributes from the contextual menu (or simply press Alt+Enter) to open the in-place attribute editor (on page 630). Enter id as the Name of the attribute and a value of your choice in the Value field. You can also use the Attributes view (on page 627) to enter a value in the id attribute.

Note:
The element may already have an id, since in some cases, Oxygen XML Editor automatically generates an ID value when the id attribute is created.

Creating a Content Reference

To create a content reference (conref), follow these steps:

1. Make sure the element you want to reference has an ID assigned to it (on page 3037).
2. In Author mode (on page 354), place the cursor at the location where you want the reused content to be inserted.
3. Select the Reuse Content action on the main toolbar (or from the DITA menu or Reuse submenu of the contextual menu). The Reuse Content dialog box (on page 3043) is displayed.
4. In the Location field of the Reuse Content dialog box, select the topic that contains the element you want to reference. The elements that you can reference are presented in a table.
5. Select the Target ID of the element (or elements) to have their content inserted, and verify the content in the Preview pane. The id value of the element that you select is automatically added to the Reference to (conref) field.
6. Make any other selections you need in the Reuse Content dialog box (on page 3043). If you select multiple elements, the Expand to (conrefend) field is automatically filled with the id value of the last element in your selection.
7. Click Insert or Insert and close to create the content reference.

Using Copy/Paste Actions to Create a Content Reference

Oxygen XML Editor also includes support for creating content references with simple copy/paste actions. The copied content must be an entire DITA XML element with an ID attribute. Also, the location in the document where you paste the element must be valid, although as long as the Smart paste and drag and drop option (on page 181) is selected in the Schema-Aware preferences page, if you try to paste it in an invalid location, Oxygen XML Editor will attempt to place it in a valid location, and may prompt you with one or more choices for where to place it.

To create a content reference (conref) using copy/paste actions, follow these steps:
1. Copy an entire DITA element that has an ID attribute assigned to it.
2. Place the cursor at a location where the copied element will be valid.
3. Select the **Paste as Content Reference** action from the **Paste Special** submenu from the contextual menu.

**Other Ways to Reuse Content**

- You can use the **Components** tab in the **DITA Reusable Components** view (on page 3065) to easily insert content references.
- An alternate way to reuse content is to use the Oxygen XML Editor **Create Reusable Component** (on page 3055) and **Insert Reusable Component** (on page 3056) actions (available in the **DITA** menu and the **Reuse** submenu of the contextual menu). They handle the details of creating an ID and *conref* and create reusable component files, separate from your normal content files. This can help you manage your reusable content more effectively.
- You can also insert reusable content using content key references (on page 3038). This may also make reusable content easier to manage, depending on your particular situation and needs.
- Other topics in this section include information about more specialized or advanced ways or reusing content, such as **code references** (on page 3051), the **conref push mechanism** (on page 3052), **variable text** (on page 3056), **key scopes** (on page 3058), and **branch filtering** (on page 3060).

**Related Information:**
- **Reuse Content Dialog Box** (on page 3043)
- **DITA Reusable Components View** (on page 3061)
- **Creating a DITA Content Key Reference** (on page 3038)
- **Editing DITA Content References** (on page 3040)
- **Working with Reusable Components** (on page 3054)
- **Working with Content References** (on page 3035)

**Creating a DITA Content Key Reference**

**DITA Content Key Reference**

A DITA content key reference, or **@conkeyref**, is a mechanism for inserting a piece of content from one topic into another. It is a version of the **DITA content reference mechanism** (on page 3036) that uses **keys** (on page 3026) to locate the content to reuse rather than direct references to topics that contain reused content.

As with a **conref**, a **conkeyref** requires that the element to be reused has an **@id** attribute. It also requires the topic that contains the reusable content to be assigned a **key** (on page 3026) in a map. As with all uses of keys, you can substitute multiple maps or use **profiling** (on page 3133) to create multiple definitions of keys in a single map. This allows the same **@conkeyref** to pull in content from various sources, depending on how your build is configured. This can make it easier to create and manage sophisticated content reuse scenarios.
Creating a Content Key Reference

To create a content key reference (@conkeyref), follow these steps:

1. Make sure the topic that contains the reusable content is assigned a key in the DITA map and the element you want to reference has an ID assigned to it.
2. In **Author mode (on page 354)**, place the cursor at the location where you want the reused content to be inserted.
3. Select **Reuse Content** on the main toolbar (or from the DITA menu or **Reuse** submenu of the contextual menu). The **Reuse Content dialog box (on page 3043)** is displayed.
4. Select the **Key** radio button for the content source and use the **Choose Key Reference** button to select the key for the topic that contains the reusable content (you can also select one from the drop-down list in the **Key** field). The elements that you can reference from the source are presented in the table in the middle of the **Reuse Content** dialog box.
5. Select the **Target ID** of the element (or elements) that you want to insert, and verify the content in the **Preview** pane. The @id value of the element that you select is automatically added to the **Reference to (conkeyref)** field.
6. Make any other selections you need in the **Reuse Content dialog box (on page 3046)**. If you select multiple elements, the **Expand to (conrefend)** field is automatically filled with the @id value of the last element in your selection.
7. Click **Insert** or **Insert and close** to create the content reference.

**Note:**
If you are using **Text mode (on page 353)**, when you insert a @conkeyref attribute, after you enter the first quote (@conkeyref=”), the **Content Completion Assistant** will list all the defined keys that you can select from. Also, after you select the key, the **Content Completion Assistant** will then list the element IDs from the referenced topic, allowing you to insert an anchor. Note that this only works for local files.

Using Copy/Paste Actions to Create a Content Key Reference

Oxygen XML Editor also includes support for creating content key references with simple copy/paste actions. When the DITA content is processed, the key references are resolved using key definitions from **DITA maps**.

The copied content must be an entire DITA XML element with an ID attribute and the topic that contains the reusable content must have a key assigned in a **DITA map**. Also, the location in the document where you paste the element must be valid, although as long as the **Smart paste and drag and drop option (on page 181)** is selected in the **Schema-Aware** preferences page, if you try to paste it in an invalid location, Oxygen XML Editor will attempt to place it in a valid location, and may prompt you with one or more choices for where to place it.

To create a content key reference (@conkeyref) using copy/paste actions, follow these steps:
1. In the DITA Maps Manager view (on page 2897), make sure that the Context combo box (on page 2901) points to the correct map that stores the keys.

2. Make sure the topic that contains the content you want to reference has a key assigned to it. To assign a key, right-click the topic with its parent map opened in the DITA Maps Manager (on page 2897), select Edit Properties, and enter a value in the Keys field.

3. In a topic with an assigned key, copy an entire DITA element that has an ID attribute assigned to it.

4. Place the cursor at a location where the copied element will be valid.

5. Select the Paste as Content Key Reference action from the Paste Special submenu from the contextual menu.

Other Ways to Reuse Content

- You can use the Components tab in the DITA Reusable Components view (on page 3065) to easily insert content key references.
- You can also insert reusable content using content references (conref) (on page 3036).
- Other topics in this section include information about more specialized or advanced ways or reusing content, such as code references (on page 3051), the conref push mechanism (on page 3052), variable text (on page 3056), key scopes (on page 3058), and branch filtering (on page 3060).

Related Information:

Reuse Content Dialog Box (on page 3043)
DITA Reusable Components View (on page 3061)
Creating a DITA Content Reference (on page 3036)
Editing DITA Content References (on page 3040)
Working with Reusable Components (on page 3054)
Working with Content References (on page 3035)

Editing DITA Content References

When you reference reusable content using a @conref or @conkeyref attribute, by default, the content is grayed out in the document and can only be edited from the source document. To edit the source of the referenced content, click the icon at the beginning of the inserted content. This will open the source document where you can edit the referenced content.

Oxygen XML Editor also includes some actions that allow you to quickly edit existing content references. When the element that contains a content reference (@conref or @conkeyref) is selected, the following actions are available in the DITA menu and the Reuse submenu of the contextual menu:

Edit Content Reference

This action is available for elements with a @conref or @conkeyref attribute. It opens the Edit Content Reference dialog box that allows you to edit the source location (or key) and source element of a content reference (or content key reference), and the reference details.
Replace Reference with Content

Replaces the referenced fragment (@conref or @conkeyref) at the cursor position with its content from its source. This action is useful if you want to make changes to the content in the currently edited document without changing the referenced fragment in its source location. If the source content includes references to other topics/resources (hrefs), the operation also resolves those references relative to the new location. Attributes are preserved according to the following priority:

1. Attributes from the elements in the current document that reference other content are preserved except for attributes with a -dita-use-conref-target value.
2. Attributes from the referenced content are brought into the replaced elements in the current document except for gid attributes.

Replace All References with Content

Replaces all referenced fragments (@keyref, @conref, or @conkeyref) in the current document with the content. Attributes are preserved according to the following priority:

1. Attributes from the elements in the current document that reference other content are preserved except for attributes with a -dita-use-conref-target value.
2. Attributes from the referenced content are brought into the replaced elements in the current document except for gid attributes.

For keyrefs inside <xref> or <link> elements, the @keyref attribute is changed to an @href attribute, while the rest of the content for the keyref is replaced with its source content.

If the source content includes references to other topics/resources (hrefs), the operation also resolves those references relative to the new location.

Remove Content Reference

Removes the content reference (@conref or @conkeyref) inside the element at the cursor position.

Converting Conrefs to Conkeyrefs

Oxygen XML Editor includes a DITA refactoring operation called Convert conrefs to conkeyrefs that will find all content references (that reference content outside the current document) and convert them to content key references. You can also use it to quickly convert all content references in the current document or multiple documents at once.

To access the Convert conrefs to conkeyrefs operation, use one of the following methods:

Single Document Method
With the document opened in the editor, right-click anywhere in the main editing pane (or right-click the topic reference in the DITA Maps Manager (on page 2897)), go to the Refactoring submenu, and choose Convert conrefs to conkeyrefs.

**Multiple Documents At Once Method**

Select XML Refactoring from the Tools menu (or from the Refactoring submenu when you right-click a document in the Project view (on page 402) or the DITA Maps Manager view (on page 2897)). Then select Convert conrefs to conkeyrefs from the DITA section and click Next.

Either method will proceed to the XML Refactoring Wizard. If you used the Multiple Documents At Once Method (on page 3042), the wizard page allows you to choose a scope for the operation and some filtering options:

- **Scope** - Select from a variety of options to define the scope that will have resources affected by the operation. For example, you can choose to affect all resources in the Project, All opened files, Current DITA map hierarchy, or just the Current file.
- **Filters** section
  - Include files - Specifies files to be excluded from the operation. You can specify multiple files by separating them with commas and the patterns can include wildcards (such as * or ?).
  - Restrict to known XML file types only - Excludes non-XML file types from the operation.
  - Look inside archives - If this option is selected, the scope of the operation will include files inside archives.

If you used the Single Document Method (on page 3041), the scope will be the current file so the scope and filtering options are not displayed.

You can then use one of the following buttons to proceed with the operation:

**Preview**

You can use the Preview button to open a comparison panel where you can review all the changes that will be made by the refactoring operation before applying the changes.

**Warning:**

It is always recommended to use the Preview button to make sure the operation is not going to do something unexpected and after you click the Finish button, any Undo action will only revert changes on the current document.

**Finish**

When you use the Finish button, the operation will be processed and all content references will be converted to content key references (either all content references in the current document or all content references in all of the documents specified in the scope). The file name for each converted document is used as the value for its new key. However, the operation does NOT automatically add the key to the DITA Map (on page 3218), so you still need to manually define each key in your DITA map (on page 2930).
Reuse Content Dialog Box

The Reuse Content dialog box provides a mechanism for reusing content fragments. DITA @conref, @conkeyref, and @keyref attributes can be used to insert references to reusable content. The @conref attribute stores a reference to another element and is processed to replace the referencing element with the referenced element. The @conkeyref attribute uses keys (on page 3026) to locate the content to reuse rather than direct references to the topic that contains the reusable content. The @keyref attribute also uses keys (on page 3026) and can be used to indirectly reference metadata that may have different values in various circumstances.

Note:
For a conref or conkeyref, to reference the content inside a DITA element, the source element must have an @id attribute assigned to it. The element containing the content reference acts as a placeholder for the referenced element. For more details about DITA @conref and @conkeyref attributes, go to https://www.oxygenxml.com/dita/1.3/specs/archSpec/base/conref.html.

Note:
For the purposes of using a @keyref, keys are defined at map level and referenced afterward. For more information about the DITA @keyref attribute, go to https://www.oxygenxml.com/dita/1.3/specs/langRef/attributes/thekkeyrefattribute.html.

Oxygen XML Editor displays the referenced content (on page 595) of a DITA content reference if it can resolve it to a valid resource. If you use URIs instead of local paths in your XML documents and your DITA-OT transformation needs an XML Catalog (on page 3225) to map the URIs to local paths, you need to add the catalog in Oxygen XML Editor (on page 826). If the URIs can be resolved, the referenced content is displayed in Author mode and in the transformation output.

In Author mode, a reference to reusable content (@conref, @conkeyref, or @keyref) can easily be inserted at the cursor position by using the Reuse Content dialog box. It can be opened with any of the following methods:

- Click the Reuse Content action on the main toolbar.
- In the contextual menu of the editing area, go to Reuse > Reuse Content.
- Go to DITA > Reuse Content.

Your selection at the top of the dialog box for choosing the content source determines whether Oxygen XML Editor will insert a @conref, @conkeyref, or @keyref.
If you select **Location** for the content source, a *content reference* (@conref) will be inserted. If you select **Key** for the content source, keys will be used to insert a *content key reference* (@conkeyref) or a *key reference* (@keyref).

### Content Reference (@conref) Options Using the Reuse Content Dialog Box

**Figure 772. Reuse Content Dialog Box (with the Default Insert Content Reference Options Displayed)**

![Reuse Content Dialog Box](image)

**Choose the content source** Section

When **Location** is selected for the content source, a *content reference* (@conref) will be inserted. Here you can specify the path of the topic that contains the content you want to reference.

The dialog box offers the following options:

- **Select an element from the content source** Section
Show elements of type

You can use this drop-down list to select specific types of elements to be displayed in the subsequent table. This can help you narrow down the list of possible source elements that you can select.

Text Filter Field

You can also use the text filter field to narrow down the list of possible source elements to be displayed in the subsequent table.

Element Table

Presents all the element IDs defined in the source topic. Use this table to select the Target ID of the element that you want to reference. You can select multiple contiguous elements to reference a block of content.

Preview Pane

Displays the content that will be references. If you select multiple elements in the element table, the content from all the selected elements is displayed.

Source Pane

Displays the source code of the element to be referenced.

Reference details Section

Reference to (conref)

Oxygen XML Editor automatically fills this text field with the value of the @conref attribute to be inserted. However, you can edit this value if need be.

Reference to range end (conrefend)

If you select multiple elements (of the same type) in the element table, Oxygen XML Editor automatically fills this text field with the @id value of the last element in your selection. This value will be inserted as a @conrefend attribute, defining the end of the conref range.
Content Key Reference (@conkeyref) Options Using the Reuse Content Dialog Box

Choose the content source Section

When **Key** is selected for the content source, you can use keys to reference content. You can use the **Choose Key Reference** button to open the **Choose Key** dialog box that allows you to select one from a list of all the keys that are gathered from the root map (on page 3223) (you can also select one from the drop-down list in the **Key** field).

**Note:**
If the current DITA map is not selected as the root map, no keys will be listed.

**Tip:**
You can also use the DITA Reusable Components view (on page 3061) for similar purposes.
The **Choose Key** dialog box includes the following:

- **Change Root Map** - Opens a small dialog box that allows you to select a root map *(on page 2914)*.
- **Search Filter** - You can enter text in the filter field at the top of the dialog box to filter the list and search for a specific key.
- **Sortable Columns** - The dialog box includes the following columns that can be sorted by clicking on the heading:
  - **Key** - The name of the key (the value of the `@keys` attribute).
  - **Description** - The description of the key that is obtained from its definition. Keys that are defined with a text value in the `<navtitle>` or `<keyword>` element have that value listed in this column.
  - **Href** - Keys that are defined with a value in an `href` attribute have that href value listed in this column.
  - **Definition Location** - The name of the **DITA map** *(on page 3218)* where the key is defined.
- **Group by Definition Location** - A contextual menu action that can be used to group (and sort) all the keys based upon the value in the **Definition Location** column.

To insert a **content key reference** (`@conkeyref`), select the key that contains the content you want to reference. Notice that the file path is shown in the **Href** column. Keys that do not have a value in the **Href** column are for referencing metadata with a `@keyref` attribute. Therefore, to insert a `@conkeyref`, you need to select a key that does have a value (file path) in the **Href** column.

After you select a key, click **OK** to return to the **Reuse Content** dialog box.

When a key that is defined as a **content key reference** has been selected, the **Reuse Content** dialog box offers the following additional options for inserting a `@conkeyref`:
Select an element from the content source Section

Show elements of type

You can use this drop-down list to select specific types of elements to be displayed in the subsequent table. This can help you narrow down the list of possible source elements that you can select.

Text Filter Field

You can also use the text filter field to narrow down the list of possible source elements to be displayed in the subsequent table.

Element Table

Presents all the element IDs defined in the source topic. Use this table to select the Target ID of the element that you want to reference. You can select multiple contiguous elements to reference a block of content.

Preview Pane

Displays the content that will be referenced. If you select multiple elements in the element table, the content from all the selected elements is displayed.

Source Pane

Displays the source code of the element to be referenced.

Reference details Section

Reference type

The type of reference that will be inserted. If you selected a key that references a DITA resource, you will notice that conkeyref value is automatically selected.

Reference to

Oxygen XML Editor automatically fills this text field with the value of the @conkeyref attribute to be inserted. However, you can edit this value if need be.

Fallback to (conref)

You can select this option to define a @conref attribute to be used as a fallback to determine the content reference relationship if the specified conkeyref cannot be resolved.

Reference to range end (conrefend)

If you select multiple elements (of the same type) in the element table, Oxygen XML Editor automatically fills this text field with the @id value of the last element in your selection. This value will be inserted as a @conrefend attribute, defining the end of the conkeyref range.
Key Reference to Metadata (@keyref) Options Using the Reuse Content Dialog Box

Figure 775. Insert Key Reference Options

Choose the content source: Section

When Key is selected for the content source, you can use keys to reference content. You can use the Choose Key Reference button to open the Choose Key dialog box that allows you to select one from a list of all the keys that are gathered from the root map (on page 3223) (you can also select one from the drop-down list in the Key field).

Note:
If the current DITA map is not selected as the root map, no keys will be listed.

Tip:
You can also use the DITA Reusable Components view (on page 3061) for similar purposes.
The **Choose Key** dialog box includes the following:

- **Change Root Map** - Opens a small dialog box that allows you to select a root map (on page 2914).

- **Search Filter** - You can enter text in the filter field at the top of the dialog box to filter the list and search for a specific key.

- **Sortable Columns** - The dialog box includes the following columns that can be sorted by clicking on the heading:
  - **Key** - The name of the key (the value of the `@keys` attribute).
  - **Description** - The description of the key that is obtained from its definition. Keys that are defined with a text value in the `<navtitle>` or `<keyword>` element have that value listed in this column.
  - **Href** - Keys that are defined with a value in an `href` attribute have that href value listed in this column.
  - **Definition Location** - The name of the DITA map (on page 3218) where the key is defined.

- **Group by Definition Location** - A contextual menu action that can be used to group (and sort) all the keys based upon the value in the **Definition Location** column.

To insert a *key reference* to metadata (`@keyref`), select the key you want to reference. Keys that do not have a value in the `Href` column are for referencing metadata with a `@keyref` attribute. Therefore, to insert a `@keyref`, you need to select a key that does not have a value (file path) in the `Href` column.

After you select a key, click **OK** to return to the **Reuse Content** dialog box.

When a key that references metadata has been selected, the **Reuse Content** dialog box offers the following additional options for inserting a `@keyref`:
Select an element from the content source Section

This section is not used when referencing metadata.

Reference details Section

Reference type

The type of reference that will be inserted. If you selected a key that does not reference a DITA resource, you will notice that keyref value is automatically selected.

Reference to

Oxygen XML Editor automatically fills this text field with the value of the @keyref attribute to be inserted.

Element name

Oxygen XML Editor automatically selects the element that is most commonly used for the selected type of key reference, but you can use the drop-down list to choose another element to use for the reference.

Finalizing Your Content Reference Configuration

Once you click Insert or Insert and close, the configured content reference is inserted into your document.

Tip:

You can easily insert multiple content references by keeping the Reuse Content dialog box opened, using the Insert button.

Related Information:

DITA Reusable Components View (on page 3061)
Working with Content References (on page 3035)

Working with Code References

Code References

The DITA <coderef> element can be used to reference an external file that contains literal code. This is especially useful if you need to reference code from an external source that may occasionally change. Another advantage is that you don't have to convert illegal characters into their character equivalents. When the <coderef> is processed, the referenced code file is imported and delimiting characters (such as < or >) are displayed as standard text, rather than treated as XML markup.

For more information about code references, see DITA 1.3 Specification: Coderef.
Example of using a Coderef

This code is an example of how to use a coderef.

```
<codeblock><coderef href="MyExternalCode.xsl"/></codeblock>
```

Defining Line Ranges

DITA-OT provides additional code reference processing support that allows you to define line ranges in case you only want to reference certain parts of the external file, rather than the whole file.

For information and examples of how to define line ranges, see DITA Open Toolkit Documentation: Extended Code Reference Processing.

Working with the Conref Push Mechanism

Content Reference Push Mechanism

The usual method of using content references pulls element content from a source element and inserts it in the current topic. DITA 1.2 introduced an alternative method of content referencing, allowing element content to be pushed, or injected, from a source topic to another topic without any special coding in the topic where the content will be re-used. This technique is known as a content reference push mechanism (conref push).

The conref push mechanism requires elements in the target topic (the topic where the content is to be pushed) to have ID elements, as the push mechanism inserts elements before or after a named element, or replaces the named element. Assuming the source topic is included in the DITA map (on page 3218), the conref push will be processed during the publishing stage for the DITA map.

Example of a Conref Push Scenario

An example of a scenario where a conref push would be useful is where a car manufacturer produces driver manuals that are distributed to various regions with their own specific regulations and certain sections need to be customized by the local car dealers before publishing. The local dealer could use a conref push technique to insert specific content without modifying the manufacturer-supplied content.

Push Current Element Action

Oxygen XML Editor includes an action that allows you to easily reference content with a conref push mechanism. The Push Current Element action is available in the DITA menu and in the Reuse subfolder of the contextual menu when editing in Author mode. Selecting this action opens the Push current element dialog box that allows you to select a target resource and element, and where to insert the current element content.
This dialog box allows you to configure the following options for the `conref push` action:

**Choose the target resource**

Allows you to select a **Location** URL or a **Key** for the target resource and the table in the next section of the dialog box will be populated using the information from the specified resource.

**Select the target element**

The table in this section contains the available elements (identified by their ID) that can be replaced by, or pushed before/after, the current element, according to the push action.

**Push action**

Allows you to choose one of the following options for where you want to insert the current element content:

- **replace the target element**
  
  The target element will be replaced with the current element content.
  
  On the technical side, the value of the `@conaction` attribute in the current element will be set to `push replace` and the `@conref` or `@conkeyref` attribute will be set to the specified reference.

- **push before**
The current element content will be inserted before the specified target element in the target resource.

On the technical side, the value of the @conaction attribute in the current element will be set to pushbefore. Another element with the same name and class as the target element will be inserted in the document after the current element. The new element will have the @conaction attribute set to mark and the @conref or @conkeyref attribute will be set to the specified reference.

**push after**

The current element content will be inserted after the specified target element in the target resource.

On the technical side, the value of the @conaction attribute in the current element will be set to pushafter. Another element with the same name and class as the target element will be inserted in the document before the current element. The new element will have the @conaction attribute set to mark and the @conref or @conkeyref attribute will be set to the specified reference.

You can also use the Preview panel to view the content that will be pushed and the Source panel to see the XML code for the content to be pushed. After you click OK, the conref push mechanism is inserted in the current document. The changes in the target resource will be processed when you transform the DITA map.

**Resources**

For more information about the conref push mechanism and other advanced DITA profiling concepts, watch our Webinar: Working with DITA in Oxygen - Advanced Profiling and Reuse Strategies.

**Related information**

The DITA Style Guide Best Practices for Authors: The Conref Push Technique

**Working with Reusable Components**

In DITA, the content of almost any element can be made reusable simply by adding an @id attribute to the element. The DITA content reference mechanism can reuse any element with an ID. However, it is not considered best practice to arbitrarily reuse pieces of text from random topics due to the difficulties this creates in trying to manage it. It also creates the possibility of authors deleting or changing content that is reused in other topics without being aware that the content is reused.

To prevent these types of problems, you can create reusable components to manage a separate set of topics that contain topics designed specifically for reuse. Then, all of your reusable content can be referenced from the reusable components and if the content needs to be updated you only need to edit it in one place.
Oxygen XML Editor allows you to select content in a topic, create a reusable component from it and reference that component in other locations by using the Create Reusable Component (on page 3055) and Insert Reusable Component (on page 3056) actions.

Related Information:
DITA Reusable Components View (on page 3061)

Creating a Reusable Content Component

Oxygen XML Editor makes it easy to create a reusable component from existing topic content.

Note:
To ensure that the topic file that contains the reusable component is a valid container for the reusable content component, Oxygen XML Editor attempts to use the same schema information in the current topic for the file that contains the reused component. If it cannot create a valid instance of the reused content file with this approach, the application creates a specialized topic type automatically. This specialization is designed to make sure that the content is compatible with the topic type that it is created from. You can make changes to the configuration file (\OXYGEN_INSTALL_DIR\frameworks\dita\reuse\reuse_configuration.properties) or override it from a framework extension to control if the created reusable component file should be based on the currently edited topic or it should be an automatically created DITA specialization topic.

Follow these steps to create a reusable component:

1. In Author mode, select the content you want to make into a reusable component (or place the cursor inside an element you want to reuse).
2. Select the Create Reusable Component action that is available in the DITA menu or the Reuse submenu of the contextual menu.
   The Create Reusable Component dialog box is displayed.
3. Use the Reuse Content drop-down list to select the scope of the content to be made reusable. It allows you to select how much of the current content you want to make reusable. The choices presented include the element at the current cursor position and its ancestor elements.
4. Add a description. This is used as a description for the reusable component, but is not part of the reused content. It is just to help you identify the reusable content and will not become part of your output.
5. If the Replace selection with content reference option is selected, the selection in the current topic will be replaced with a content reference (@conref) that points to the new reusable component. If multiple elements are selected (for example, multiple steps or list items), the selection is replaced with a content reference range (@conref and @conrefend).
6. Select a file name and location to save the topic containing the reusable component and click Save. It is considered best practice to save or store reusable components in an area set aside for that purpose.
A file is created that contains one reusable component. You can reference the reused content in other topics by using a content reference (on page 3036) or content key reference (on page 3038). Also, if the Replace selection with content reference option was selected, Oxygen XML Editor replaces the selected content with a content reference that will be displayed in your current topic with a gray background and it can only be edited in the source file (the new reusable component). To edit the source file, click the Edit Content icon at the beginning of the content reference.

Inserting a Reusable Content Component

Oxygen XML Editor includes an Insert Reusable Content action that allows you to easily insert a reusable content component that you created using the Create Reusable Component action (on page 3055).

**CAUTION:**
This action is only designed to insert reusable components created using the Oxygen XML Editor Create Reusable Component action. It assumes certain things about the structure of the reusable content file that may not be true of reusable content created by other methods and it may not provide the expected results if used with content that does not have the same structure.

The Insert Reusable Content action creates a DITA @conref to insert the content, and creates a parent element for the @conref attribute based on the type of the reusable element in the reusable component file. This action ensures that the correct element is used to create the @conref. However, that element must still be inserted at a point in the current topic where that element type is permitted.

To insert a reusable component that was created using the Create Reusable Component action, follow these steps:

1. Place the cursor at the insertion point where you want the reusable component to be inserted.
2. Select the Insert Reusable Component action that is available in the DITA menu or the Reuse submenu of the contextual menu.
   The Insert Reusable Component dialog box is displayed.
3. Locate the reusable content file that you want to insert its content.
4. If you select Content reference in the Insert as drop-down list, the action will add a @conref attribute to the DITA element at the current location. If you select Copy in the drop-down list, the content of the reusable component file will simply be pasted at the current location (assuming the content is valid at the current location).
5. Click Insert to perform the action.

Working with Variable Text in DITA

You may often find that you want a certain piece of text in a topic to have a different value in various circumstances. For example, if you are reusing a topic about a feature that is shared between several products, you might want to make the name of the product a variable so that the correct product name is used in the manual for each product.
For example, you might have a sentence like this:

The quick-heat feature allows [product-name] to come up to temperature quickly.

You need a way to substitute the correct product name for each product.

One way to do this would be to use conditional profiling to provide conditional values using the `@product` profiling attribute, as in the following example:

```xml
<p>The quick-heat feature allows
<ph product="basic">Basic Widget</ph>
<ph product="pro">Pro Widget</ph>
to come up to temperature quickly.</p>
```

However, this approach means that you are repeating the product names over and over again everywhere the product name is mentioned. This is time consuming for authors and will create a maintenance problem if the product names change.

The alternative is to use a key reference, as in the following example:

```xml
<p>The quick-heat feature allows <ph keyref="product"/>
to come up to temperature quickly.</p>
```

The definition of the key reference determines the name of the product:

```xml
<keydef keys="product" product="basic">
  <topicmeta>
    <keywords>
      <keyword>Basic Widget</keyword>
    </keywords>
  </topicmeta>
</keydef>
<keydef keys="product" product="pro">
  <topicmeta>
    <keywords>
      <keyword>Pro Widget</keyword>
    </keywords>
  </topicmeta>
</keydef>
```

When the content is published, the value defined in the `product` key will be inserted for each product.

**Inserting a Keyref**

To insert a defined key reference (on page 2930) into a document in Oxygen XML Editor Author mode, use one of the following methods (the method you choose simply depends on which Oxygen XML Editor feature you prefer):
• **DITA Reusable Components View Method**

Use the [DITA Reusable Components view](on page 3061) to insert a-variable reference to the [defined key](on page 2930). For example, in the Keys tab, find a key defined as a variable and double-click it. Oxygen XML Editor will insert the variable as a `<ph>` element with a `@keyref` attribute that references the specified key.

• **Code Template Method**

Add the source code pattern of the-[defined key](on page 2930) to a [code template](on page 217) so that it appears in the list of proposals in the [Content Completion Assistant](on page 3217). For example, the code pattern could be something like `<ph keyref="product">` for defined product key.

• **Reuse Content Dialog Box Method**

Use the [Reuse Content](on page 3043) action on the main toolbar to open the [Reuse Content](on page 3043) dialog box. Use the Key option to select a key that is defined as a variable (key reference to metadata) and Oxygen XML Editor will insert the variable as a `<ph>` element with a `@keyref` attribute that references the specified key.

• **Manual Method**

Manually insert the `@keyref` attribute using the attributes editor as follows:

1. Press Enter and select a DITA element (for example, `<ph>`) that supports the `@keyref` attribute.
2. Select [Edit Attributes](on page 630) from the contextual menu (or simply press Alt+Enter) to bring up the attributes editor.
3. In the Name field, select keyref.
4. In the Value field, select or enter the name of the [defined key](on page 2930).

**Related Information:**

[DITA Reusable Components View](on page 3061)
[Defining Keys in DITA Maps](on page 2930)

**Working with DITA 1.3 Key Scopes**

DITA 1.3 includes the possibility of using a concept called Key Scopes (or scoped keys). It allows you to reuse a topic in multiple places within the same DITA map, but with slightly different content in each instance.

**Key Scopes Use-Case**

Suppose that you develop a software product and you have a topic in your user guide that explains how to install your product on a Windows operating system. Suppose that the steps are exactly the same for installing it on Linux and the only difference is the name of the operating system. Therefore, it would be helpful if you could reuse the exact same content in two different topics, but with the name of the operating
system different in each instance. In DITA 1.2, this is not possible since keys can only be resolved to a single value. However, with the DITA 1.3 Key Scopes mechanism, you can define multiple values for the same key depending on the context.

**How to Use Key Scopes in Oxygen XML Editor**

To use DITA 1.3 key scopes in Oxygen XML Editor, follow these steps:

1. Define the keys *(on page 3026)* to be used in multiple places within your DITA map.

2. For each particular topic that contains the keys, define the key scopes:
   a. Right-click the topic in the DITA Maps Manager *(on page 2897)* and select **Edit properties**.
   b. In the Keys tab *(on page 2934)*, enter a value (or multiple values) in the Key scopes field.
   c. Click **OK** to save your changes.

3. Save the DITA map.

   **Result:** In the DITA Maps Manager *(on page 2897)*, you can now see the key scopes in brackets and when you open each topic reference.

   ![Figure 778. Key Scopes in DITA Maps Manager](image)

   The content will also be expanded in Author mode according to the context of the key scope you defined for that particular topic. Also, when you transform the DITA map, the scoped keys will be reflected in the published content.

**Resources**

- You can find a more detailed example and download samples for reuse possibilities based on key scopes in the DITA 1.3 Key Scopes - Next Generation of Reuse blog post.
- You can also watch our DITA 1.3 video tutorial to see how key scopes can be used in Oxygen XML Editor.
- For more information about key scopes and other advanced DITA reuse concepts, watch our Webinar: Working with DITA in Oxygen - Advanced Profiling and Reuse Strategies.

**Related information**

- Working with DITA 1.3 Branch Filtering *(on page 3060)*
- Oxygen XML Blog: DITA 1.3 Key Scopes - Next Generation of Reuse
- Oxygen Video Tutorial: DITA 1.3 (Key Scopes, Branch Filtering)
Working with DITA 1.3 Branch Filtering

DITA 1.3 allows you to use a mechanism called *Branch Filtering* that enables you to set filtering conditions for specific branches of a *DITA map* (*on page 3218*). This makes it possible for multiple conditional profiles to be applied within a single publication, each time with a different filter.

**Branch Filtering Use-Case**

Suppose that you sell two models of a mobile phone and you need to create a brochure for each model. You want both brochures to have the same structure and most of the content is the same for both brochures. The only differences are in the values for certain details (for example, the model name, size dimensions, battery life, etc.) Therefore, it would be helpful if you could use the same topic and reference it twice in the same map, with each reference using different filtering conditions. In DITA 1.2, this is not possible since you can only apply one DITAVAL filter to a map. However, with the DITA 1.3 *Branch Filtering* mechanism, you can reuse content multiple times within the same map, each time using different filters.

**How to Use Branch Filtering in Oxygen XML Editor**

To use DITA 1.3 branch filtering in Oxygen XML Editor, follow these steps:

1. The support for DITA 1.3 must be enabled in the *DITA preferences page* (*on page 269*).
2. Assuming you have already defined your profiling attributes (*on page 670*), create a DITAVAL filter file (*on page 3157*).
3. Insert a reference to the DITAVAL filter file in the *DITA map*:
   a. Right-click the *DITA map* reference in the *DITA Maps Manager* (*on page 2897*) and select *Append Child > DITAVAL Reference*.
   b. Select the DITAVAL file.
   c. Click *Insert and Close*.
4. Save the *DITA map*.

**Result:** You can now see the ditaval files referenced in the *DITA Maps Manager* (*on page 2897*) and when you transform the *DITA map*, filtered content will be reflected in the published output.

Figure 779. Branch Filtering in DITA Maps Manager

![Diagram of DITA Maps Manager showing Syncro Mobile Phones, X1000, X2000 models, and their respective ditaval files](image)
Resources

- You can find a more detailed example and download samples for reuse possibilities based on key scopes in the DITA 1.3 Branch Filtering - Next Generation of Reuse blog post.
- You can also watch our DITA 1.3 video tutorial to see how branch filtering can be used in Oxygen XML Editor.
- For more information about branch filtering and other advanced DITA reuse concepts, watch our Webinar: Working with DITA in Oxygen - Advanced Profiling and Reuse Strategies.

Related information
- Working with DITA 1.3 Key Scopes (on page 3058)
- Oxygen XML Blog: DITA 1.3 Branch Filtering - Next Generation of Reuse
- Oxygen Video Tutorial: DITA 1.3 (Key Scopes, Branch Filtering)

DITA Reusable Components View

The DITA Reusable Components view is helpful if you use a large number of keys or reusable components in your DITA project. It collects all of the keys and reusable components that are defined in the root map (on page 3223) and presents them in a side view where you can easily locate and insert references to them. It recollects the keys anytime the root map is changed (on page 2914) or you switch the editor focus to a different file.

If the view is not displayed, it can be opened by selecting it from Window > Show View. By default, it appears in the bottom-right section of the editor.

Tip:
You can also assign a keyboard shortcut to open the view using the Menu Shortcut Keys preference page (on page 294).

It includes the following tabs:

- **Keys (on page 3062)** - Displays all the keys that are defined in the root map (on page 3223) and provides ways to easily insert references to them as cross reference links, key references, or variables. It includes a search field, some filtering and sorting options to help you find particular keys, and some contextual menu actions. It also supports drag and drop actions and double-clicking a key is the fastest way to insert a reference.

- **Components (on page 3065)** - Displays all the reusable components found in the root map (on page 3223) and provides ways to easily insert them as content references or content key references. To determine which components to display in this tab, Oxygen XML Editor looks for any topicref in the root map (on page 3223) that is marked as resource-only and then looks for elements with an assigned @id attribute value. This tab includes a search field, some filtering options, and some simple links and...
contextual menu actions to quickly insert references or open their source file. It also supports drag and drop and double-clicking actions.

- **Media (on page 3068)** - Displays all images referenced as keys in the root map along with all images found in the user-defined working sets.

### Keys Tab

The **DITA Reusable Components** view collects all the `keys` that are defined in the current root map (**on page 3223**) and displays them in the **Keys** tab. This tab has two view modes. The default **tiles** style view mode and a **table** style view mode.

#### Tiles Mode

The default **tiles** mode displays the keys as blocks (cards). The advantage of this display mode is that more information about each particular key can be seen even when the view is sized with a small width. Each block (card) displays the name of the key (the value of the `@keys` attribute), followed by its description and/or `@href` value, then followed by the name of the DITA map file where the key is defined.

![DITA Reusable Components View - Keys Tab (Default tiles mode)](image)

#### Table Mode

You can switch to a **table** style display mode by selecting the **Table mode** toggle action from the **Settings** menu. The advantage of this display mode is that more keys can be listed at once. In this mode, keys that are defined with a text value in the `<navtitle>` or `<keyword>` element have that value listed in the **Description** column, while keys that are defined with a value in an `@href` attribute have that href value listed in the **Href** column.
Both display modes in the **Keys** tab include a variety of features and options:

**Search Filter**

You can enter text in the filter field at the top of this tab to filter the list and search for specific keys.

**Sorting**

**Tiles Mode:** In the default *tiles* display mode, to sort the keys alphabetically in ascending order, select **Sort by key name** from the 🌐, **Settings** menu.

**Table Mode:** In the *table* display mode, the following columns can be sorted by clicking the heading:

- **Key** - The name of the key (the value of the `@keys` attribute).
- **Description** - The description of the key that is obtained from its definition. Keys that are defined with a text value in the `<navtitle>` or `<keyword>` element have that value listed in this column.
- **Href** - Keys that are defined with a value in an `@href` attribute have that href value listed in this column.
- **Definition Location** - The name of the **DITA map** *(on page 3218)* where the key is defined.

**Double-Click Mechanism**

You can double-click any key listed in this tab to insert a key reference at the current cursor position or surrounding the current selection.

- If the selected key points to an `@href` value, it is inserted as a **cross reference link** *(xref)* *(on page 3072)*.
- If the selected key is a reference to an image, it is inserted as an `<image>` element.
- If the selected key does not have an associated `@href`, it is inserted as a **variable reference** *(ph)* *(on page 3056)*.

**Drag and Drop Mechanism**
You can drag a key from this tab and drop it in the main editor to insert a key reference at the current cursor position.

- If the selected key points to an @href value, it is inserted as a cross reference link (xref) (on page 3072).
- If the selected key is a reference to an image, it is inserted as an <image> element.
- If the selected key does not have an associated @href, it is inserted as a variable reference (ph) (on page 3056).

**Contextual Menu Actions**

**Insert as Link**

Inserts a cross reference link (xref) (on page 3072) to the selected key at the current cursor position or surrounding the current selection.

**Insert as Variable**

Inserts a variable reference (ph) (on page 3056) to the selected key at the current cursor position or surrounding the current selection. However, if the selected key is a reference to an image, this action inserts the key reference in an <image> element.

**Insert as Keyref**

Presents a submenu with all the elements that can be inserted at the current cursor position. Selecting an element will insert that element at the current cursor position or surrounding the current selection with a @keyref attribute and its value set to the selected key.

**Rename Key**

Opens a refactoring wizard (on page 844) where you can easily rename the key and define the scope of the operation. It also updates all references to it.

**Notes:**

- This action does not work on DITA 1.3 key scopes.
- This action is only available if the DITA map opened in the DITA Maps Manager is also selected as the Root map.

**Go to Definition**

Opens the DITA map where the key is defined.

**Search References**

Searches for all references to the selected key in the entire DITA map structure.

**Group by Definition Location (Available in Table mode only)**
A toggle action that can be used to group (and sort) all the keys based on the value in the **Definition Location** column.

**Settings Menu**

This menu includes the following options:

**Filtering Options**

- **Show all** - Shows all defined keys found in the current root map *(on page 3223)*.
- **Show only variables** - Filters the keys to show only those defined as variable references *(on page 3056)*.
- **Show only maps and topics** - Filters the keys to show only those that reference DITA maps or topics.
- **Show only multimedia resources** - Filters the keys to show only those that reference multimedia resources (such as images).
- **Show only external resources** - Filters the keys to show only those that reference external resources (such as web links).

**Sort by key name (Available in Tiles mode only)**

Sorts the keys alphabetically in ascending order.

**Table mode**

A toggle action that switches between the *table* and *tiles* display modes.

**Components Tab**

The **DITA Reusable Components** view collects all the topics from the current root map *(on page 3223)* that are marked as *resource-only*, then collects the reusable components from those topics, and displays them in the **Components** tab. To set a topic as *resource-only*, the `<topicref>` in the DITA map file must have a `@processing-role` attribute set like this:

```xml
<topicref href="topics/randomize-xml-content.dita" processing-role="resource-only"/>
```

The **DITA Reusable Components** view considers topic references that contain `processing-role="resource-only"` to be candidates to contain reusable components. The reusable components inside these topics are collected from all elements that have an *ID* specified. These reusable components are displayed in the **Components** tab along with the file name and the specific names of the elements that contain an *ID* attribute.
The **Components** tab includes the following features and options:

**Search Filter**

You can enter text in the filter field at the top of this tab to filter the list and search for specific content inside the list of reusable components. This field supports many of the Lucene-based search patterns, such as wildcards (* or ?), boolean operators (AND, OR, NOT), fuzzy searches (~), boosting searches (^), and more.

**Settings Menu**

This menu includes the following options:

**Compact Mode**

You can use this toggle action to switch the display for the **Components** tab to a compact visualization mode. When switched to **Compact mode**, fewer details are shown for each component, but more components are displayed in the view.

**Reindex**

You can use this action force a re-indexing of the reusable components.

**Show Elements of Type**

You can use this drop-down list to select specific types of elements to be displayed in the list of components. This can help you narrow down the list of possible source elements that you can select.

**Source File(s)**
You can use this combo box to search for specific source files (the topics that contain reusable components) or select a file from its drop-down list. You can also use wildcards (such as * or ?) in this field.

**Double-Click Mechanism**

You can double-click any reusable component listed in preview window in this tab to insert it as a content reference or content key reference at the current cursor position or replace the current selection.

- If the parent topic of the selected component has a key defined, it is inserted as a content key reference (`conkeyref`) *(on page 3038)*.
- If the parent topic of the selected component does not have a key defined, it is inserted as a content reference (`conref`) *(on page 3036)*.

**Drag and Drop Mechanism**

You can drag a reusable component from the preview window in this tab and drop it in the main editor to insert a content reference or content key reference at the current cursor position.

- If the parent topic of the selected component has a key defined, it is inserted as a content key reference (`conkeyref`) *(on page 3038)*.
- If the parent topic of the selected component does not have a key defined, it is inserted as a content reference (`conref`) *(on page 3036)*.

**Hover and Click Actions**

If you hover over a component shown in the preview window, you have access to the following link actions:

- **Insert**
  
  Inserts the component as a content reference or content key reference at the current cursor position or replaces the current selection. If the parent topic has a key defined, it is inserted as a content key reference (`conkeyref`) *(on page 3038)*. Otherwise, it is inserted as a content reference (`conref`) *(on page 3036)*.

- **Open**
  
  Opens the source file that contains the reusable component.

**Contextual Menu Actions**

- **Insert Content Reference**
  
  Inserts the component as a content reference (`conref`) *(on page 3036)* at the current cursor position or replaces the current selection.

- **Insert Content Key Reference**
Inserts the component as a **content key reference** *(conkeyref)* *(on page 3038)* at the current cursor position or replaces the current selection. This action is only available if the parent topic has a key defined.

**Go to Definition**

Opens the source file that contains the reusable component.

**Search References**

Searches for all references to the selected component in the entire **DITA map** *(on page 3218)* structure.

**Media Tab**

The **Media** tab displays all media resources (images, audio, video) referenced as keys in the current root map along with all audio, image, and video resources found in **user-defined working sets** *(on page 3069)*.

**Figure 783. DITA Reusable Components View - Media Tab**

The **Media** tab includes the following features and options:

**Search Filter**

You can enter text in the filter field at the top of this tab to filter the list and search for specific media resource key or file names in the list of available resources.

**Resource Type Filter**

You can quickly show resources of a specific type by clicking one of the type buttons (**Images**, **Audio**, **Video**, **Others**).
Settings Menu

This menu includes the following options:

Sort by date

Sorts the presented resources based on both the date when they were last modified and the date they were created.

Configure working sets

Use this option to define folders where the media resources will be gathered. The Media tab will include audio, image, and video resources collected from the current root map as well as media resources located in the folders defined as active working sets. The working sets are defined at project level so that they can be easily shared with others. To save working sets in the global user-specific settings instead, in the Preferences > Project Level Settings page you can uncheck the Save DITA media working sets at project level checkbox.

Reload

Refreshes the list of displayed media resources. This is useful if resources were recently added in the searched folders.

Double-Click Mechanism

You can double-click any media resource in the list to add a reference to it.

Drag and Drop Mechanism

You can drag a media resource from the list and drop it in the main editor to add a reference to it.

Contextual Menu Actions

Insert As Image Reference

Inserts an image reference. If the resource is referenced using a key in the DITA map, an indirect reference using the defined key will be used. Otherwise, the reference will point directly to the resource location.

Insert As Audio Reference

Inserts an audio reference. If the resource is referenced using a key in the DITA map, an indirect reference using the defined key will be used. Otherwise, the reference will point directly to the resource location.

Insert As Video Reference

Inserts a video reference. If the resource is referenced using a key in the DITA map, an indirect reference using the defined key will be used. Otherwise, the reference will point directly to the resource location.

Insert As Embedded Reference
Inserts as an embedded reference. If the resource is referenced using a key in the DITA map, an indirect reference using the defined key will be used. Otherwise, the reference will point directly to the resource location.

**Insert as Link**

Inserts a link to the resource, either as a DITA `<xref>` or `<link>`, depending on the cursor position.

**Insert as Variable**

Inserts a variable reference (pH) (on page 3056) to the selected key at the current cursor position or surrounding the current selection. However, if the selected key is a reference to an image, this action inserts the key reference in an `<image>` element.

**Insert as Keyref**

Presents a submenu with all the elements that can be inserted at the current cursor position. Selecting an element will insert that element at the current cursor position or surrounding the current selection with a `@keyref` attribute and its value set to the selected key.

**Preview**

Shows the selection in an Image Preview side view.

**Open in System Application**

Opens the default system editor/viewer associated with the resource type.

**Show in Explorer/Finder**

Opens the default file browser at the specific folder where the resource is located.

Items in the Media tab are presented in the following order:

- Key definitions are always presented first, in document order.
- Resources defined and collected from working sets are sorted alphabetically by name, for each folder separately.

**Related Information:**

Working with Reusable Components (on page 3054)
Linking in DITA Topics (on page 3072)
Working with Variable Text in DITA (on page 3056)
Working with Keys in DITA (on page 3026)
Creating a DITA Content Reference (on page 3036)
Creating a DITA Content Key Reference (on page 3038)
Working with Content References (on page 3035)
Linking in DITA

DITA provides support for various types of linking between topics, some of which is automated, while others are specified by the author. Oxygen XML Editor provides support for all forms of linking in DITA.

Linking Between Parent, Child, and Sibling Topics

A DITA map (on page 3218) creates a hierarchical relationship between topics. That relationship map expresses a narrative flow from one topic to another, or it may be used as a classification system to help the reader find topics based on their classification, without creating a narrative flow. Since there may be various types of relationships between topics in a hierarchy, you may want to create links between topics in a variety of ways. For instance, if your topics are supposed to be organized into a narrative flow, you may want to have links to the next and previous topics in that flow. If your topics are part of a hierarchical classification, you may want links from parent to child topics, and vice versa, but not to the next and previous topics.

Parent, child, and sibling links are created automatically by the DITA output transformations (and may differ between various output formats). The kinds of links that are created are determined by the DITA collection-type attribute (on page 2936).

In-Line Linking in the Content of a Topic

DITA supports linking within the text of a topic using the `<xref>` element. The destination of the link can be expressed directly using the `@href` attribute or indirectly using the `@keyref` attribute. If you use the `@keyref` attribute, you link to a key rather than directly to a topic. That key is then assigned to a topic in a map that includes that topic. This means that you can change the destination that a key points to by editing the key definition in the map or by substituting another map in the build.

Linking Between Related Topics

In addition to the relationships between topics that expressed by their place in the hierarchy of a map, a topic may be related to other topics in various ways. For instance, a task topic may be related to a concept topic that gives the background of the task, or to a reference topic that provides data needed to complete the task. Task topics may also be related to other tasks in a related area, or concepts to related concepts.

Typically, they are grouped in a list at the end of the topic, although this depends on the behavior of the output transformation. DITA provides two mechanisms for expressing relationships between topics at the topic level: the Related Links section of a topic and relationship tables in maps. To add related links, select Related Link to Topic, Related Link to File, or Related Link to Web Page from the ‒ Link drop-down menu from the toolbar (or the ‒ Link submenu in the contextual menu or DITA menu).

Tip:

You can use the Find Similar Topics action (available in the contextual menu or DITA menu) to quickly find related topics that can be added as related links. It opens the Open/Find Resource view and performs a search using text content from the `<title>`, `<shortdesc>`, `<keyword>`, and `<indexterm>` elements.
Managing Links

Links can break for a variety of reasons. The topic that a link points to may be renamed or removed. A topic may be used in a map that does not include a linked topic. A topic or a key may not exist in a map when a particular profile is applied. The DITA Maps Manager (on page 2897) provides a way to validate all the links in the documents that are included in the map (on page 2941). This can include validating all the profiling conditions that are applied.

Hierarchical Linking in DITA Maps

To create hierarchical linking between the topics in a DITA map (on page 3218), you set the appropriate value of the `@collection-type` attribute on the map. See the DITA documentation for the meaning of each of the values of the `@collection-type` attribute.

**Note:**

Publishing scripts determine when and how to create hierarchical links. The `@collection-type` attribute does not force a particular style of linking. Instead, it declares what the nature of the relationship is between the topics. The publishing scripts use that information to determine how to link topics. Scripts for different types of media might make the determination depending on what is appropriate for the particular type of media. You can provide additional instructions to the scripts using the `@linking` attribute.

To add the `@collection-type` to an item in a map:

1. Right-click the topic and choose **Edit Properties**. The **Edit Properties** dialog box is displayed.
2. In the Attributes tab, select the appropriate value from the Collection type drop-down list.
3. You can use the Other attributes table to add a value to the `@linking` attribute.

Linking in DITA Topics

Direct Links

Inline links can be created DITA topics using the `<xref>` element. The destination of the link can be expressed directly by using the `@href` attribute and the target can be another topic or a specific element within the other topic, another location within the same topic, a file, or a web link. You can also create direct related links to topics, files, or websites in a DITA topic using the `<related-links>` element.

Indirect Links Using Keys

The destination of the link can also be expressed indirectly by using keys (on page 3026) to create either inline links or related links (with the `@keyref` attribute). By using keys, you avoid creating a direct dependency between topics. This makes links easier to manage and can make it easier to reuse topics in various publications. It can also be helpful in verifying the completeness of a publication, by ensuring that a publication map provides a key definition for every key reference used in the content.
Links based on keys require two pieces:

- **Key Definition** - Assigns a key to a topic so that other topics can link to it. For more information, see Defining Keys in DITA Maps (on page 2930).
- **Key Reference** - Created in an `<xref>` element and specifies the key to link to.

The key reference points to a key definition, and the key definition points to a topic. Key definitions are created in maps, as an element on the `<topicref>` element that points to a topic. This allows you to assign a particular key to one topic in one map and to another topic in another map. When a topic that links to that key is used in each of these maps, the links work correctly in both maps.

**Inserting a Link in Oxygen XML Editor**

To insert a link in **Author mode (on page 354)**, use the actions available in the Link drop-down menu from the toolbar (or the Link submenu in the contextual menu or DITA menu). You can choose between the following types of inline links:

**Cross Reference**

Opens the Cross Reference (xref) dialog box (on page 3075) that allows you to insert a cross reference link to a target DITA resource at the current location within a document. The target resource can be the location of a file or a key that is already defined in your DITA map structure. Once the target resource has been selected, you can also target specific elements within that resource. Depending on the context where it is invoked, the action inserts one of the following two elements:

- `<xref>` - Used to link to other topics or another location within the same topic and points to the target using the `@href` or `@keyref` attribute.
- `<fragref>` - A logical reference to a fragment element within a syntax diagram and points to the target using the `@href` or `@keyref` attribute.

**File Reference**

Opens a dialog box that allows you to insert a link to a target non-DITA file resource at the current location within a document. The target resource can be the location of a file or a key that is already defined in your DITA map structure. It inserts an `<xref>` element with either an `@href` attribute or a `@keyref` attribute. If you select Location for the target, the link is expressed in an `@href` attribute. If you select Key for the target, keys will be used to express the link in a `@keyref` attribute. You can select a key from the drop-down list or click the Choose Key Reference button to use the Choose Key dialog box (on page 3075).

**Web Link**

Opens a dialog box that allows you to insert a link to a target web-related resource at the current location within a document. The target resource can be a URL or a key that is already defined in your DITA map structure. It inserts an `<xref>` element with either an `@href` attribute or a `@keyref` attribute. If you select URL for the target resource, the link is expressed in an `@href` attribute. If
you select Key for the target, keys will be used to express the link in a @keyref attribute. You can select a key from the drop-down list or click the Choose Key Reference button to use the Choose Key dialog box (on page 3075).

**Related Link to Topic**

Opens the Cross Reference (xref) dialog box (on page 3075) that allows you to insert a link to a target DITA resource in a related links section that is typically at the bottom of your topic (although this depends on the behavior of the output transformation). The target resource can be the location of a file or a key that is already defined in your DITA map structure. Once the target resource has been selected, you can also target specific elements within that resource. If a related links section does not already exist, this action creates one. Specifically, it inserts a `<link>` element inside a related-links element.

**Tip:**

You can use the Find Similar Topics action (available in the contextual menu or DITA menu) to quickly find related topics that can be added as related links. It opens the Open/Find Resource view and performs a search using text content from the `<title>`, `<shortdesc>`, `<keyword>`, and `<indexterm>` elements.

**Related Link to File**

Opens a dialog box that allows you to insert a link to a target non-DITA file resource in a related links section that is typically at the bottom of your topic (although this depends on the behavior of the output transformation). The target resource can be the location of a file or a key that is already defined in your DITA map structure. If a related links section does not already exist, this action creates one. Specifically, it inserts a `<link>` element inside a related-links element. If you select Location for the target, the link is expressed in an @href attribute. If you select Key for the target, keys will be used to express the link in a @keyref attribute. You can select a key from the drop-down list or click the Choose Key Reference button to use the Choose Key dialog box (on page 3075).

**Related Link to Web Page**

Opens the Web Link dialog box that allows you to insert a link to a target web-related resource in a related links section that is typically at the bottom of your topic (although this depends on the behavior of the output transformation). The target resource can be a URL or a key that is already defined in your DITA map structure. If a related links section does not already exist, this action creates one. Specifically, it inserts a `<link>` element inside a related-links element. If you select URL for the target resource, the link is expressed in an @href attribute. If you select Key for the target, keys will be used to express the link in a @keyref attribute. You can select a key from the drop-down list or click the Choose Key Reference button to use the Choose Key dialog box (on page 3075).
Cross Reference (xref) Dialog Box

The Cross Reference (xref) dialog box is displayed when you insert a Cross Reference or Related Link to Topic (from the \Link drop-down menu). It allows you to insert a link to a target resource at the current location within a document (for a Cross Reference link) or in a related links section (for a Related Link to Topic). The target resource can be the location of a file or a key that is already defined in your DITA map structure. Once the target resource has been selected, you can also target specific elements within that resource.

Figure 784. Cross Reference (xref) Dialog Box

This dialog box includes the following sections and fields:

Choose the Target Resource Section

Location

If you select Location for the target, the link is expressed in an @href attribute.

Key

If you select Key for the target, keys will be used to express the link in a @keyref attribute. You can use the \Choose Key Reference button to open the Choose
**Key** dialog box that allows you to select one from a list of all the keys that are gathered from the root map (on page 3223) (you can also select one from the drop-down list in the **Key** field).

**Tip:**
You can also use the DITA Reusable Components view (on page 3061) for similar purposes.

![Choose Key Dialog Box](image)

The **Choose Key** dialog box includes the following:

- **Change Root Map** - Opens a small dialog box that allows you to select a root map (on page 2914).

- **Search Filter** - You can enter text in the filter field at the top of the dialog box to filter the list and search for specific keys.

- **Sortable Columns** - The dialog box includes the following columns that can be sorted by clicking on the heading:
  - **Key** - The name of the key (the value of the @keys attribute).
  - **Description** - The description of the key that is obtained from its definition. Keys that are defined with a text value in the `<navtitle>` or `<keywords>` element have that value listed in this column.
  - **Href** - Keys that are defined with a value in an @href attribute have that href value listed in this column.
  - **Definition Location** - The name of the DITA map (on page 3218) where the key is defined.

- **Group by Definition Location** - A contextual menu action that can be used to group (and sort) all the keys based upon the value in the Definition Location column.

**Select the Target Element Section**

This section can be used to target a specific element inside the target resource.
Show elements of type
You can use this drop-down list to select specific types of elements to be displayed in the subsequent table. This can help you narrow down the list of possible source elements that you can select.

Text Filter Field
You can also use the text filter field to narrow down the list of possible source elements to be displayed in the subsequent table.

Element Table
Presents all the element IDs defined in the source topic. Use this table to select the Target ID of the element that you want to reference.

Preview Pane
Displays the content that will be referenced.

Source Pane
 Displays the XML source code of the element to be referenced.

Once you click Insert or Insert and close, the configured cross reference is inserted into your document.

Tip:
You can easily insert multiple cross references by keeping the dialog box opened, using the Insert button.

Using Copy/Paste or Drag/Drop Actions to Insert a Cross Reference
Oxygen XML Editor also includes support for inserting cross reference links with simple copy/paste or drag/drop actions (additionally, you can insert them using the Paste as Link or Paste as Link (keyref) actions found in the Paste Special submenu from the contextual menu). The copied/dragged content must be an entire DITA XML element with an @id attribute or a <topicref>. Also, the location in the document where you paste or drop the link must be valid, although as long as the Smart paste and drag and drop option (on page 181) is selected in the Schema-Aware preferences page, if you try to paste it in an invalid location, Oxygen XML Editor will attempt to place it in a valid location, and may prompt you with one or more choices for where to place it.

When the link is inserted, Oxygen XML Editor automatically tries to populate certain attributes based on detected values. The @format, @scope, and @type attributes are populated if their corresponding options are selected in the Inserting Links section of the DITA Topics preferences page (on page 274). Even if their corresponding options are not selected, the @format and @scope attributes are populated if their detected values are different than the default values.

Note:
For the sake of performance, the @type attribute is never automatically computed in the following cases:
Typically, cross reference links are inserted with an `@href` attribute, but it is also possible to insert them with a `@keyref` attribute using the Paste as Link (keyref) contextual menu action or copy/paste or drag/drop actions. For the latter method, follow these steps:

1. In the DITA Maps Manager view (on page 2897), make sure that the Context combo box (on page 2901) points to the correct map that stores the keys.
2. Make sure the topic that contains the content you want to reference has a key assigned to it. To assign a key, right-click the topic with its parent map opened in the DITA Maps Manager (on page 2897), select Edit Properties, and enter a value in the Keys field.
3. Copy an entire DITA element that has an ID attribute assigned to it from a topic with an assigned key, or a `<topicref>` from a DITA map.
4. Place the cursor at a location, where you want to insert the link.
5. Select the Paste as Link (keyref) action from the Paste Special submenu from the contextual menu.

Related Information:
- Defining Keys in DITA Maps (on page 2930)
- DITA Reusable Components View (on page 3061)

## Linking with Relationship Tables in DITA

A relationship table is used to express relationships between topics outside of the topics themselves. The DITA publishing scripts can then create links between related topics when the content is published.

The reason for using a relationship table is to help make topics easier to reuse. If a topic links directly to another topic, this creates a dependency between the topics. If one topic is reused in a publication where the other is not used, the link is broken. By defining relationships between topics in a relationship table, you avoid creating this dependency.

To create an appropriate set of links between topics in multiple publications, you can create a separate relationship table for each publication. If you are creating multiple publications by applying profiling conditions to a single map, you can also profile your relationship table.

To create a relationship table, follow these steps:
1. If the map is currently open in the DITA Maps Manager (on page 2897), double-click the map icon (ifiant emoji) to open the map in Author mode. If it opens in Text mode, click Author at the bottom left to switch to Author mode.

2. Move the insertion point inside the map root element (usually `<map>`, but it might be `<bookmap>`, or another specialization of the `<map>` element). The easiest way to do this is to click below the title of the map in the editor and then press the up arrow once. Confirm that you are inside the map root element by checking the breadcrumbs at the top left of the editor window. You should only see the name of the map root element.

3. Select the Insert Relationship Table action on the toolbar or from the Relationship Table submenu of the contextual menu. The Insert Relationship Table dialog box is displayed.

4. Set the number of rows, the number of columns, a table title (optional), and select whether or not you want a table header. Click Insert.

5. Enter the type of the topics in the header of each column. The header of the table (the `<relheader>` element) already contains a `<relcolspec>` element for each table column. You should set the value of the `@type` attribute of each `<relcolspec>` element to a value such as `concept`, `task`, or `reference`. When you click in the header cell of a column (that is a `<relcolspec>` element), you can see all the attributes of that `<relcolspec>` element, including the `@type` attribute in the Attributes view (on page 627). You can edit the attribute type in this view.

6. To insert a topic reference in a cell, place the cursor in a table cell and select Insert Reference (on page 2961) from the contextual menu or the DITA Map toolbar.

7. To add a new row to the table or remove an existing row use Insert Relationship Row/Delete Relationship Row from the contextual menu or the DITA Map toolbar.

8. To add a new column to the table or remove an existing column, use Insert Relationship Column/Delete Relationship Column contextual menu or the DITA Map toolbar. If you double-click the relationship table (or select it and press Enter, or choose Open from the contextual menu) the DITA map is opened in the editor with the cursor positioned inside the corresponding relationship table.

9. To add topic references to your relationship table, drag and drop topics from the DITA Maps Manager (on page 2897) or the Project (on page 402) view into the appropriate cell in the relationship table. See the DITA documentation for a full explanation of the relationship table format and its options. Note that you can change all the selections that you make here later by using the actions on the toolbar (or in the Relationship Table submenu of the contextual menu) or by editing the underlying XML in Text mode.

10. Save the DITA map.

Relationship tables are also displayed in the DITA Maps Manager view (on page 2897), along with the other elements in its DITA map.
You can open the DITA map to edit the relationship table by doing one of the following:
- Double-click the appropriate relationship table in the DITA Maps Manager (on page 2897).
- Select the relationship table in the DITA Maps Manager (on page 2897) and press Enter.
- Select Open from the contextual menu of the relationship table in the DITA Maps Manager (on page 2897).

Publishing DITA Output

As a structured writing format, DITA produces structured content (content that is annotated with specific structural and semantic information rather than with formatting information). To create a publication, your DITA map (on page 3218) and its associated topics must be processed by a transformation script. That script is responsible for how the structural and semantic information in the DITA files is converted into formatting information for display.

Oxygen XML Editor publishes DITA content to various output sources using a bundled version of the DITA Open Toolkit. The DITA-OT is an open-source publishing engine that can publish DITA content to various output sources such as XHTML, PDF, or Windows Help (CHM). Since it has a plugin-based architecture, it can be extended with extra plugins that either define new formats for conversion or customize an existing conversion format. You can run the DITA-OT from Oxygen XML Editor using a transformation scenario or you can run it directly from a command line: http://www.dita-ot.org/dev/topics/building-output.html.

The DITA-OT that comes bundled with Oxygen XML Editor contains more plugins than the standard DITA-OT that can be downloaded from their official website. For example, it contains pre-installed plugins for converting DITA content to Word, EPUB, WebHelp, or to publish to PDF using CSS to customize the output.

You can download and install extra publishing plugins either from the DITA Open Toolkit registry or from the list of free plugins (on page 3168) on the Oxygen XML Editor GitHub account.

**Warning:**

Keep in mind that there could be instances where there are differences between what you see in Author mode and what you see in the published output. This is typically due to certain limitations in the publishing engine, especially when the source documents contain advanced constructs such as
DITA Map Transformation Scenarios

Built-in transformation scenarios allow you to transform DITA maps to a variety of outputs, such as WebHelp, PDF, ODF, XHTML, EPUB, CHM, Kindle, and MS Word. Oxygen XML Editor also includes a special Run DITA-OT Integrator that can be used to integrate a DITA-OT plugin and a DITA Map Metrics Report transformation that generates a statistics report for your DITA map. All of them are listed in the DITA Map section in the Configure Transformation Scenario(s) dialog box.

A variety of transformations scenarios are available for DITA maps:

- **Built-in transformation scenarios** allow you to transform a DITA map to a variety of outputs, such as WebHelp, PDF, ODF, XHTML, EPUB, CHM, Kindle, and MS Word.
- **Run DITA-OT Integrator** - Use this transformation scenario if you want to integrate a DITA-OT plugin. This scenario runs an Ant task that integrates all the plugins from the DITA-OT/plugins directory.
- **DITA Map Metrics Report** - Use this type of transformation scenario if you want to generate a DITA map statistics report. It contains information such as:
  - The number of processed maps and topics.
  - Content reuse percentage.
  - Number of elements, attributes, words, and characters used in the entire DITA map structure.
  - DITA conditional processing attributes used in the DITA maps.
  - Processing instructions.
  - External links.
  - All @outputclass attribute values gathered from the DITA project.

**Important:**

If you have cross references that point to content outside the scope of the DITA map, that referenced content will not be counted. For example, if you have links to topics that are not included in the DITA map hierarchy, the content in those topics is ignored when generating the statistics.

**Related Information:**
- Editing a Transformation Scenario
- Configure Transformation Scenario(s) Dialog Box
- Applying Associated Transformation Scenarios
- DITA Topic Transformation Scenarios
DITA Map WebHelp Responsive Transformation

DITA content can be transformed into several types of WebHelp Responsive systems (with or without a feedback section). The WebHelp Responsive layout and features (on page 1560) are designed to adapt to any device and screen size to provide an optimal viewing and interaction experience. Oxygen XML Editor also provides numerous possibilities for customizing the WebHelp Responsive output (on page 1642).

WebHelp Responsive Transformation Scenario

To publish a DITA map (on page 3218) as WebHelp Responsive output, follow these steps:

1. Select the Configure Transformation Scenario(s) action from the DITA Maps Manager (on page 2897) toolbar.
2. Select the DITA Map WebHelp Responsive scenario from the DITA Map section.
3. If you want to configure the transformation, click the Edit button.

**Step Result:** This opens an Edit scenario configuration dialog box that allows you to configure various options in the following tabs:

- **Templates Tab (on page 3106)** - This tab contains a set of built-in publishing templates (on page 1603) that you can use for the layout of your WebHelp system output. You can also create your own publishing templates or edit existing ones (on page 1642).
- **Parameters Tab (on page 3112)** - This tab includes numerous parameters that can be set to customize your WebHelp system output. See the Parameters section (on page ) below for details about the most commonly used parameters for WebHelp Responsive transformations.
- **Feedback Tab (on page 3113)** - This tab is for those who want to add the Oxygen Feedback comments component at the bottom of each WebHelp page so that you can interact with your readers.
- **Filters Tab (on page 3114)** - This tab allows you to filter certain content elements from the generated output.
- **Advanced Tab (on page 3116)** - This tab allows you to specify some advanced options for the transformation scenario.
- **Output Tab (on page 3118)** - This tab allows you to configure options that are related to the location where the output is generated.
4. Click Apply associated to process the transformation.

**Result:** When the DITA Map WebHelp Responsive transformation is complete, the output is automatically opened in your default browser.

General Parameters for Customizing WebHelp Responsive Output

To customize a transformation scenario, you can edit various parameters, including the following most commonly used ones:

- `default.language`
This parameter is used if the language is not detected in the DITA map. The default value is en-us.

clean.output

Deletes all files from the output folder before the transformation is performed (only no and yes values are valid and the default value is no).

editlink.remote.ditamap.url

Use this parameter in conjunction with editlink.web.author.url to add an Edit link next to the topic title in the WebHelp output. When a user clicks the link, the topic is opened in Oxygen XML Web Author where they can make changes that can be saved to a file server. The value should be set as the custom URL of the main DITA map. For example, a GitHub custom URL might look like this: https://getFileContent/oxyengxml/userguide/master/UserGuide.ditamap.

editlink.web.author.url

This parameter needs to be used in conjunction with editlink.remote.ditamap.url to add an Edit link next to the topic title in the WebHelp output. When a user clicks the link, the topic is opened in Oxygen XML Web Author where they can make changes that can be saved to a file server. The value should be set as the URL of the Web Author installation. For example: https://www.oxygenxml.com/oxygen-xml-web-author/.

editlink.present.only.path.to.topic

When this parameter is set to "true", the DITA topic path is displayed to the right of each topic title in the WebHelp Responsive output. Also, when this parameter is used, the editlink.ditamap.edit.url, editlink.remote.ditamap.url, and editlink.web.author.url parameters are ignored.

fix.external.refs.com.oxygenxml (Only supported when the DITA-OT transformation process is started from Oxygen XML Editor)

The DITA Open Toolkit usually has problems processing references that point to locations outside of the directory of the processed DITA map. This parameter is used to specify whether or not the application should try to fix such references in a temporary files folder before the DITA Open Toolkit is invoked on the fixed references. The fix has no impact on your edited DITA content. Allowed values: true or false (default).

force.unique

When set to true (default value), the transformation will be forced to create unique output files for each instance of a resource when a map contains multiple references to a single topic.

use.stemming

Controls whether or not you want to include stemming search algorithms into the published output (default setting is false).

webhelp.custom.resources

The file path to a directory that contains resources files. All files from this directory will be copied to the root of the WebHelp output.
webhelp.favicon

The file path that points to an image to be used as a favicon in the WebHelp output.

webhelp.reload.stylesheet

Set this parameter to true if you have out of memory problems when generating WebHelp. It will increase processing time but decrease the memory footprint. The default value is false.

webhelp.search.custom.excludes.file

The path of the file that contains name patterns for HTML files that should not be indexed by the WebHelp search engine. Each exclude pattern must be on a new line. The patterns are considered to be relative to the output directory, and they accept wildcards such as ⋆ (matches zero or more characters) or ? (matches one character). For more information about the patterns, see https://ant.apache.org/manual/dirtasks.html#patterns.

webhelp.search.japanese.dictionary

The file path of the dictionary that will be used by the Kuromoji morphological engine for indexing Japanese content in the WebHelp pages. The encoding for the dictionary must be UTF8.

webhelp.search.enable.pagination

Specifies whether or not search results will be displayed on multiple pages. Allowed values are yes or no.

webhelp.search.index.elements.to.exclude

Specifies a list of HTML elements that will not be indexed by the search engine. The value of the @class attribute can be used to exclude specific HTML elements from indexing. For example, the div.not-indexed value will not index all <div> elements that have a @class attribute with the value of not-indexed. Use a comma separator to specify more than one element.

webhelp.search.page.numberOfItems

Specifies the number of search results items displayed on each page. This parameter is only used when the webhelp.search.enable.pagination parameter is enabled.

webhelp.search.ranking

If this parameter is set to false then the 5-star rating mechanism is no longer included in the search results that are displayed on the Search tab (default setting is true).

webhelp.search.stop.words.include

Specifies a list of words that will be ignored by the search engine. Use a comma separator to specify more than one word.

webhelp.show.changes.and.comments

When set to yes, user comments, replies to comments, and tracked changes are published in the WebHelp output. The default value is no.

webhelp.sitemap.base.url
Base URL for all the `<loc>` elements in the generated `sitemap.xml` file. If this parameter is specified, the `<loc>` element will contain the value of this parameter plus the relative path to the page. If this parameter is not specified, the `<loc>` element will only contain the relative path of the page (the relative file path from the `@href` attribute of a `<topicref>` element from the `DITA map`, appended to this base URL value).

**webhelp.sitemap.change.frequency**

The value of the `<changefreq>` element in the generated `sitemap.xml` file. The `<changefreq>` element is optional in `sitemap.xml`. If you leave this parameter set to its default empty value, then the `<changefreq>` element is not added in `sitemap.xml`. Allowed values: `<empty string>` (default), `always`, `hourly`, `daily`, `weekly`, `monthly`, `yearly`, `never`.

**webhelp.sitemap.priority**

The value of the `<priority>` element in the generated `sitemap.xml` file. It can be set to any fractional number between 0.0 (least important priority) and 1.0 (most important priority). For example, 0.3, 0.5, or 0.8. The `<priority>` element is optional in `sitemap.xml`. If you leave this parameter set to its default empty value, then the `<priority>` element is not added in `sitemap.xml`.

**Parameters Specific to Oxygen WebHelp Responsive**

**webhelp.fragment.feedback**

You can integrate Oxygen Feedback with your WebHelp Responsive output to provide a comments area at the bottom of each page where readers can offer feedback. When you create an Oxygen Feedback site configuration, an HTML fragment is generated during the final step of the creation process and that fragment should be set as the value for this parameter.

**webhelp.default.collection.type.sequence**

Specifies if the `sequence` value will be used by default when the `@collection-type` attribute is not specified. This option is helpful if you want to have Next and Previous navigational buttons generated for all HTML pages. Allowed values are `no` (default) and `yes`.

**webhelp.enable.search.autocomplete**

Specifies if the Autocomplete feature is enabled in the WebHelp search text field. The default value is `yes`.

**webhelp.enable.html.fragments.cleanup**

Enables or disables the automatic conversion of HTML fragments to well-formed XML. If set to `true` (default), the transformation automatically converts non-well-formed HTML content to a well-formed XML equivalent. If set to `false`, the transformation will fail if at least one HTML fragment is not well-formed.

**webhelp.fragment.after.body**
This parameter can be used to display a given XHTML fragment after the body in all types of pages. The value of the parameter can be either a well-formed XHTML fragment or a path to a file that contains a well-formed XHTML fragment.

**webhelp.fragment.after.body.main.page**

This parameter can be used to display a given XHTML fragment after the body in the main page. The value of the parameter can be either a well-formed XHTML fragment or a path to a file that contains a well-formed XHTML fragment.

**webhelp.fragment.after.body.topic.page**

This parameter can be used to display a given XHTML fragment after the body in the topic page. The value of the parameter can be either a well-formed XHTML fragment or a path to a file that contains a well-formed XHTML fragment.

**webhelp.fragment.after.body.search.page**

This parameter can be used to display a given XHTML fragment after the body in the search results page. The value of the parameter can be either a well-formed XHTML fragment or a path to a file that contains a well-formed XHTML fragment.

**webhelp.fragment.after.body.terms.page**

This parameter can be used to display a given XHTML fragment after the body in the index terms page. The value of the parameter can be either a well-formed XHTML fragment or a path to a file that contains a well-formed XHTML fragment.

**webhelp.fragment.after.logo_and_title**

This parameter can be used to display a given XHTML fragment after the logo and title in all types of pages. The value of the parameter can be either a well-formed XHTML fragment or a path to a file that contains a well-formed XHTML fragment.

**webhelp.fragment.after.search.input**

This parameter can be used to display a given XHTML fragment after the search field in all types of pages. The value of the parameter can be either a well-formed XHTML fragment or a path to a file that contains a well-formed XHTML fragment.

**webhelp.fragment.after.main.page.search (deprecated)**

This parameter is deprecated. Use [webhelp.fragment.after.search.input.main.page](#) instead.

**webhelp.fragment.after.search.input.main.page**

This parameter can be used to display a given XHTML fragment after the search field in all the main page. The value of the parameter can be either a well-formed XHTML fragment or a path to a file that contains a well-formed XHTML fragment.

**webhelp.fragment.after.search.input.topic.page**

This parameter can be used to display a given XHTML fragment after the search field in all the topic page. The value of the parameter can be either a well-formed XHTML fragment or a path to a file that contains a well-formed XHTML fragment.
**webhelp.fragment.after.search.input.search.page**

This parameter can be used to display a given XHTML fragment after the search field in all the search results page. The value of the parameter can be either a well-formed XHTML fragment or a path to a file that contains a well-formed XHTML fragment.

**webhelp.fragment.after.search.input.terms.page**

This parameter can be used to display a given XHTML fragment after the search field in all the index terms page. The value of the parameter can be either a well-formed XHTML fragment or a path to a file that contains a well-formed XHTML fragment.

**webhelp.fragment.after.toc_or_tiles**

This parameter can be used to display a given XHTML fragment after the table of contents or tiles in the main page. The value of the parameter can be either a well-formed XHTML fragment or a path to a file that contains a well-formed XHTML fragment.

**webhelp.fragment.after.top_menu**

This parameter can be used to display a given XHTML fragment after the top menu in all types of pages. The value of the parameter can be either a well-formed XHTML fragment or a path to a file that contains a well-formed XHTML fragment.

**webhelp.fragment.before.body**

This parameter can be used to display a given XHTML fragment before the page body in all types of pages. The value of the parameter can be either a well-formed XHTML fragment or a path to a file that contains a well-formed XHTML fragment.

**webhelp.fragment.before.body.main.page**

This parameter can be used to display a given XHTML fragment before the page body in the main page. The value of the parameter can be either a well-formed XHTML fragment or a path to a file that contains a well-formed XHTML fragment.

**webhelp.fragment.before.body.topic.page**

This parameter can be used to display a given XHTML fragment before the page body in the topic page. The value of the parameter can be either a well-formed XHTML fragment or a path to a file that contains a well-formed XHTML fragment.

**webhelp.fragment.before.body.search.page**

This parameter can be used to display a given XHTML fragment before the page body in the search results page. The value of the parameter can be either a well-formed XHTML fragment or a path to a file that contains a well-formed XHTML fragment.

**webhelp.fragment.before.body.terms.page**

This parameter can be used to display a given XHTML fragment before the page body in the index terms page. The value of the parameter can be either a well-formed XHTML fragment or a path to a file that contains a well-formed XHTML fragment.

**webhelp.fragment.before.logo_and_title**
This parameter can be used to display a given XHTML fragment before the logo and title. The value of the parameter can be either a well-formed XHTML fragment or a path to a file that contains a well-formed XHTML fragment.

**webhelp.fragment.before.search.input**

This parameter can be used to display a given XHTML fragment before the search field in all types of pages. The value of the parameter can be either a well-formed XHTML fragment or a path to a file that contains a well-formed XHTML fragment.

**webhelp.fragment.before.main.page.search (deprecated)**

This parameter is deprecated. Use `webhelp.fragment.before.search.input.main.page` instead.

**webhelp.fragment.before.search.input.main.page**

This parameter can be used to display a given XHTML fragment before the search field in the main page. The value of the parameter can be either a well-formed XHTML fragment or a path to a file that contains a well-formed XHTML fragment.

**webhelp.fragment.before.search.input.topic.page**

This parameter can be used to display a given XHTML fragment before the search field in the topic page. The value of the parameter can be either a well-formed XHTML fragment or a path to a file that contains a well-formed XHTML fragment.

**webhelp.fragment.before.search.input.search.page**

This parameter can be used to display a given XHTML fragment before the search field in the search results page. The value of the parameter can be either a well-formed XHTML fragment or a path to a file that contains a well-formed XHTML fragment.

**webhelp.fragment.before.search.input.terms.page**

This parameter can be used to display a given XHTML fragment before the search field in the index terms page. The value of the parameter can be either a well-formed XHTML fragment or a path to a file that contains a well-formed XHTML fragment.

**webhelp.fragment.before.toc_or_tiles**

This parameter can be used to display a given XHTML fragment before the table of contents or tiles in the main page. The value of the parameter can be either a well-formed XHTML fragment or a path to a file that contains a well-formed XHTML fragment.

**webhelp.fragment.before.top_menu**

This parameter can be used to display a given XHTML fragment before the top menu in all types of pages. The value of the parameter can be either a well-formed XHTML fragment or a path to a file that contains a well-formed XHTML fragment.

**webhelp.fragment.footer**

This parameter can be used to display a given XHTML fragment as the page footer in all types of pages. The value of the parameter can be either a well-formed XHTML fragment or a path to a file that contains a well-formed XHTML fragment.
Important:
This parameter should only be used if you are using a valid, purchased license of Oxygen XML Editor (do not use it with a trial license).

**webhelp.fragment.head**
This parameter can be used to display a given XHTML fragment in the header section in all types of pages. The value of the parameter can be either a **well-formed** XHTML fragment or a path to a file that contains a **well-formed** XHTML fragment.

**webhelp.fragment.head.main.page**
This parameter can be used to display a given XHTML fragment in the header section in the main page. The value of the parameter can be either a **well-formed** XHTML fragment or a path to a file that contains a **well-formed** XHTML fragment.

**webhelp.fragment.head.topic.page**
This parameter can be used to display a given XHTML fragment in the header section in the topic page. The value of the parameter can be either a **well-formed** XHTML fragment or a path to a file that contains a **well-formed** XHTML fragment.

**webhelp.fragment.head.search.page**
This parameter can be used to display a given XHTML fragment in the header section in the search results page. The value of the parameter can be either a **well-formed** XHTML fragment or a path to a file that contains a **well-formed** XHTML fragment.

**webhelp.fragment.head.terms.page**
This parameter can be used to display a given XHTML fragment in the header section in the index terms page. The value of the parameter can be either a **well-formed** XHTML fragment or a path to a file that contains a **well-formed** XHTML fragment.

**webhelp.fragment.welcome**
This parameter can be used to display a given XHTML fragment as a welcome message (or title). The value of the parameter can be either a **well-formed** XHTML fragment or a path to a file that contains a **well-formed** XHTML fragment.

**webhelp.fragment.after.header**
This parameter can be used to display a given XHTML fragment after the header section in all types of pages. The value of the parameter can be either a **well-formed** XHTML fragment or a path to a file that contains a **well-formed** XHTML fragment.

**webhelp.fragment.after.header.main.page**
This parameter can be used to display a given XHTML fragment after the header section in the main page. The value of the parameter can be either a **well-formed** XHTML fragment or a path to a file that contains a **well-formed** XHTML fragment.

**webhelp.fragment.after.header.topic.page**
This parameter can be used to display a given XHTML fragment after the header section in the topic page. The value of the parameter can be either a **well-formed** XHTML fragment or a path to a file that contains a **well-formed** XHTML fragment.

**webhelp.fragment.after.header.search.page**

This parameter can be used to display a given XHTML fragment after the header section in the search results page. The value of the parameter can be either a **well-formed** XHTML fragment or a path to a file that contains a **well-formed** XHTML fragment.

**webhelp.fragment.after.header.terms.page**

This parameter can be used to display a given XHTML fragment after the header section in the index terms page. The value of the parameter can be either a **well-formed** XHTML fragment or a path to a file that contains a **well-formed** XHTML fragment.

**webhelp.fragment.before.search.input**

This parameter can be used to display a given XHTML fragment before the search field in all types of pages. The value of the parameter can be either a **well-formed** XHTML fragment or a path to a file that contains a **well-formed** XHTML fragment.

**webhelp.fragment.after.search.input**

This parameter can be used to display a given XHTML fragment after the search field in all types of pages. The value of the parameter can be either a **well-formed** XHTML fragment or a path to a file that contains a **well-formed** XHTML fragment.

**webhelp.fragment.before.main.content.area**

This parameter can be used to display a given XHTML fragment before the main content section in all types of pages. The value of the parameter can be either a **well-formed** XHTML fragment or a path to a file that contains a **well-formed** XHTML fragment.

**webhelp.fragment.before.main.content.area.main.page**

This parameter can be used to display a given XHTML fragment before the main content section in the main page. The value of the parameter can be either a **well-formed** XHTML fragment or a path to a file that contains a **well-formed** XHTML fragment.

**webhelp.fragment.before.main.content.area.topic.page**

This parameter can be used to display a given XHTML fragment before the main content section in the topic page. The value of the parameter can be either a **well-formed** XHTML fragment or a path to a file that contains a **well-formed** XHTML fragment.

**webhelp.fragment.before.main.content.area.search.page**

This parameter can be used to display a given XHTML fragment before the main content section in the search results page. The value of the parameter can be either a **well-formed** XHTML fragment or a path to a file that contains a **well-formed** XHTML fragment.

**webhelp.fragment.before.main.content.area.terms.page**
This parameter can be used to display a given XHTML fragment before the main content section in the index terms page. The value of the parameter can be either a well-formed XHTML fragment or a path to a file that contains a well-formed XHTML fragment.

**webhelp.fragment.after.main.content.area**

This parameter can be used to display a given XHTML fragment after the main content section in all types of pages. The value of the parameter can be either a well-formed XHTML fragment or a path to a file that contains a well-formed XHTML fragment.

**webhelp.fragment.after.main.content.area.main.page**

This parameter can be used to display a given XHTML fragment after the main content section in the main page. The value of the parameter can be either a well-formed XHTML fragment or a path to a file that contains a well-formed XHTML fragment.

**webhelp.fragment.after.main.content.area.topic.page**

This parameter can be used to display a given XHTML fragment after the main content section in the topic page. The value of the parameter can be either a well-formed XHTML fragment or a path to a file that contains a well-formed XHTML fragment.

**webhelp.fragment.before.topic.toolbar**

This parameter can be used to display a given XHTML fragment before the toolbar buttons above the topic content in the topic page. The value of the parameter can be either a well-formed XHTML fragment or a path to a file that contains a well-formed XHTML fragment.

**webhelp.fragment.after.topic.toolbar**

This parameter can be used to display a given XHTML fragment after the toolbar buttons above the topic content in the topic page. The value of the parameter can be either a well-formed XHTML fragment or a path to a file that contains a well-formed XHTML fragment.

**webhelp.fragment.before.topic.breadcrumb**

This parameter can be used to display a given XHTML fragment before the breadcrumb component in the topic page. The value of the parameter can be either a well-formed XHTML fragment or a path to a file that contains a well-formed XHTML fragment.

**webhelp.fragment.after.topic.breadcrumb**

This parameter can be used to display a given XHTML fragment after the breadcrumb component in the topic page. The value of the parameter can be either a well-formed XHTML fragment or a path to a file that contains a well-formed XHTML fragment.

**webhelp.fragment.before.publication.toc**

This parameter can be used to display a given XHTML fragment before the publication's table of contents component in the topic page. The value of the parameter can be either a well-formed XHTML fragment or a path to a file that contains a well-formed XHTML fragment.

**webhelp.fragment.after.publication.toc**

This parameter can be used to display a given XHTML fragment after the publication's table of contents component in the topic page. The value of the parameter can be either a well-formed XHTML fragment or a path to a file that contains a well-formed XHTML fragment.
This parameter can be used to display a given XHTML fragment before the publication's table of contents component in the topic page. The value of the parameter can be either a well-formed XHTML fragment or a path to a file that contains a well-formed XHTML fragment.

**webhelp.fragment.before.topic.content**

This parameter can be used to display a given XHTML fragment before the topic's content in the topic page. The value of the parameter can be either a well-formed XHTML fragment or a path to a file that contains a well-formed XHTML fragment.

**webhelp.fragment.after.topic.content**

This parameter can be used to display a given XHTML fragment after the topic's content in the topic page. The value of the parameter can be either a well-formed XHTML fragment or a path to a file that contains a well-formed XHTML fragment.

**webhelp.fragment.before.feedback**

This parameter can be used to display a given XHTML fragment before the Oxygen Feedback commenting component in the topic page. The value of the parameter can be either a well-formed XHTML fragment or a path to a file that contains a well-formed XHTML fragment.

**webhelp.fragment.after.feedback**

This parameter can be used to display a given XHTML fragment after the Oxygen Feedback commenting component in the topic page. The value of the parameter can be either a well-formed XHTML fragment or a path to a file that contains a well-formed XHTML fragment.

**webhelp.fragment.before.topic.toc**

This parameter can be used to display a given XHTML fragment before the topic's table of contents component in the topic page. The value of the parameter can be either a well-formed XHTML fragment or a path to a file that contains a well-formed XHTML fragment.

**webhelp.fragment.after.topic.toc**

This parameter can be used to display a given XHTML fragment after the topic's table of contents component in the topic page. The value of the parameter can be either a well-formed XHTML fragment or a path to a file that contains a well-formed XHTML fragment.

**webhelp.fragment.custom.search.engine.results**

This parameter can be used to replace the search results area with custom XHTML content. The value of the parameter is the path to an XHTML file that contains your custom content.

**webhelp.fragment.custom.search.engine.script**

This parameter can be used to replace WebHelp's built-in search engine with your own custom search engine. The value of the parameter is the path to an XHTML file that contains the scripts required for your custom search engine to run.

**webhelp.merge.nested.topics.related.links**

Specifies if the related links from nested topics will be merged with the links in the parent topic. Thus the links will be moved from the topic content to the related links component and all
of the links from the same group (for example, Related Tasks, Related References, Related Information) are merged into a single group. The default value is yes.

**webhelp.search.stop.words.exclude**

Specifies a list of words that will be excluded from the default list of stop words that are filtered out before the search processing. Use comma separators to specify more than one word (for example: if,for,is).

**webhelp.show.breadcrumb**

Specifies if the breadcrumb component will be presented in the output. The default value is yes.

**webhelp.show.child.links**

Specifies if child links will be generated in the output for all topics that have subtopics. The default value is no.

**webhelp.show.indexterms.link**

Specifies if an icon that links to the index will be presented in the output. The default value is yes.

**webhelp.show.main.page.tiles**

Specifies if the tiles component will be presented in the main page of the output. For a tree style layout, this parameter should be set to no.

**webhelp.show.main.page.toc**

Specifies if the table of contents will be presented in the main page of the output. The default value is yes.

**webhelp.show.navigation.links**

Specifies if navigation links will be presented in the output. The default value is yes.

**webhelp.show.print.link**

Specifies if a print link or icon will be presented within each topic in the output. The default value is yes.

**webhelp.show.related.links**

Specifies if the related links component will be presented in the WebHelp Responsive output. The default value is yes. The webhelp.merge.nested.topics.related.links parameter can be used in conjunction with this one to merge the related links from nested topics into the links in the parent topic.

**webhelp.show.publication.toc**

Specifies if a table of contents will be presented on the left side of each topic in the output. The default value is yes.

**webhelp.show.topic.toc**

Specifies if a topic table of contents will be presented on the right side of each topic in the output. This table of contents contains links to each `<section>` within the current topic that
contains an `@id` attribute and the section corresponding to the current scroll position is highlighted. The default value is `yes`.

**webhelp.show.top.menu**

Specifies if a menu will be presented at the topic of the main page in the output. The default value is `yes`.

**webhelp.publication.toc.hide.chunked.topics**

Specifies if the table of contents will contain links for *chunked* topics. The default value is `yes`.

**webhelp.publication.toc.links**

Specifies which links will be included in the table of contents. The possible values are:

- `chapter` (default) - The TOC will include links for the current topic, its children, its siblings, and its direct ancestor (including the direct ancestor’s siblings), and the parent chapter.
- `topic` - The TOC will only include links for the current topic and its direct children.
- `all` - The TOC will include all links.

**webhelp.publication.toc.tooltip.position**

By default, if a topic contains a `<shortdesc>` element, its content is displayed in a tooltip when the user hovers over its link in the table of contents. This parameter controls whether or not this tooltip is displayed and its position relative to the link. The possible values are:

- `left`
- `right` (default)
- `top`
- `bottom`
- `hidden` - The tooltip will not be displayed.

**webhelp.top.menu.activated.on.click**

When this parameter is activated (set to `yes`), clicking an item in the top menu will expand the submenu (if available). You can then click on a submenu item to open the item (topic). You can click outside the menu or press `ESC` to hide the menu. When set to `no` (default), hovering over a menu item displays the menu content.

**webhelp.top.menu.depth**

Specifies the maximum depth level of the topics that will be included in the top menu. The default value is `3`. A value of `0` means that the menu has unlimited depth.

**webhelp.topic.collapsible.elements.initial.state**

Specifies the initial state of collapsible elements (tables with titles, nested topics with titles, sections with titles, index term groups). The possible values are `collapsed` or `expanded` (default value).
Parameters for Adding a Link to PDF Documentation in WebHelp Responsive Output

The following transformation parameters can be used to generate a PDF link component in the WebHelp Responsive output (for example, it could link to the PDF equivalent of the documentation):

- **webhelp.pdf.link.url**
  Specifies the target URL for the PDF link component.

- **webhelp.pdf.link.text**
  Specifies the text for the PDF link component.

- **webhelp.pdf.link.icon.path**
  Specifies the path or URL of the image icon to be used for the PDF link component. If not specified, a default icon is used.

- **webhelp.show.pdf.link**
  Specifies whether or not the PDF link component is shown in the WebHelp Responsive output. Allowed values are: **yes** (default) and **no**.

- **webhelp.pdf.link.anchor.enabled**
  Specifies whether or not the current topic ID should be appended as the name destination at the end of the PDF link. Allowed values are: **yes** (default) and **no**.

Related information

- Customizing WebHelp Responsive Output *(on page 1642)*
- Layout and Features *(on page 1560)*

DITA Map PDF - based on HTML5 & CSS Transformation

Oxygen XML Editor includes a built-in **DITA Map PDF - based on HTML5 & CSS** transformation scenario based on a **DITA-OT CSS-based PDF Publishing plugin** that converts DITA maps to PDF using a CSS-based processing engine and an HTML5 intermediate format. Oxygen XML Editor comes bundled with a built-in CSS-based PDF processing engine called **Oxygen PDF Chemistry**. Oxygen XML Editor also supports some third-party processors.

For those who are familiar with CSS, this makes it very easy to style and customize the PDF output of your DITA projects without having to work with **xsl:fo** customizations. This transformation also includes some built-in publishing templates that you can use for the layout of your PDF output and you can create your own templates or edit existing ones.

The following CSS-based PDF processors can be used:
• **Oxygen PDF Chemistry** - A built-in processor that is bundled with Oxygen XML Editor. For more information, see the Oxygen PDF Chemistry User Guide. This is the supported processor.

• **Prince Print with CSS** (not included in the Oxygen XML Editor installation kit) - A third-party component that needs to be purchased from [http://www.princexml.com](http://www.princexml.com).

• **Antenna House Formatter** (not included in the Oxygen XML Editor installation kit) - A third-party component that needs to be purchased from [http://www.antennahouse.com/antenna1/formatter/](http://www.antennahouse.com/antenna1/formatter/).

### How to Create the Transformation Scenario

To create a **DITA Map PDF - based on HTML5 & CSS** transformation scenario, follow these steps:

1. Click the 🏅 **Configure Transformation Scenario(s)** button from the **DITA Maps Manager** (on page 2897) toolbar.
2. Select the **DITA Map PDF - based on HTML5 & CSS** transformation scenario.
3. If you want to configure the transformation, click the **Edit** button.

**Step Result:** This opens an **Edit scenario** configuration dialog box that allows you to configure various options in the following tabs:

- **Templates Tab (on page 3106)** - This tab contains a set of built-in publishing templates that you can use for the layout of your WebHelp system output. You can also create your own publishing templates by saving one from the gallery and changing it.

**Figure 787. DITA Map to PDF Templates**

![Figure 787. DITA Map to PDF Templates](image)
4. In the **Parameters** tab, configure any of the following parameters (if applicable):

- **args.css** - Specifies a path to a custom CSS to be used in addition to those specified in the publishing template. The files must have URL syntax and be separated using semicolons. Also, the `dita.css.list` parameter must be left empty to use these files in addition to the selection in the Styles drop-down menu.

- **css.processor.type** - This is where you choose the processor type. You can select between Oxygen PDF Chemistry, Prince XML, or Antenna House.

- **css.processor.path.chemistry** (if you are using the Oxygen PDF Chemistry processor) - Specifies the path to the Oxygen PDF Chemistry executable file that will be run to generate the PDF. If this parameter is not set, the transformation will use the processor specified in the CSS-based Processors preferences page (on page 264).

- **css.processor.path.prince** (if you are using the Prince Print with CSS processor) - Specifies the path to the Prince executable file that will be run to produce the PDF. If you installed Prince using its default settings, you can leave this blank.

- **css.processor.path.antenna-house** (if you are using the Antenna House Formatter processor) - Specifies the path to the Antenna House executable file that will be run to produce the PDF. If you installed Antenna House using its default settings, you can leave this blank.

- **show.changes.and.comments** - When set to yes, user comments, replies to comments, and tracked changes are published in the PDF output. The default value is no.

- **figure.title.placement** - Controls the position of the figure title relative to the image. Allowed values are “top” and “bottom”; “top” is the default

5. Click **OK** and run the transformation scenario.

### Customizing the Output

For information about customizing the output, see **CSS-based DITA to PDF Customization (on page 1768)**.

**Related Information:**

- Editing a Transformation Scenario (on page 1545)
- Configure Transformation Scenario(s) Dialog Box (on page 1548)
- Oxygen PDF Chemistry User Guide
- CSS-based DITA to PDF Customization (on page 1768)
DITA Map PDF - based on XSL-FO Transformation

Oxygen XML Editor comes bundled with the DITA Open Toolkit that provides a mechanism for converting DITA maps to PDF output.

Creating a DITA Map PDF - based on XSL-FO Transformation Scenario

To create a DITA Map PDF - based on XSL-FO transformation scenario, follow these steps:

1. Click the Configure Transformation Scenario(s) button from the DITA Maps Manager toolbar.
2. Select DITA Map PDF - based on XSL-FO and click the Edit button (or use the Duplicate button if your framework is read-only).
3. Use the various tabs to configure the transformation scenario. In the Parameters tab, you can use a variety of parameters to customize the output. For example, the following parameters are just a few of the most commonly used ones:
   - `show.changes.and.comments` - If set to yes, user comments, replies to comments, and tracked changes are published in the PDF output.
   - `customization.dir` - Specifies the path to a customization directory.
   - `editlink.present.only.path.to.topic` - When this parameter is set to "true", the DITA topic path is displayed to the right of each topic title in the PDF output.
4. Click OK and then the Apply Associated button to run the transformation scenario.

Related Information:
XSL FO-based DITA to PDF Customization

DITA Map MS Office Word Transformation

Oxygen XML Editor comes bundled with a transformation scenario that allows you to convert DITA maps to Microsoft Office Word documents. It utilizes the DITA to Word plugin created by Jarno Elovirta. This plugin contains a Word document named Normal.docx (located in: \OXYGEN_INSTALL_DIR\frameworks\dita\DITA-OT3.x\plugins\com.elovirta.ooxml\resources) that is used by the transformation scenario as a template to generate the final Word document.

Tip:
You can make general modifications to the Normal.docx template file to alter the published output.

Configuring the Transformation Scenario

To configure a DITA Map to MS Office Word transformation scenario, follow these steps:

1. Open the DITA map in the DITA Maps Manager.
2. Click the Configure Transformation Scenario(s) button from the DITA Maps Manager toolbar.
3. Select **DITA Map MS Office Word**.

4. For advanced customizations, in the **Parameters** tab you can use any of the following parameters that are unique to this transformation scenario to specify paths to files that affect the output in various ways:

   - **dotx.file** - Specifies the path to a Word template file (.docx) that will be used in the transformation to generate the final Word document. Set this parameter if you want to use a different template file other than the **Normal.docx** file that is used by default.
   - **document.flat.xsl** - Specifies the path to a pre-process clean-up stylesheet.
   - **core.xsl** - Specifies the path to a core metadata stylesheet.
   - **custom.xsl** - Specifies the path to a custom metadata stylesheet.
   - **document.xsl** - Specifies the path to a main document stylesheet.
   - **comments.xsl** - Specifies the path to a comments stylesheet.
   - **numbering.xsl** - Specifies the path to a list and title numbering stylesheet.
   - **footnotes.xsl** - Specifies the path to a footnote stylesheet.
   - **document.xml.xsl** - Specifies the path to a document relations metadata stylesheet.
   - **inkscape.exec** - Specifies the path to an Inkscape (open-source vector graphics editor) executable file.

5. Click **OK** and run the transformation scenario.

**Result:** The result of the transformation will automatically be opened in your system's default word processing application (such as Microsoft Word).

**Related Information:**
- Editing a Transformation Scenario (**on page 1545**)
- Configure Transformation Scenario(s) Dialog Box (**on page 1548**)
- Migrating MS Office Documents to DITA (**on page 3188**)

**DITA Map CHM (Compiled HTML Help) Transformation**

To perform a **Compiled HTML Help (CHM)** transformation, Oxygen XML Editor needs **Microsoft HTML Help Workshop** to be installed on your computer. Oxygen XML Editor automatically detects if **HTML Help Workshop** is installed and uses it.

**Note:**

**HTML Help Workshop** might fail if the files used for transformation contain accents in their names, due to different encodings used when writing the .hhp and .hhc files. If the transformation fails to produce the CHM output but the .hhp (HTML Help Project) file is already generated, you can manually try to build the CHM output using **HTML Help Workshop**.
Changing the Output Encoding

Oxygen XML Editor uses the `htmlhelp.locale` parameter to correctly display specific characters of different languages in the output of the Compiled HTML Help (CHM) transformation. By default, the DITA Map CHM transformation scenario that comes bundled with Oxygen XML Editor has the `htmlhelp.locale` parameter set to `en-US`.

To customize this parameter, follow this procedure:

1. Use the Configure Transformation Scenario(s) (Ctrl + Shift + C (Command + Shift + C on macOS)) action from the toolbar or the Document > Transformation menu.
2. Select the DITA Map CHM transformation scenario and click the Edit button.
3. In the Parameter tab, search for the `htmlhelp.locale` parameter and change its value to the desired language tag.

**Note:**
The format of the `htmlhelp.locale` parameter is `LL-CC`, where `LL` represents the language code (`en`, for example) and `CC` represents the country code (`us`, for example). The language codes are contained in the ISO 639-1 standard and the country codes are contained in the ISO 3166-1 standard. For further details about language tags, go to [http://www.rfc-editor.org/rfc/rfc5646.txt](http://www.rfc-editor.org/rfc/rfc5646.txt).

Customizing the CHM Output

There are several possibilities available for customizing the CHM output:

- You can use a custom CSS stylesheet to customize how the HTML content is rendered in the output:
  1. Create the custom CSS.
  2. Select the DITA Map CHM transformation scenario and click the Edit button.
  3. In the Parameter tab, set the `args.css` parameter to point to the location of your custom CSS and make sure the `args.copy.css` parameter is set to `yes` to instruct the transformation to copy the custom CSS to the output folder.
  4. Run the transformation.

- If you are familiar with XSLT, there are two XSLT stylesheets that are used in the transformation to compile various settings and components in the CHM output. They are found in the following directory: `OXGEN_INSTALL_DIR/frameworks/dita/DITA-OT3.x/plugins/org.dita.htmlhelp/xsl/map2htmlhelp`. The files are as follows:
  - `map2hhimpl.xsl` - This file is used to compile the table of contents.
  - `map2hhimpl.xsl` - This file contains information for compiling the CHM and various settings that are read by the HTML Help Workshop when creating the output.
DITA Map Kindle Transformation

Oxygen XML Editor requires KindleGen to generate Kindle output from DITA maps (on page 3218). To install KindleGen for use by Oxygen XML Editor, follow these steps:

1. Go to www.amazon.com/kindleformat/kindlegen and download the zip file that matches your operating system.
2. Unzip the file on your local disk.
3. Start Oxygen XML Editor and open a DITA map in the DITA Maps Manager view (on page 2897).
4. Click the Configure Transformation Scenario(s) button.
5. Select the DITA Map Kindle transformation and click the Edit button to edit it.
6. Go to Parameters tab and set the kindlegen.executable parameter as the path to the KindleGen directory.
7. Accept the changes.

DITA Map Zendesk Publishing

Restriction:
This feature is only available in the Enterprise edition of Oxygen XML Editor.

Oxygen XML Editor includes a built-in transformation scenario that provides the ability to publish DITA topics to XHTML output and upload them directly as articles to the Zendesk Help Center.

1. Start Oxygen XML Editor and open a DITA map in the DITA Maps Manager view (on page 2897).
2. Click the Configure Transformation Scenario(s) button.
3. Create a new DITA-OT transformation scenario and choose the Zendesk Help Center transformation type.
4. Go to Parameters tab and set the following parameters:
   
   **Host**
   The URL reference to the Zendesk Help Center (for example, https://your-domain.zendesk.com).

   **Username**
   The username (e-mail address) for the account used to upload the content.

   **API Token**

   **Article category**
   The name of the category where the articles are uploaded. The category needs to be created in the Zendesk admin pages: https://support.zendesk.com/hc/en-
Article section

The name of the section (inside the parent category) where articles are uploaded. The section needs to be created in the Zendesk admin pages: https://support.zendesk.com/hc/en-us/articles/218222877-Organizing-knowledge-base-content-in-categories-and-sections#topic_hjs_tl4_kk.

Create article draft

This setting controls whether the articles should be published (if the value is false) or saved as drafts (if the value is true). The default value is false.

Permission group name

The name of the Zendesk permission group that controls who edits and publishes articles: https://support.zendesk.com/hc/en-us/articles/203661966-Creating-managing-and-using-groups.

5. Save the changes and run the transformation.

For more information about publishing content to the Zendesk Help Center, watch our video demonstration:

https://www.youtube.com/embed/QZ_9Fk_LOk8

Run DITA-OT Integrator Transformation

Oxygen XML Editor comes bundled with a transformation scenario designed to integrate DITA-OT plugins (on page 3222). These DITA-OT plugins are used for various customizations. It is called Run DITA-OT Integrator and is found in the DITA Map section of the Configure Transformation Scenario(s) dialog box (on page 1548).

Attention:

The integration will be performed on the DITA-OT version specified in the DITA Open Toolkit section of the DITA preferences page (on page 269).

Running the Transformation Scenario

To integrate a DITA-OT plugin, follow these steps:

1. If Oxygen XML Editor was installed in the default location, you may need to restart and run it as an administrator.

2. Select the Apply Transformation Scenario(s) or Configure Transformation Scenario(s) (on page 1548) action from the DITA Maps Manager toolbar (you could also use the same action on the main toolbar or open the Transformation Scenarios view (on page 1555)).
3. Select the Run DITA-OT Integrator transformation scenario. If the integrator is not visible, select the Show all scenarios action that is available in the Settings drop-down menu.

4. Apply the scenario (on page 1548).

5. Check the Results panel at the bottom of the application to make sure the build was successful.

6. Restart Oxygen XML Editor with your normal permissions.

Related Information:
- Configure Transformation Scenario(s) Dialog Box (on page 1548)
- Installing a DITA-OT Plugin (on page 3164)
- Integrating a DITA Specialization (on page 3176)

DITA Topic Transformation Scenarios

Oxygen XML Editor includes built-in transformation scenarios for transforming individual DITA Topics to HTML5, XHTML, or PDF output. They can be found in the DITA section in the Configure Transformation Scenario(s) dialog box (on page 1548).

The available transformations scenarios for individual DITA topics include:

- **DITA HTML5** - This DITA-OT transformation scenario generates HTML5 output from a single DITA topic.
- **DITA XHTML** - This DITA-OT transformation scenario generates XHTML output from a single DITA topic. This was the first transformation scenario created for the DITA Open Toolkit and it originally served as the basis for all HTML-based transformations.
- **DITA PDF - based on HTML5 & CSS** - This transformation scenario converts individual DITA topics to PDF using a CSS-based processing engine and an HTML5 intermediate format. Oxygen XML Editor comes bundled with a built-in CSS-based PDF processing engine called Oxygen PDF Chemistry. Oxygen XML Editor also supports some third-party processors.

For those who are familiar with CSS, this makes it very easy to style and customize the PDF output of your DITA projects without having to work with xsl:fo customizations. Another advantage of this transformation scenario is that you can use the same customization CSS (on page 1814) or publishing template (on page 1783) that you use for converting entire DITA maps.

The transformation scenario automatically detects the currently selected context DITA map (root map) (on page 2901) so that keys and references are properly resolved (the detected context map is set as the value of the args.root.map parameter (this can be changed in the Parameters tab). It also automatically detects the currently applied profiling condition set (on page 3142) to be used as the default filtering option in the transformation scenario (this can be changed in the Filters tab).

The transformation scenario also supports a parameter named args.enable.root.map.key.processing that can be used to specify whether or not the values for @keyref and @conkeyref attributes within the transformed topics are resolved. The possible values are:
• no - This means that the values for all @keyref and @conkeyref attributes are ignored in the transformation. This results in lower processing times.
• yes - This means that the values for any @keyref and @conkeyref attributes found in the transformed topic are processed and resolved using the value of the args.root.map parameter.
• auto - This means that the process will search for any @keyref and @conkeyref attributes within the transformed topic and if any are found, the values will be processed and resolved using the value of the args.root.map parameter. If none are found, the @keyref and @conkeyref attributes are ignored.

• DITA PDF - based on XSL-FO - This DITA-OT transformation scenario converts individual DITA topics to PDF using an xsl:fo processor.

Related Information:
Editing a Transformation Scenario (on page 1545)
Configure Transformation Scenario(s) Dialog Box (on page 1548)
Applying Associated Transformation Scenarios (on page 1548)
DITA Map Transformation Scenarios (on page 3081)

Running a DITA Transformation Scenario
To select and run a transformation scenario on your DITA map, follow these steps:

1. Click the Configure Transformation Scenario(s) button on the DITA Maps Manager toolbar (on page 2899). The Configure Transformation Scenario(s) dialog box (on page 1548) appears. This dialog box lists all the transformation scenarios that have been configured in your project. Oxygen XML Editor provides a default set of transformation scenarios, but the people in charge of your DITA system may have provided others that are specifically configured for your needs.
2. Select the transformation scenario you want to run and click Apply Associated. The transformation scenario runs in the background. You can continue to work in Oxygen XML Editor while the transformation is running. If there are errors or warnings, Oxygen XML Editor displays them when the transformation is complete. If the transformation is successful, Oxygen XML Editor opens the output in the appropriate application.
3. To rerun the same scenario again, click the Apply Transformation Scenario(s) button.

Creating or Editing a DITA-OT Transformation

Creating a DITA-OT Transformation Scenario
To create a DITA-OT Transformation scenario, use one of the following methods:
• Use the Configure Transformation Scenario(s) (Ctrl + Shift + C (Command + Shift + C on macOS)) action from the DITA Maps Manager toolbar, main toolbar, or the Document > Transformation menu. Then click the New button and select DITA-OT Transformation.

• Go to Window > Show View and select Transformation Scenarios to display this view. Click the New Scenario drop-down menu button and select DITA-OT Transformation.

Both methods open the DITA Transformation Type dialog box that presents the list of possible outputs.

![DITA Transformation Type Dialog Box](image)

Select the desired type of output and click OK. This opens the New Scenario dialog box.

The upper part of the dialog box allows you to specify the Name of the transformation scenario and the following Storage options:

- **Project Options (on page 3222)** - The scenario is stored in the project file and can be shared with other users. For example, if your project is saved on a source versioning/sharing system (CVS, SVN, Source Safe, etc.) or a shared folder, your team can use the scenarios that you store in the project file.
- **Global Options (on page 3219)** - The scenario is saved in the global options that are stored in the user home directory and is not accessible to other users.

The lower part of the dialog box contains several tabs that allow you to configure the options that control the transformation.
Editing a DITA-OT Transformation Scenario

Editing a transformation scenario is useful if you need to configure some of its parameters.

To configure an existing transformation scenario, follow these steps:

1. Select the Configure Transformation Scenario(s) (Ctrl + Shift + C (Command + Shift + C on macOS)) action from the DITA Maps Manager toolbar, main toolbar, or the Document > Transformation menu.

   **Step Result:** The Configure Transformation Scenario(s) dialog box (on page 1548) is opened.

2. Select the particular transformation scenario and click the Edit button at the bottom of the dialog box or from the contextual menu.

   **Note:** Since transformation scenarios that are associated with built-in frameworks (on page 3219) are read-only, these scenarios will prompt you to use the Duplicate button and then edit the duplicated scenario (on page 1547).

   **Result:** This will open an Edit scenario configuration dialog box (on page 1545) that contains several tabs that allow you to configure the options that control the transformation.

**Related Information:**
- Creating a DITA-OT Plugin (on page 3160)
- Installing a DITA-OT Plugin (on page 3164)
- DITA Open Toolkit Documentation

**Templates Tab (DITA-OT Transformations)**

When you create a new transformation scenario (on page 1464) or edit an existing one (on page 1545), a configuration dialog box is displayed that allows you to customize the transformation with various options in several tabs.

The Templates tab is available for DITA-OT transformations with WebHelp Responsive or PDF - based on HTML5 & CSS output types and it provides a set of built-in publishing templates (on page 1603). You can use one of them to publish your documentation or as a starting point for a new publishing template.
Filtering and Previewing Templates

You can click on the tags at the top of the pane to filter the templates and narrow your search. Each built-in template also includes an Online preview icon in the bottom-right corner that opens a webpage in your default browser providing a sample of how the main page will look when that particular template is used to generate the output.

Built-in Templates Locations

Oxygen XML Editor scans the following locations to find the built-in templates to display in the dialog box:

- **WebHelp Responsive Templates** - All built-in WebHelp Responsive publishing templates are stored in the following directory: `DITA-OT-DIR/plugins/com.oxygenxml.webhelp.responsive/templates`.

- **PDF - based on HTML5 & CSS** - All built-in PDF publishing templates are stored in the following directories:
  - `DITA-OT-DIR/plugins/com.oxygenxml.pdf.css/templates`
  - `DITA-OT-DIR/plugins/com.oxygenxml.webhelp.responsive/templates`

Custom Templates Locations

Oxygen XML Editor scans the locations specified in the DITA > Publishing preferences page (on page 275) to find custom templates to display in the dialog box. You can access that preferences page directly from the Template tab by clicking on the Configure Publishing Templates Gallery link.
Selecting Custom Templates

Once you are finished configuring your template, you can click the Choose Custom Publishing Template link to select your template.

You can also add your custom templates (on page 1644) to the list of templates displayed in the Templates tab. To do this, store them in a directory, then click the Configure Publishing Templates Gallery link to open the DITA > Publishing preferences page (on page 275) where you can add that directory to the list. All the templates contained in your template directory will be displayed in the preview pane along with all the built-in templates.

Save Template As Button

You can use the Save template as button (at the bottom-left of the transformation dialog box) to export the currently selected template into a new template package that can be used as a starting point to create your own custom template (on page 1791). Clicking this button will open a template package configuration dialog box (on page 3109) that contains some options and displays the parameters that will be exported to your template package.

Template Errors

When the Templates tab is opened, all templates (built-in and custom) are loaded and validated. Specifically, certain elements in the template descriptor file are checked for validity. If errors are encountered that prevents the template from loading, the following message will be displayed toward the bottom of the dialog box:

⚠️ Some templates could not be loaded. More details

If you click the More details link, a window will open with more information about the encountered error. For example, it might offer a hint that the element is missing from the expected descriptor file structure.

Also, if a template could be loaded, but certain elements could not be found in the descriptor file, a warning icon (⚠️) will be displayed on the template’s image (in the Templates tab of the transformation dialog box). For example, this happens if a valid preview-image element cannot be found.

Sharing Publishing Template

To share a publishing template with others, following these steps:

1. Copy your template in a new folder.
2. Go to Options > Preferences > DITA > Publishing (on page 275) and add that new folder to the list.
3. Switch the option as the bottom of that preferences page to Project Options.
4. Share your project file (.xpr).

For more information about customizing publishing templates, watch our video demonstration:

https://www.youtube.com/embed/zNmXfKWXwO8
Template Package Configuration Dialog Box

The **Save template as** button (at the bottom-left of the transformation dialog box for **WebHelp Responsive** or **PDF - based on HTML5 & CSS** transformations) can be used to export the currently selected template into a new template package that can be used as a starting point to create your own custom template (on page 1791). The result will be a ZIP archive that contains a template descriptor file and other resources (such as CSS files) that were attached to the selected template.

Clicking the **Save template as** button opens a template package configuration dialog box contains the following options and components:

**Name**
- Required field used to specify the name for the new template. This will become the text value of the `<name>` element in the template descriptor file. This information is displayed as the name of the template in the transformation scenario dialog box.

**Description**
- Optional field used to specify a template description. This will become the text value of the `<description>` element in the template descriptor file. This information is displayed when the user hovers over the template in the transformation scenario dialog box.

**Parameter Table**
- This table displays the parameters that will be exported. Only certain relevant parameters are exported. The parameters and their values will be inserted in the `<parameters>` section of the template descriptor file. If any of the parameter values point to a file path that references a template resource (such as CSS files, custom HTML fragments, images), those resources will automatically be copied to the new template package and their references will be changed accordingly.

**Note:**
- Additional resources that are referenced in CSS files or other resources will not be copied to the new template package, so you will need to copy them manually and update their references in the template descriptor file.

**Include WebHelp Customization**
- The same publishing template package can contain both a WebHelp Responsive and PDF customization and you can use the same template in both types of transformations (DITA Map)
WebHelp Responsive (on page 1435) or DITA Map to PDF - based on HTML5 & CSS (on page 1449)). This option specifies that the custom template will include a WebHelp Responsive customization.

Include HTML Page Layout Files

For WebHelp Responsive customizations, select this option if you want to copy the default HTML Page Layout Files (on page 1622) into your template package. They are helpful if you want to change the structure of the generated HTML pages.

Include PDF Customization

The same publishing template package can contain both a WebHelp Responsive and PDF customization and you can use the same template in both types of transformations (DITA Map WebHelp Responsive (on page 1435) or DITA Map to PDF - based on HTML5 & CSS (on page 1449)). This option specifies that the custom template will include a PDF customization.

Save as

Use this field to specify the name and path of the ZIP file where the template will be saved.

Figure 790. Template Package Configuration Dialog Box
FO Processor Tab (DITA-OT Transformations)

When you create a new transformation scenario (on page 1464) or edit an existing one (on page 1545), a configuration dialog box is displayed that allows you to customize the transformation with various options in several tabs.

The **FO Processor** tab is available for DITA-OT transformations with a **PDF** output type.

This tab allows you to select an FO Processor to be used for the transformation.

![Figure 791. FO Processor Configuration Tab](image)

You can choose one of the following processors:

**Apache FOP**

The default processor that comes bundled with Oxygen XML Editor.

**XEP**

The RenderX XEP processor. If XEP is already installed, Oxygen XML Editor displays the detected installation path under the drop-down menu. XEP is considered installed if it was detected in one of the following sources:
• XEP was configured as an external FO Processor in the FO Processors option page (on page 261).
• The system property com.oxygenxml.xep.location was set to point to the XEP executable file for the platform (for example: xep.bat on Windows).
• XEP was installed in the DITA-OT-DIR/plugins/org.dita.pdf2/lib directory of the Oxygen XML Editor installation directory.

Antenna House

The Antenna House (AH Formatter) processor. If Antenna House is already installed, Oxygen XML Editor displays the detected installation path under the drop-down menu. Antenna House is considered installed if it was detected in one of the following sources:

• Environment variable set by Antenna House installation (the newest installation version will be used).
• Antenna House was added as an external FO Processor in the Oxygen XML Editor preferences pages.

To further customize the PDF output obtained from the Antenna House processor, follow these steps:

1. Edit the transformation scenario.
2. Open the Parameters tab (on page 3112).
3. Add the env.AXF_OPTS parameter and point to the Antenna House configuration file.

Related information
FO Processors Preferences (on page 261)
XSL-FO (Apache FOP) Processor for Generating PDF Output (on page 1526)

Parameters Tab (DITA-OT Transformations)

When you create a new transformation scenario (on page 1464) or edit an existing one (on page 1545), a configuration dialog box is displayed that allows you to customize the transformation with various options in several tabs.

The Parameters tab allows you to configure the parameters sent to the DITA-OT build file.

The table in this tab displays all the parameters that the DITA-OT documentation specifies as available for each chosen type of transformation (for example, XHTML or PDF), along with their description and current values. You can find more information about each parameter in the DITA-OT Documentation. You can also add, edit, and remove parameters, and you can use the text box to filter or search for a specific term in the entire parameters collection. Note that edited parameters are displayed with their name in bold.

Depending on the type of a parameter, its value can be one of the following:
- A simple text field for simple parameter values.
- A combo box with some predefined values.
- A file chooser and an editor variable (on page 323) selector to simplify setting a file path as the value of a parameter.

Note:
To input parameter values at runtime, use the ask editor variable (on page 325) in the Value column.

Below the table, the following actions are available for managing parameters:

New
Opens the Add Parameter dialog box that allows you to add a new parameter to the list. You can specify the Value of the parameter by using the Insert Editor Variables (on page 323) button or the Browse button. You can also use the Open in editor button to open the specified file in the editor panel.

Unset
Resets the selected parameter to its default value. Available only for edited parameters with set values.

Edit
Opens the Edit Parameter dialog box that allows you to change the value of the selected parameter or its description.

Delete
Removes the selected parameter from the list. It is available only for new parameters that have been added to the list.

Parameters Contributed by an Oxygen Publishing Template

Transformation parameters that are defined in an Oxygen Publishing Template (on page 1783) descriptor file are displayed in italics. After creating a publishing template (on page 1791) and adding it to the templates gallery (on page 1644), when you select the template in the Templates tab (on page 3106), the Parameters tab will automatically be updated to include the parameters defined in the template descriptor file.

Related Information:
DITA Open Toolkit Documentation

Feedback Tab (DITA-OT Transformations)

When you create a new transformation scenario (on page 1464) or edit an existing one (on page 1545), a configuration dialog box is displayed that allows you to customize the transformation with various options in several tabs.
The **Feedback** tab is for those who want to provide a way for users to offer feedback and ask questions in the published output and it is available for the **DITA Map WebHelp Responsive** transformation type. To add a comments component in the output, you need to use **Oxygen Feedback** to create a site configuration for the website where your WebHelp output is published and use this **Feedback** tab to instruct the transformation to install the comments component at the bottom of each WebHelp page.

When you create a site configuration in the **Oxygen Feedback** administration interface, an HTML fragment is generated during the final step of the creation process. You need to click the **Edit** button at the bottom-right of this tab to open a dialog box where you will paste the generated HTML fragment. The HTML fragment can also be set in an **Oxygen Publishing Template** (on page 1783), either as an **HTML fragment extension point** (on page 1613) or as a **transformation parameter** (on page 1612) (the `webhelp.fragment.feedback` parameter). If the fragment is specified in multiple places, the order of precedence (from highest to lowest) is:

- The fragment specified directly in the **Feedback** tab.
- The fragment specified in a publishing template as an HTML fragment extension point.
- The fragment specified in a publishing template as a transformation parameter.

**Filters Tab (DITA Transformations)**

When you create a new transformation scenario (on page 1464) or edit an existing one (on page 1545), a configuration dialog box is displayed that allows you to customize the transformation with various options in several tabs.

The **Filters** tab allows you to add filters to remove certain content elements from the generated output.
You can choose one of the following options to define filters:

**Use DITAVAL file**

If you already have a DITAVAL file associated with the DITA map (on page 3218), you can specify the file to be used when filtering content. You can specify the path by using the text field, its history drop-down, the Insert Editor Variables (on page 323) button, or the browsing actions in the Browse drop-down list. You can find out more about constructing a DITAVAL file in the DITA Documentation.

**Note:**

If a filter file is specified in the args.filter parameter (in the Parameters tab (on page 3112)), the filters are combined (neither file takes precedence over the other).

**Use profiling condition set**

Sets the profiling condition set (on page 3142) that will be applied to your transformation.

**Exclude from output all elements with any of the following attributes**

By using the New, Edit, or Delete buttons at the bottom of the pane, you can configure a list of attributes (name and value) to exclude all elements that contain any of these attributes from the output.
Note:
The colors and styles of the profiled content (on page 3147) settings are used for rendering it in Author mode but are not applied in the output.

Advanced Tab (DITA-OT Transformations)

When you create a new transformation scenario (on page 1464) or edit an existing one (on page 1545), a configuration dialog box is displayed that allows you to customize the transformation with various options in several tabs.

The Advanced tab allows you to specify advanced options for the transformation scenario.

You can specify the following options:

**Prefer using the "dita" command**

When selected, Oxygen XML Editor will attempt to use the dita.bat executable script (dita.sh for macOS and Linux) that is bundled with DITA-OT to run the transformation. If not selected,
the transformation will run as an ANT process. Also, when this option is selected, other options (Custom build file, Build target, Ant Home) become unavailable. This setting is checked by default in newly created DITA-OT transformation scenario.

Note:
Even when this option is selected, the dita.bat (dita for macOS and Linux) executable cannot be used in some cases. For example, if the DITA Map is published from a remote location or if the fix.external.refs parameter is enabled in the Parameters tab, the transformation is started as an ANT process instead of using the executable.

Custom build file

If you use a custom DITA-OT build file, you can specify the path to the customized build file. If empty, the build.xml file from the dita.dir parameter that is configured in the Parameters tab (on page 3112) is used. You can specify the path by using the text field, the Insert Editor Variables (on page 323) button, or the Browse button.

Build target

Optionally, you can specify a build target for the build file. If no target is specified, the default init target is used.

Additional Ant arguments

You can specify additional Ant-specific command-line arguments (such as -diagnostics).

Ant Home

You can choose between the default or custom Ant installation to run the transformation. The default path can be configured in the Ant preferences page (on page 265).

Java Home

You can choose between the default or custom Java installation to run the transformation. The default path is the Java installation that is used by Oxygen XML Editor.

Note:
It may be possible that the used Java version is incompatible with the DITA Open Toolkit engine. For example, DITA-OT 2.0 and newer requires Java 1.7 or newer. Thus, if you encounter related errors running the transformation, consider installing a Java VM that is supported by the DITA-OT publishing engine and using it in the Java Home text field.

JVM Arguments

This parameter allows you to set specific parameters for the Java Virtual Machine used by Ant. When performing a large transformation, you may want to increase the memory allocated to the Java Virtual Machine. This will help avoid Out of Memory error messages (OutOfMemoryError). For example, if it is set to -Xmx2g, the transformation process is allowed to use a maximum 2 gigabytes of memory. If you do not specify an -Xmx value in this field, by default the application
will use a maximum of 512 megabytes when used with the 32-bit Java Virtual Machine and one gigabyte with the 64-bit Java Virtual Machine.

**Libraries**

By default, Oxygen XML Editor adds libraries (as high priority) that are not transformation-dependent and also patches for certain DITA Open Toolkit bugs. You can use this button to specify additional libraries (*JAR* (on page 3220) files or additional class paths) to be used by the transformer.

**Tip:**

You can specify the path to the additional libraries using wildcards (for example, `${oxygenHome}/lib/*.jar`).

**Output Tab (DITA-OT Transformations)**

When you create a new transformation scenario (on page 1464) or edit an existing one (on page 1545), a configuration dialog box is displayed that allows you to customize the transformation with various options in several tabs.

The **Output** tab allows you to configure options that are related to the location where the output is generated.

**Figure 794. Output Settings Tab**
You can specify the following parameters:

**Base directory**

All the relative paths that appear as values in parameters are considered relative to the base directory. The default value is the directory where the transformed map is located. You can specify the path by using the text field, the ✑ Insert Editor Variables (on page 323) button, or the ✉️ Browse button.

**Temporary files directory**

This directory is used to store pre-processed temporary files until the final output is obtained. You can specify the path by using the text field, the ✑ Insert Editor Variables (on page 323) button, or the ✉️ Browse button.

**Output directory**

The folder where the content of the final output is stored. You can specify the path by using the text field, the ✑ Insert Editor Variables (on page 323) button, or the ✉️ Browse button.

**Note:**

If the DITA map (on page 3218) or topic is opened from a remote location or a ZIP file, the parameters must specify absolute paths.

**Open in Browser/System Application**

If selected, Oxygen XML Editor automatically opens the result of the transformation in a system application associated with the file type of the result (for example, in Windows PDF files are often opened in Acrobat Reader).

**Note:**

To set the web browser that is used for displaying HTML/XHTML pages, go to Options > Preferences > Global, and set it in the Default Internet browser field.

- **Output file** - When Open in Browser/System Application is selected, you can use this button to automatically open the default output file at the end of the transformation.
- **Other location** - When Open in Browser/System Application is selected, you can use this option to open the file specified in this field at the end of the transformation. You can specify the path by using the text field, the ✑ Insert Editor Variables (on page 323) button, or the ✉️ Browse button.

**Open in editor**

When this is option is selected, at the end of the transformation, the default output file is opened in a new editor panel with the appropriate built-in editor type (for example, if the result is an XML
file it is opened in the built-in XML editor, or if it is an XSL-FO file it is opened with the built-in FO editor).

At the bottom of the pane there is a link to the Console options (on page 275) preferences page that contains options to control the display of the console output received from the publishing engine.

Customizing DITA Transformations

You can customize the appearance of any of the output types by customizing the output transformations. There are several ways to do this:

- Most transformations are configurable by passing parameters to the transformation script. Oxygen XML Editor allows you to set parameters (on page 3112) on a transformation scenario and you can save and share them with others (on page 1555). You can also use the ${ask} editor variable (on page 325) in the Parameters tab to instruct Oxygen XML Editor to prompt you for a particular parameter whenever a transformation scenario is run. You can set up multiple transformation scenarios for a given output type, allowing you to maintain several customized transformation scenarios for multiple types of output configurations.

- If you want to customize an output in a way not supported by the built-in customization options, you can create a modified version of the transformation code and execute it using a custom build file (on page 3122). Sometimes the transformation code exports specific forms of extensions or customizations. You should consult the DITA Open Toolkit for the transformation type that you are interested in to see what customization options are supported. Oxygen XML Editor provides full editing and debugging support from XSLT and CSS stylesheets (on page 2102), which you can use to modify transformation code.

- You can also write your own transformation code (and execute it using a custom build file (on page 3122)) to produce a type of output not supported by the DITA Open Toolkit. Oxygen XML Editor provides a full source editing environment for developing such transformations. You can create Oxygen XML Editor transformation scenarios to run these scripts once they are complete.

There are also many other ways to customize specific types of output generated from DITA transformations:

- **WebHelp** - For information about customizing WebHelp output, see the WebHelp Output section (on page 1559).

- **PDF** - For information about customizing PDF output generated from DITA content, see XSL FO-based DITA to PDF Customization (on page 1986).

Publishing Customizations

Some customizations (usually for HTML-based output), can be made simply by creating a custom CSS and they do not involve modifying the DITA-OT engine in any way. Instead, most customizations involve adding a new plugin to the DITA-OT. Here are some best practices to follow before beginning your customization:
1. Copy the bundled DITA-OT folder (usually OXYGEN_INSTALL_DIR\frameworks\dita\DITA-OT3.x) to a location where you have full write access so that you have the ability to install new DITA-OT plugins (on page 3164).
2. Go to Options > Preferences > DITA, select Custom for the DITA Open Toolkit option and set the Location to be the path to the location where you copied the bundled DITA-OT folder. This will allow you to upgrade the version of Oxygen XML Editor at anytime without affecting the publishing system.
3. Share that external DITA-OT folder with the rest of the team. If you are using a repository (such as Subversion or Git), you can commit the entire modified DITA-OT publishing engine as part of your project. This will allow everyone in your team to use the official changes that you made. This will also allow you to set up some kind of automatic publishing system using an open-source integration server (such as Travis CI or Jenkins).

Customizing XHTML-based Output

XHTML-based output can be modified by using a custom CSS stylesheet to override various styles. If you edit an XHTML transformation scenario (on page 3104), there is a parameter called args.css that can be set to point to your custom CSS and a parameter called args.copy.css that as long as it is set to yes, the CSS is copied to the output folder.

You can also create plugins to customize the XHTML-based output by adding an extra XSLT stylesheet. For information, see: https://blog.oxygenxml.com/topics/creating-simple-dita-open-toolkit.html. A list with all DITA-OT XSLT extension points can be found here: http://www.dita-ot.org/dev/extension-points/plugin-extension-points-xslt-import.html.

Customizing WebHelp-based Output

The DITA-OT that comes bundled in Oxygen XML Editor includes specific plugins that provide the ability to publish DITA content to WebHelp Responsive (on page 1560) output.

For information about customizing WebHelp Responsive output, see Customizing WebHelp Responsive Output (on page 1642).

Customizing PDF-based Output

DITA to PDF output can be customized either by creating a PDF customization folder (in this case, the DITA-OT folder will not be modified at all) or by creating a PDF customization plugin. For information about customizing DITA to PDF output, see XSL FO-based DITA to PDF Customization (on page 1986).

There is also a book called DITA For Print that contains details about how to customize various aspects.

Customizing PDF Output with CSS

Oxygen XML Editor also includes a transformation scenario called DITA Map PDF - based on HTML5 & CSS (on page 3095) that is based on a DITA-OT CSS-based PDF Publishing plugin that allows you to convert DITA maps (on page 3218) to PDF using a CSS layout processor. For those who are familiar with CSS, this makes it very easy to style and customize the PDF output of your DITA projects without having to work with
**Using a Custom Build File**

You can use a *Custom Build File* to customize transformation scenarios.

To use a custom build file in a DITA-OT transformation, follow these steps:

1. Use the ✉️ **Configure Transformation Scenario(s)** action to open the **Configure Transformation Scenario(s)** dialog box *(on page 1548).*
2. Select the transformation scenario and click **Edit**.
3. Go to the **Advanced** *(on page 3116)* tab and change the **Custom build file** path to point to the custom build file.

As an example, if you want to call a custom script before running the DITA-OT, your custom build file would have the following content:

```xml
<project basedir="." default="dist">
    <!--The DITA-OT default build file-->
    <import file="build.xml"/>
    <target name="dist">
        <!-- You could run your script here -->
        <!----<exec></exec>-->
        <!-- Call the DITA-OT default target-->
        <antcall target="init"/>
    </target>
</project>
```

**Note:**

If you use the built-in Ant 1.9.8 build tool that comes bundled with Oxygen XML Editor, it is located in the `{OXYGEN_INSTALL_DIR}/tools/ant` directory. Any additional libraries for Ant must be copied to the Oxygen XML Editor Ant lib directory.

**Adding a Watermark in DITA Map to XHTML Output**

To add a watermark to the XHTML output of a *DITA map* *(on page 3218)* transformation, follow these steps:
1. Create a custom CSS stylesheet that includes the watermark image, as in the following example:

```css
body {
  background-image: url(MyWatermarkImage.png);
}
```

2. Edit a DITA Map XHTML transformation scenario and in the Parameters tab set the value of the `args.css` parameter as the path to your watermark image.

3. Set the value of the `args.copycss` parameter to `yes`.

4. Apply the transformation scenario.

5. Copy the watermark image in the output directory of the transformation scenario, next to the CSS file created in step 1.

Related Information:
Adding a Watermark to PDF Output (on page 1990)

Adding Syntax Highlights for Codeblocks in the Output

Syntax Highlighting makes it easier to read the semantics of the structured content by displaying each type of code (language) in different colors and fonts. The application provides the ability to add syntax highlights in codeblocks for DITA to PDF or HTML-based output through the use of the `@outputclass` attribute and a variety of predefined values are available.

To provide syntax highlighting in the codeblocks that appear in the output, add the `@outputclass` attribute on the `<codeblock>` element and set its value to one of the predefined language values. The Content Completion Assistant offers a list of the possible values when adding the `@outputclass` attribute in Text mode but there are also two very simple ways to set the value in Author mode:

- Select the `<codeblock>` element in the editor and in the Attributes view, click on the Value cell for the `@outputclass` attribute and select one of the predefined values (for example, `language-xml`).
- Select the `<codeblock>` element in the editor and use the Alt + Enter keyboard shortcut to open the in-place attributes editor window. Then select one of the predefined values from the Value drop-down menu.

The predefined values that can be selected are:

- language-bourne
- language-c
- language-cpp
- language-csharp
- language-css
- language-ini
- language-java
- language-javascript
- language-json
- language-lua
- language-perl
- language-php
- language-python
- language-ruby
- language-sql
- language-xml
- language-xquery

**Attention:**

It is recommended that you do not add inline elements in the codeblocks when using the `@outputclass` attribute, as it may lead to improper highlighting.

**Tip:**

Starting with version 24.0, the language values can also be set without using the `language-` prefix.

**Example:**

This codeblock:

```xml
<codeblock outputclass="language-xml">
  <p>This code is an example of how to use a coderef.</p>
  <codeblock><coderef href="MyExternalCode.xsl"/></codeblock>
</codeblock>
```

would result in this in the output:

```xml
<p>This code is an example of how to use a coderef.</p>
<codeblock><coderef href="MyExternalCode.xsl"/></codeblock>
```

**Publishing with a DITA-OT Project File**

The *DITA Open Toolkit* project file allows you to define all your DITA map input and filter pairs and to produce the desired output formats by applying the publishing engine over this single project file: [https://www.dita-ot.org/dev/topics/using-project-files.html](https://www.dita-ot.org/dev/topics/using-project-files.html).

Once a DITA-OT project file is opened in the application, two predefined publishing scenarios become available in the *Configure Transformation Scenario(s)* dialog box *(on page 1548):*
• **Publish DITA-OT Project (all deliverables)** - Runs the publishing engine and produces output for all deliverables defined in the project file.

• **Publish DITA-OT Project (select deliverable)** - Runs the publishing engine and produces output for only one deliverable specified by the end-user.

Some of the allowed transformation parameters that are relevant to the DITA-OT project file include:

- **project.file** - Specifies the path to the project file.
- **dita-ot.dir** - Specifies the directory where DITA-OT, used in transformation is installed.
- **additional.args** - Specifies the additional arguments used in transformation.
- **deliverable.id** - Specifies the id of the deliverable. This parameter is only available in the **Publish DITA-OT Project (select deliverable)** transformation.

**Related Information:**
DITA Open Toolkit Project *(on page 3171)*

### Dynamic Word/Excel/HTML/Markdown to DITA Converter

**Restriction:**
This feature works with no restrictions if the publishing process is done using Oxygen XML Editor/Author. However, if the publishing process is done from a command line, this feature requires an Oxygen Publishing Engine license.

The publishing engine has support to dynamically convert various types of non-DITA resources to DITA while publishing. To enable this support for a particular resource that is referenced in a DITA map, you must specify one of the following values for the `@format` attribute on the `<topicref>` element:

**Word to DITA (word-to-dita)**

_Microsoft Word_ documents that are referenced in the DITA map using the `word-to-dita` value for the `@format` attribute get dynamically converted to DITA topics during publishing. Image references and internal links are preserved.

**Example:**

```xml
<topicref href="sample.docx" format="word-to-dita"/>
```

**Excel to DITA (excel-to-dita)**

_Microsoft Excel_ documents that are referenced in the DITA map using the `excel-to-dita` format attribute get dynamically converted to DITA topics that contain one or more tables during publishing.

**Example:**

```xml
<topicref href="sample.xlsx" format="excel-to-dita"/>
```
HTML to DITA (html-to-dita)

HTML documents that are referenced in the DITA map using the html-to-dita format attribute get dynamically converted to DITA topics during publishing.

Example:

```xml
<topicref href="sample.html" format="html-to-dita"/>
```

Markdown to DITA (markdown)

Markdown documents that are referenced in the DITA map using the markdown format attribute get dynamically converted to DITA topics during publishing using the support for Markdown bundled with the publishing engine by default.

Example:

```xml
<topicref href="sample.md" format="markdown"/>
```

Markdown to DITA (markdown-to-dita)

Markdown documents that are referenced in the DITA map using the markdown-to-dita format attribute get dynamically converted to DITA topics during publishing using the special conversion plugin provided by Oxygen XML Editor. The markdown-to-dita format conversion is more flexible than the built-in markdown conversion, allowing the conversion of Markdown documents that do not have consistent heading levels.

Example:

```xml
<topicref href="sample.md" format="markdown-to-dita"/>
```

Troubleshooting DITA Transformation Problems

This section contains some topics to help you troubleshoot DITA transformation issues.

DITA Map Transformation Fails (Cannot Connect to External Location)

Problem

*DITA map (on page 3218)* transformation fails because it cannot connect to an external location.

Solution

The transformation is run as an external Ant process so you can continue using the application as the transformation unfolds. All output from the process appears in the DITA Transformation tab.

The HTTP proxy settings are used for the Ant transformation, so if the transformation fails because it cannot connect to an external location, you can check the *Proxy preferences page (on page 301)*
DITA Map WebHelp Transformation Fails (Duplicate Topic References Found)

Problem

DITA Map WebHelp transformation fails with a message that indicates duplicate topic references were found.

Cause

By default the WebHelp transformation uses the force-unique parameter set to true to force the transformation to create unique output files for each instance of a resource when a map contains multiple references to a single topic. However, there are cases when this feature does not work as expected and the duplicate topic references are not handled properly.

Solution

To solve this issue, you should manually set a unique @copy-to attribute on any duplicate topic reference that was not handled automatically by DITA-OT:

```xml
<map>
  ...
  <topicref href="../topics/MyTopic.dita"/>
  ...
  <topicref href="../topics/MyTopic.dita" copy-to="../topics/MyTopic-2.dita"/>
</map>
```

DITA-OT Transformation Takes a Long Time to Process

Problem

A DITA transformation takes an extremely long time to process (over an hour, for example).

Cause

Large delays in DITA-OT processing are usually caused by intensive disk operations, CPU usage, or connections to remote websites. The DITA-OT processing is very disk-intensive, each stage takes the entire content from the transformation temporary files folder, reads it, modifies it, and then writes it back.

Solution

There are several things you can try to troubleshoot this problem:

- If you are using a shared or remote drive, it is recommended to specify a local drive for the output and temporary files directory (edit the transformation scenario and in the Output tab, select a local directory for Temporary files directory and Output directory).
- If you want to test if the publishing has a problem downloading remote resources, you could disable the network adapter on the computer and then try to publish. The purpose is to see if the publishing finishes without any reported error about obtaining a certain HTTP resource.
• Using DTDs instead of XML Schemas is faster. This is because of a default transformation parameter called `args.grammar.cache` that only works for DTD-based DITA topics.

• You can increase the memory available to Oxygen XML Editor (on page 2856). Sometimes, just increasing the amount of memory available to the DITA-OT process may be enough to lower the time necessary for the publishing to run.

• You can enable some logging to help you determine which stage in the process is taking a long time. Edit the transformation scenario and in the Advanced tab, enter `logger org.apache.tools.ant.listener.ProfileLogger` in the Additional arguments field. Then go to Options > Preferences > DITA > Logging and select Always for the Show console output option.

• You could try disabling antivirus applications since the publishing process is very disk intensive and certain antivirus application might slow down the process.

• If the published DITA map is part of a larger DITA project with lots of maps and topics, references from topics in the current map to topics in other sub-projects might result in problems resolving those references. You could look in the output folder to see if the number of HTML documents match the number of DITA topics in your map.

DITA PDF Transformation Fails

Problem

The DITA to PDF transformation fails.

Cause

To generate the PDF output, Oxygen XML Editor uses the DITA Open Toolkit. This process sometimes results in errors. For information about some of the most common errors, see DITA PDF Processing Common Errors (on page 3129).

Solution

If your transformation fails, you can detect some of the problems that caused the errors by running the Validate and Check for Completeness action (on page 2941). Depending on the options you select when you run it, this action reports errors such as topics referenced in other topics but not in the DITA map (on page 3218), broken links, and missing external resources.

You can analyze the Results tab of the DITA transformation and search for messages that contain text similar to `[fop] [ERROR]`. If you encounter this type of error message, edit the transformation scenario you are using and set the clean.temp parameter to no and the retain.topic.fo parameter to yes. Run the transformation, go to the temporary directory of the transformation, open the topic.fo file and go to the line indicated by the error. Depending on the XSL FO context try to find the DITA topic that contains the text that generates the error.

If none of the above methods helps you, go to Help > About > Components > Frameworks and check what version of the DITA Open Toolkit you are using. Copy the whole output from the DITA-OT console output and either report the problem on the DITA User List or send it to support@oxygenxml.com.
DITA PDF Processing Common Errors

There are cases when the PDF processing fails when trying to publish DITA content to a PDF file. This topic lists some of the common problems and possible solutions.

**Problem: Cannot Save PDF**

The FO processor cannot save the PDF at the specified target. The console output contains messages like this:

```
[fop] [ERROR] Anttask - Error rendering fo file:
C:\samples\dita\temp\pdf\oxygen_dita_temp\topic.fo
<Failed to open C:\samples\dita\out\pdf\test.pdf>
Failed to open samples\dita\out\pdf\test.pdf
............
[fop] Caused by: java.io.FileNotFoundException:
C:\Users\default\Desktop\bev\out\pdf\test.pdf
(The process cannot access the file because it is being used by another process)
```

**Solution: Cannot Save PDF**

Such an error message usually means that the PDF file is already opened in a PDF reader application. The solution is to close the open PDF before running the transformation.

**Problem: Table Contains More Cells Than Defined in Colspec**

One of the DITA tables contains more cells in a table row than the defined number of `<colspec>` elements. The console output contains messages like this:

```
[fop] [ERROR] Anttask - Error rendering fo file:
D:\projects\eXml\samples\dita\flowers\temp\pdf\oxygen_dita_temp\topic.fo
<net.sf.saxon.trans.XPathException: org.apache.fop.fo.ValidationException:
The column-number or number of cells in the row overflows the number of
fo:table-columns specified for the table.
(See position 179:-1)>net.sf.saxon.trans.XPathException:
org.apache.fop.fo.ValidationException: The column-number or number of cells
in the row overflows the number of fo:table-columns specified for the table.
(See position 179:-1)
[fop] at org.apache.fop.tools.anttasks.FOPTaskStarter.renderInputHandler
(Fop.java:657)
[fop] at net.sf.saxon.event.ContentHandlerProxy.startContent
(ContentHandlerProxy.java:375)
```

Related Information:

How to Enable Debugging for FO Processor Transformations (on page 1529)
Solution: Table Contains More Cells Than Defined in Colspec

To resolve this issue, correct the `@colspec` attribute on the table that caused the issue. To locate the table that caused the issue:

1. Edit the transformation scenario and set the parameter `clean.temp` to `no`.
2. Run the transformation, open the `topic.fo` file in Oxygen XML Editor, and look in it at the line specified in the error message (See position 179:-1).
3. Look around that line in the XSL-FO file to find relevant text content that you can use (for example, with the Find/Replace in Files action in the DITA Maps Manager view (on page 2897)) to find the original DITA topic where the table was generated.

Problem: Broken Link

There is a broken link in the generated XSL-FO file. The PDF is generated but contains a link that is not working. The console output contains messages like this:

```
[fop] 1248 WARN [ main ] org.apache.fop.apps.FOUserAgent -
Page 6: Unresolved ID reference "unique_4_Connect_42_wrongID" found.
```

Solution: Broken Link

To resolve this issue:

1. Use the Validate and Check for Completeness action available in the DITA Maps Manager view (on page 2897) to find such problems.
2. If you publish to PDF using a DITAVAL filter, select the same DITAVAL file in the DITA Map Completeness Check dialog box.
3. If the Validate and Check for Completeness action does not discover any issues, edit the transformation scenario and set the `clean.temp` parameter to `no`.
4. Run the transformation, open the `topic.fo` file in Oxygen XML Editor, and search for the unresolved ID references (for example: `unique_4_Connect_42_wrongID`).
5. Look in the XSL-FO file to find relevant text content that you can use (for example, with the Find/Replace in Files action in the DITA Maps Manager view (on page 2897)) to find the original DITA topic where the table was generated.

Related Information:

How to Enable Debugging for FO Processor Transformations (on page 1529)
DITA to CHM Transformation Fails - Cannot Open File

**Problem**

The DITA to CHM transformation fails with the following error: [exec] HHC5010: Error: Cannot open "fileName.chm". Compilation stopped.

**Cause**

This error occurs when the CHM output file is opened and the transformation scenario cannot rewrite its content.

**Solution**

To solve this issue, close the CHM help file and run the transformation scenario again.

**Tip:**

It is a good practice to validate the DITA map (on page 3218) before executing the transformation scenario. To do so, run the Validate and Check for Completeness action (on page 2941). Depending on the selected options, it will report detected errors, such as topics referenced in other topics (but not in the DITA map), broken links, and missing external resources.

**Related Information:**

DITA Map CHM (Compiled HTML Help) Transformation (on page 3099)

DITA to CHM Transformation Fails - Compilation Failed

**Problem**

The DITA to CHM transformation fails with the following error: [exec] HHC5003: Error: Compilation failed while compiling fileName.

**Cause 1**

One possible cause for this error is that the processed file does not exist.

**Solution 1**

To solve this issue, fix the file reference before executing the transformation scenario again.

**Cause 2**

Another possible cause for this error is that the processed file has a name that contains space characters.

**Solution 2**

To solve the issue, remove any spacing from the file name and run the transformation scenario again.
Tip:

It is a good practice to validate the DITA map (on page 3218) before executing the transformation scenario. To do so, run the Validate and Check for Completeness action (on page 2941). Depending on the selected options, it will report detected errors, such as topics referenced in other topics (but not in the DITA map), broken links, and missing external resources.

Related Information:

DITA Map CHM (Compiled HTML Help) Transformation (on page 3099)

Solving DITA Transformation Errors

If a DITA transformation results in errors or warnings, the information is displayed in the message panel at the bottom of the editor. The information includes the severity, description of the problem, the name of the resource, and the path of the resource.

To help prevent and solve DITA transformation problems, follow these steps:

1. Validate the DITA map (on page 2941) by using the Validate and Check for Completeness action that is available on the DITA Maps Manager (on page 2897) toolbar and in the DITA Maps menu.
2. If this action results in validation errors, solve them prior to executing the transformation. Also, you should pay attention to the warning messages because they may identify problems in the transformation.
3. Run the DITA transformation scenario (on page 1490).
4. If the transformation results in errors or warnings, they are displayed in the Results panel (on page 548) at the bottom of the editor. The following information is presented to help you troubleshoot the problems:
   - **Severity** - The first column displays the following icons that indicate the severity of the problem:
     - **Informational** - The transformation encountered a condition of which you should be aware.
     - **Warning** - The transformation encountered a problem that should be corrected.
     - **Error** - The transformation encountered a more severe problem, and the output is affected or cannot be generated.
   - **Info** - Click the See More icon to open a web page that contains more details about DITA-OT error messages.
   - **Description** - A description of the problem.
   - **Resource** - The name of the transformation resource.
   - **System ID** - The path of the transformation resource.
5. Use this information or other resources from the online DITA-OT community to solve the transformation problems before re-executing the transformation scenario.
6. If you need to contact the Oxygen technical support team, they will need you to send the entire transformation scenario execution log. To obtain it:
a. Go to the Options > Preferences > DITA preferences page and set the Show console output option to Always.

b. Execute the transformation scenario again. The console output messages are displayed in the DITA-OT view.

c. Copy the entire log, save it in a text file, then send it to the Oxygen technical support team.

d. After your issue has been solved, go back to the Options > Preferences > DITA preferences page and set the Show console output option to When build fails.

Related Information:
Troubleshooting DITA Transformation Problems (on page 3126)

DITA Profiling / Conditional Text

DITA offers support for conditionally profiling content by using profiling attributes. With Oxygen XML Editor, you can define values for the DITA profiling attributes and they can be easily managed to filter content in the published output. You can switch between profile sets to see how the edited content looks like before publishing. The profiling configuration can also be shared between content authors through the project file and there is no need for coding or editing configuration files.

Oxygen XML Editor includes a Attributes and Condition Sets preferences page (on page 188) where you can create and manage profiling attributes and condition sets. Oxygen XML Editor also offers convenient support for customizing and controlling profiling attribute values with a subject scheme (on page 3151) or DITAVAL file (on page 3155).

Profiling Attributes

You can profile content elements or map elements by adding one or more of the default DITA profiling attributes (@product, @platform, @audience, @rev, @props, and @otherprops). You can also create your own custom profiling attributes and profiling condition sets. The profiling attributes may contain one or more tokens that represent conditions to be applied to the content when a publication is built.

For example, you could define a section of a topic that would only be included for a publication related to the Windows platform by adding the @platform profiling attribute:

```
<section platform="windows">
```

For information about creating and editing profiling attributes, see Creating and Editing Profiling Attributes in DITA (on page 3134) (for information about sharing them, see Sharing Profiling Attribute Configurations (on page 3137)).

Profiling Conditions

DITA allows you to conditionally profile parts of a topic so that certain parts of the topic are displayed when certain profiling conditions are set. Profiling conditions can be set both within topics and in maps. When set in a topic, they allow you to suppress an element (such as paragraph), step in a procedure, item in a list, or even
A phrase within a sentence. When set in a map, they allow you to suppress an entire topic or group of topics. You can then create a variety of publications from a single map by applying profiling conditions to the build.

For information about creating and editing condition sets, see Creating and Editing Profiling Condition Sets in DITA (on page 3140) (for information about sharing them, see Sharing Condition Set Configurations (on page 3142)).

Resources

For more information about DITA profiling, see the following resources:

- Webinar: Working with DITA in Oxygen - Basic Profiling and Reuse Strategies
- Webinar: Working with DITA in Oxygen - Advanced Profiling and Reuse Strategies

Creating and Editing Profiling Attributes in DITA

You can filter DITA content or the structure of a document by using profiling attributes or profiling condition sets (on page 3140).

Defining Profiling Attributes for DITA Content

To create or edit profiling attributes for filtering DITA content, follow these steps:

1. If you are creating a new attribute, make sure the attribute is already defined in the document DTD or schema before continuing with the procedure.

   **Tip:**
   For less technical users who do not want to create attribute specializations in DTD/XML Schema, you may want to use profiling attribute groups (on page 3149) instead (use an existing profiling attribute with sub-attributes).

2. Open the Preferences dialog box (Options > Preferences) (on page 124) and go to Editor > Edit modes > Author > Profiling / Conditional Text > Attributes and Condition Sets.

   **Information:**
   The Profiling Attributes section (on page 188) is used to define the attributes and their values.
   For DITA documents, the default attributes are included (@audience, @platform, @product, @props, @otherprops, and @rev), but if a Subject Scheme Map (on page 3151) is used for profiling your content, you will see the attributes defined in your subject scheme map instead.

3. To add new attributes and values, click the New button at the bottom of the Profiling Attributes table. To customize existing attributes and their values, select an attribute and click the Edit button.

   **Step Result:** In either case, this opens a Profiling Attribute configuration dialog box where you can define attributes that exist in your schema.
The following options are available in this dialog box:

**Document type**

Select the document type ([framework (on page 3219)])

- **Tip:**
  You can use the * or ? wildcards in this combo box. For example, DITA* would match any document type that starts with "DITA". You can also specify multiple document types by using commas to separate them.

**Attribute name**

The name of the profiling attribute.

**Display name**

This optional field is used for descriptive rendering in profiling dialog boxes.

**Attribute Values Table**

This table displays information about the values for the profiling attribute. You can configure them by using the buttons at the bottom of the table (New, Edit, Delete).

The columns are as follows:
Value - The attribute value. You can also define profiling attribute groups using the following format: ParentAttrValue(SubAttrValue1 SubAttrValue2). For more information, see Conditional Profiling Attribute Groups (on page 3149).

Label - You can specify a label for the attribute value that will be rendered as its name in various components in Author mode (Edit Profiling Attributes dialog box (on page 3137), Condition Set dialog box (on page 3140), Profiling tab in the Edit Properties dialog box (on page 2937), DITA Maps Manager (on page 2897)). If the Label is not specified, the Value will be used as its rendered name.

Description - A description for the attribute value that will be displayed in this table.

Single value
Select this option if you want the attribute to only accept a single value.

Multiple values separated by
Select this option if you want the attribute to accept multiple values, and you can choose the type of delimiter to use. You can choose between space, comma, and semicolon, or you can enter a custom delimiter in the text field. A custom delimiter must be supported by the specified document type. For example, the DITA document type only accepts spaces as delimiters for attribute values.

4. After defining or configuring the attributes and their values according to your needs, click OK to confirm your selections and close the Profiling Attributes configuration dialog box.

5. Click Apply to save the changes.

Result: You should see your changes in the Profiling Attribute table.

You can also use the Profiling Condition Sets section to apply more complex filters on your DITA content.

Adding Profiling Attribute Values Directly in a Document

You can add values directly to the existing profiling attributes in a document using the In-Place Attributes Editor (on page 610) in Author mode, the Attributes view (on page 627), or in the source code in Text mode. However, this just adds them to the document and does not change the conditional text configuration. If you invoke the Edit Profiling Attributes action (from the contextual menu in Author mode) on the new value, the Profiling Values Conflict dialog box will appear and it includes an Add these values to the configuration action that will automatically add the new value to the particular profiling attribute. It also includes an Edit the configuration action that opens the Attributes and Condition Sets preferences page (on page 188) where you can edit the profiling configuration.

Note:
If the Allow contributing extra profiling attribute values option (on page 189) is not selected in the Attributes and Condition Sets preferences page, the Profiling Values Conflict dialog box will never appear, so this automatically adding value not be possible.
Sharing Profiling Attribute Configurations

Your profiling configuration can be shared with other users through a project file. If you select Project Options (on page 3222) at the bottom of the Profiling/Conditional Text preferences page, your configuration is stored in the project file and can be shared with others. For instance, if your project file is saved on a version control system (such as SVN, CVS, or Source Safe) or in a shared folder, your team will have the same option configuration that you stored in the project file.

For more information about sharing project files, see Sharing a Project - Team Collaboration (on page 415).

Related Information:
- Applying Profiling Attributes in DITA (on page 3137)
- Creating and Editing Profiling Condition Sets in DITA (on page 3140)
- Applying Profiling Condition Sets in DITA (on page 3142)
- Showing and Filtering Profiled Content in DITA (on page 3144)
- Customizing Colors and Styles for Rendering Profiling in Author Mode (on page 3147)
- Conditional Profiling Attribute Groups (on page 3149)
- Filtering Profiling Values with a DITAVAL File (on page 3155)

Applying Profiling Attributes in DITA

Profiling attributes are applied on element nodes. You can apply profiling attributes on a text fragment (it will automatically be wrapped into a phrase-type element), on a single element, or on multiple elements at the same time. If there is no selection in your document, the profiling attributes are applied on the element at the cursor position.
You can apply defined DITA profiling attributes (on page 3134) as follows:

**DITA Topics**

To profile DITA topics, right-click a topic reference in the DITA Maps Manager (on page 2897), select **Edit Properties** from the contextual menu, go to the **Profiling** tab, and select the appropriate values.

**DITA Content**

To profile DITA content in **Author** mode, highlight the content and select **Edit Profiling Attributes** from the contextual menu and select the appropriate values in the **Edit Profiling Attributes** dialog box.

**DITA Elements**

To profile specific XML elements in **Author** mode, position the cursor inside the element, right-click, select **Edit Profiling Attributes** (you can also right-click the element in the breadcrumb (on page 602) or Outline (on page 539) view), and select the appropriate values in the **Edit Profiling Attributes** dialog box. You can also use the **Attributes view** (on page 627) to set the profiling attributes on the element at the current cursor position.

**Figure 797. Edit Profiling Attributes Dialog Box**

The profiling attributes, and their potential values, that appear in this dialog box depend on what has been configured in Oxygen XML Editor. If you have a large list of profiling attributes, you can use the text filter field to search for attributes or values, and you can expand or collapse attributes by using the **Expand All**/**Collapse All** buttons to the right of the text filter or the arrow button to the left of the profiling attribute name.
The attributes and values that appear in the dialog box are determined as follows:

- If your root map (on page 3223) references a DITA subject scheme map (on page 3224) that defines values for the profiling attributes (on page 3151), those values are used. Oxygen XML Editor collects all the profiling values from the subject scheme map that is referenced in the map that is currently opened in the DITA Maps Manager (on page 2897) (or set as the root map (on page 2914)). In the image above (on page 3138) (taken from the Oxygen XML Editor documentation project), you see values for eight products. They are the only values that are defined in the subject scheme map and thus, are the only ones that appear in the dialog box.

- If you have defined profiling attribute values (on page 3134) for the DITA document type in the Attributes and Condition Sets preferences page (on page 188) and you store them at project-level (on page 3222), those values are displayed in the dialog box.

- If you have defined profiling attribute values (on page 3134) for the DITA document type in the Attributes and Condition Sets preferences page (on page 188) and you store them at global-level (on page 3219), those values are displayed in the dialog box.

- If you have defined profiling attribute values (on page 3134) for the DITA document type in the Attributes and Condition Sets preferences page (on page 188), those values are displayed in the dialog box.

- Otherwise, a generic default set of profiling attributes and values are available.

The attribute names and values selected in the Edit Profiling Attributes dialog box are set on the elements contained in the profiled fragment. If you only select a fragment of content (rather than the entire element), this fragment is wrapped in phrase-type elements where the profiling attributes are set.

If the Show Profiling Attributes option (on page 679) (available in the Profiling / Conditional Text toolbar menu) is selected, a green border is painted around profiled text in the Author mode and all profiling attributes set on the current element are listed at the end of the highlighted block. To edit the attributes of a profiled fragment, click one of the listed attribute values. A form control pops up and allows you to add or remove attribute values.

**Figure 798. Profiling Attribute Value Form Control Pop Up**
Creating and Editing Profiling Condition Sets in DITA

Multiple profiling attributes can be aggregated into a profiling condition set that allows you to apply more complex filters on the document content. In DITA, profiling conditions can be set within both topics and in maps. When set in a topic, you can filter an element (such as paragraph), step in a procedure, item in a list, or even a phrase within a sentence. When set in a map, you can filter an entire topic or group of topics.

Creating Profiling Condition Sets

To create a new profiling condition set, follow these steps:

1. **Open the Preferences dialog box (Options > Preferences) (on page 124) and go to Editor > Edit modes > Author > Profiling/Conditional Text > Attributes and Condition Sets.**

   **Information:**
   
   The Profiling Condition Sets section (on page 189) is used to define condition sets.

2. **To add new condition set, click the New button at the bottom of the Profiling Condition Sets table. To customize existing condition sets, select an existing condition set and click the Edit button.**

   **Step Result:** In either case, this opens a Condition Set configuration dialog box where you can define attributes that exist in your schema.
The following options are available in this dialog box:

**Name**

The name of the new condition set.

**Document type**

Select the document type (framework (on page 3219)) that has profiling attributes defined.

**Use DITAVAL file**

For DITA projects, select this option if you want the *Profiling Condition Set* to reference a DITAVAL file (on page 3155). You can specify the path by using the text field, its history drop-down, the Insert Editor Variables (on page 323) button, or the browsing actions in the Browse drop-down list.

**Include the content matching the following conditions**

You can select this option to define the combination of attribute values for your condition set by selecting the appropriate checkboxes for the values you want to be included in this
particular condition set. If you have defined a lot of profiling attributes, you can use the filter text field to search for specific conditions.

Shortcut key

You can click the Choose button to open a dialog box that allows you to define a shortcut key for this particular condition set. You can then use that shortcut key anytime you want to select this condition set to filter content.

3. After defining or configuring the condition sets according to your needs, click OK to confirm your selections and close the Condition Set configuration dialog box.

4. Click Apply to save the condition set.

Sharing Condition Set Configurations

Your condition set configuration can be shared with other users through a project file. If you select Project Options (on page 3222) at the bottom of the Profiling/Conditional Text preferences page, your configuration is stored in the project file and can be shared with others. For instance, if your project file is saved on a version control system (such as SVN, CVS, or Source Safe) or in a shared folder, your team will have the same option configuration that you stored in the project file.

For more information about sharing project files, see Sharing a Project - Team Collaboration (on page 415).

Related Information:

- Applying Profiling Condition Sets in DITA (on page 3142)
- Creating and Editing Profiling Attributes in DITA (on page 3134)
- Applying Profiling Attributes in DITA (on page 3137)
- Showing and Filtering Profiled Content in DITA (on page 3144)
- Customizing Colors and Styles for Rendering Profiling in Author Mode (on page 3147)

Applying Profiling Condition Sets in DITA

All defined Profiling Condition Sets (on page 3140) are available as shortcuts in the Profiling / Conditional Text toolbar menu (on page 679). Select a menu entry to apply the condition set. The filtered content is then grayed-out in the Author mode, Outline view (on page 539), and DITA Maps Manager view (on page 2897). Your selection will also be used as the default condition set (on page 3159) in transformation scenarios (this can be changed in the Filters tab). An element is filtered-out when one of its attributes is part of the condition set and its value does not match any of the values covered by the condition set.

EXAMPLE:

Suppose that you have the following document:
Spray painting

Short Description: When paint is applied using a spray nozzle, it is referred to as spray painting.

Context:

The garage is a good place to spray paint.

Step 1
Move the car out of the garage to avoid getting paint on it. Audience [novice]

Step 2
Place newspaper, cardboard, or a drop-cloth on the garage floor. Audience [expert]

Step 3
Place the object to be painted on the covered area. Audience [expert] Other [prop2]

Step 4
Follow the directions on the paint can to paint the object. Audience [expert] Other [prop1]

Step 5
Let the paint dry thoroughly before you move the object. Audience [novice] Other [prop1]

If you apply the following condition set, it means that you want to filter out the content to only include content profiled with the expert value for the @audience attribute and content that has the prop1 value for the @other attribute.
This is how the document looks in **Author** mode after you apply the condition set:

```
Spray painting

Short Description: When paint is applied using a spray nozzle, it is referred to as spray painting.
Context:

The garage is a good place to spray paint.

Step 1
Move the car out of the garage to avoid getting paint on it. Audience [novice]

Step 2
Place newspaper, cardboard, or a drop-cloth on the garage floor. Audience [expert]

Step 3
Place the object to be painted on the covered area. Audience [expert] Other [prop2]

Step 4
Follow the directions on the paint can to paint the object. Audience [expert] Other [prop1]

Step 5
Let the paint dry thoroughly before you move the object. Audience [novice] Other [prop1]
```

Related Information:
- Creating and Editing Profiling Condition Sets in DITA *(on page 3140)*
- Creating and Editing Profiling Attributes in DITA *(on page 3134)*
- Applying Profiling Attributes in DITA *(on page 3137)*
- Showing and Filtering Profiled Content in DITA *(on page 3144)*
- Customizing Colors and Styles for Rendering Profiling in Author Mode *(on page 3147)*

**Showing and Filtering Profiled Content in DITA**

You can visualize the effect of profiling content by using the profiling tools in the **Profiling/Conditional Text** drop-down menu that is located on the **DITA Maps Manager** *(on page 2897)* toolbar and on the main toolbar. This drop-down menu includes the following filtering options:

**Show Profiling Colors and Styles**

Select this option to show colors and styles for profiled content in **Author** mode and the **DITA Maps Manager** *(on page 2897)*. You can configure the colors and styles or specify whether or not this option is selected by default in the **Profiling/Conditional Text > Colors and Styles** preferences page *(on page 190)*.

**Show Profiling Attributes**

Select this option to display the values of the profiling attributes at the end of profiled content in **Author** mode and next to the nodes in the **DITA Maps Manager** *(on page 2897)*. You can specify
whether or not this option is selected by default in the **Profiling/Conditional Text preferences page** (on page 187).

### Show Excluded Content

Controls whether the content filtered out by a particular condition set is hidden or grayed-out in **Author** mode, the **DITA Maps Manager** (on page 2897), and the **Outline** (on page 539) view. When this option is selected and a **condition set is selected in this drop-down menu** (on page 3145), the filtered content is grayed-out. If this option is not selected and a **condition set is selected in this drop-down menu** (on page 3145), the filtered content is hidden. You can specify whether or not this option is selected by default in the **Profiling/Conditional Text preferences page** (on page 188).

**Choose Condition Set (Available if more than 15 condition sets are defined)**

This option is available if you have more than 15 conditions sets defined. It opens a dialog box that makes it easier to find and select condition sets that are not displayed in this drop-down menu.

**List of Defined Condition Sets**

Up to 15 defined condition sets are listed and you can toggle each one of them on to filter the content in **Author** mode and the **DITA Maps Manager** (on page 2897) to only show content that will appear in the output for that particular condition set. If there are more than 15 defined condition sets, the rest of them can be accessed in the **More** submenu or by using the **Choose Condition Set** option (on page 3145) to access a dialog box that presents all of them.

**Profiling Settings**

Opens the **Attributes and Condition Sets preferences page** (on page 188) where you can add and edit profiling attributes and condition sets.

**Figure 800. Example: Profiling Controls in Author Mode**

<table>
<thead>
<tr>
<th>Mowing equipment needs regular checks and maintenance. Monthly, you should:</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Refill the oil:</td>
</tr>
<tr>
<td>○ Remove the oil fill cap;</td>
</tr>
<tr>
<td>○ Pour new oil gradually. Regularly check the dipstick to see if the oil level reached the maximum mark;</td>
</tr>
<tr>
<td><strong>product [Gasoline]</strong></td>
</tr>
<tr>
<td>• Sharpen the blades:</td>
</tr>
<tr>
<td>○ Clamp the blade to a vice or to the edge of a solid surface;</td>
</tr>
<tr>
<td>○ Using an electric grinder or a file, grind the length of the blade until it is sharp;</td>
</tr>
<tr>
<td>• Check the electric cable for any signs of wear. Replace it if worn; <strong>product [Electric]</strong></td>
</tr>
<tr>
<td>• Clean the mower's underside for debris; <strong>product [Electric, Gasoline]</strong></td>
</tr>
<tr>
<td>• Inspect the general state of the mower. Use a ratchet to tighten any loose bolts;</td>
</tr>
<tr>
<td>• Lubricate the gears of the manual lawn mower; <strong>product [Manual]</strong></td>
</tr>
</tbody>
</table>
If the **Show Profiling Attributes** option is selected, a green border is painted around profiled text in the **Author** mode. Also, all profiling attributes set on the current element are listed at the end of the highlighted block and in its tooltip message. To edit the attributes of a profiled fragment, click one of the listed attribute values. A form control pops up and allows you to add or remove attribute values.

![Figure 801. Profiling Attribute Value Form Control Pop Up](image)

Also, the following icons are used to mark profiled and non-profiled topics in the **DITA Maps Manager**:

- ■ - The topic reference contains profiling attributes.
- □ - The topic reference inherits profiling attributes from its ancestors.
- ☑ - The topic reference contains and inherits profiling attributes.
- -(hyphen) - The topic reference neither contains nor inherits profiling attributes.
Customizing Colors and Styles for Rendering Profiling in Author Mode

By applying profiling colors and styles, you can mark profiled content in Author mode and the DITA Maps Manager (on page 2897) so that you can instantly spot differences between multiple variants of the output. This allows you to preview the content that will go into the published output. The excluded text is grayed-out or hidden in Author mode and excluded nodes are grayed-out or hidden in the DITA Maps Manager (on page 2897).
Choosing the right style for a specific profiling attribute is a matter of personal taste, but be aware of the following:

- If the same block of text is profiled with two or more profiling attributes, their associated styles combine. Depending on the styling, this might result in an excessively styled content that may prove difficult to read or work with.
- It is recommended that you only profile the differences. There is no need to profile common content, since excessive profiling can visually pollute the document.
- A mnemonic associated with a style will help you instantly spot differences in the types of content.

**Styling Profiling Attribute Values**

To set colors and styles for profiling attribute values, follow these steps:

1. Select the **Show Profiling Colors and Styles** option (on page 679) from the **Profiling / Conditional Text** toolbar drop-down menu.
2. Select **Profiling Settings** (on page 680) from the **Profiling / Conditional Text** toolbar drop-down menu. This is a shortcut to the **Attributes and Condition Sets** preferences page (on page 188).
3. Go to the **Colors and Styles** preferences page (on page 190) to configure the colors and styling for the profiling attributes.
4. Go to the **Attributes** preferences page (on page 191) to configure how you want the profiling attributes to appear in Oxygen XML Editor.

**Result:** The styling is now applied in the **Author** editing mode, the **Outline view** (on page 539), and in the **DITA Maps Manager view** (on page 2897). Also, to help you more easily identify the profiling you want to apply in...
the current context, the styling is applied in the **Edit Profiling Attributes** dialog box (on page 670) and in the inline form control pop-up that allows you to quickly set the profiling attributes.

![Figure 804. Profiling Attribute Value Form Control Pop Up](image)

Alternate Method with a DITAVAL File: If you are using a DITAVAL filter file to control the filtering of profiled content in DITA topics, you can use a flag filter to define the colors and styles that will be used when rendering the profiling. For detailed information about this alternate method, see the procedure in the **Styling the Rendering of Profiled Content Using a DITAVAL File (on page 3158)** topic.

**Related Information:**

- Creating and Editing Profiling Condition Sets in DITA (on page 3140)
- Applying Profiling Attributes in DITA (on page 3137)
- Creating and Editing Profiling Attributes in DITA (on page 3134)
- Applying Profiling Condition Sets in DITA (on page 3142)
- Showing and Filtering Profiled Content in DITA (on page 3144)

### Conditional Profiling Attribute Groups

**Overview**

Conditional processing attributes can be specified using grouped values. Groups organize the attributes into subcategories. This is intended to support situations where an attribute applies to multiple specialized subcategories. For example, suppose a company needs to filter content for several internal teams (operations and support) and they use the [@audience](#) attribute with the values ops and support, but the Support team has several levels of personnel (L1, L2, and L3). They could use a group to define the levels (L1, L2, and L3) as subcategories for the support value. Using groups for these subcategories allows each category to be processed independently.

A major advantage is that you do not need to add new profiling attributes using a DTD specialization. You can re-use existing DITA profiling attributes (such as @product, @audience, @otherprops) and specify multiple attribute subcategories.
Creating a Conditional Profiling Attribute Group

To create a group in Oxygen XML Editor:

1. Open the Preferences dialog box (Options > Preferences) (on page 124) and go to Editor > Edit modes > Author > Profiling / Conditional Text > Attributes and Condition sets.

2. To add new attributes and values, click the New button at the bottom of the Profiling Attributes table. To customize existing attributes and their values, select an attribute and click the Edit button.

   **Step Result:** In either case, this opens a Profiling Attribute configuration dialog box where you can define attributes that exist in your schema.

3. Specify the appropriate values for the Document type, Attribute name, and Display name.

   For information about the Profiling Attribute configuration dialog box, see Defining Profiling Attributes for DITA Content (on page 3134).

4. Click the New button at the bottom of the attribute values table.

5. In the Value field of the resulting dialog box, define groups using the following format:

   ParentAttrValue(SubAttrValue1 SubAttrValue2). For example:

   ```
   support(L1 L2 L3)
   ```

6. Click OK and Apply to save and apply the changes.

Using Conditional Groups in Conjunction with a DITAVAL File

You can use groups to customize a hierarchy of profiling attribute values and then use it in conjunction with a DITAVAL file to filter or flag (on page 3155) the values. For example, suppose the company described in the example in the Overview section (on page 3149) needed to generate content for the Support team, but only for L1 and L2 support personnel. The DITAVAL file could look like this:

```
<val>
  <prop action="include" att="support" val="L1"/>
  <prop action="include" att="support" val="L2"/>
  <prop action="exclude" att="support" val="L3"/>
</val>
```

That DITAVAL file could then be used for a condition set (on page 3140) to filter content in Author mode or during the transformation stage to filter content in the output (on page 3159) and content profiled with the L1 and L2 values would be included while content with the L3 value would be excluded.

This example company could also have another DITAVAL file for filtering out all content profiled for any of the three subcategories (L1, L2, L3) by simply excluding the support value (since L1, L2, and L3 are subcategories of it).
Customizing Profiling Values with a Subject Scheme Map

Overview

A subject scheme map (on page 3224) allows you to create and manage custom profiling values in DITA documents without having to write a DITA specialization. Ultimately, this allows you to filter and flag content in Author mode or in transformed output.

Subject scheme maps use key definitions to define a collection of profiling values. You can also use subject scheme maps to filter out (reject) the values for certain attributes so that you only see the attributes or values that you want to use in Author mode or the transformed output.

The highest level of map (main DITA map (on page 3223)) that uses the set of profiling values must reference the subject scheme map where the profiling values are defined and the @type attribute needs to be set to subjectScheme for the reference, as in the following example:

```xml
<topicref href="test.ditamap" format="ditamap" type="subjectScheme"/>
```

Advantages of Using a Subject Scheme Map

The advantages of using a subject scheme to control profiling attribute values include:

- You can create a hierarchy of profiling attribute values and use a DITAVAL file to filter or flag the tree of values.
- You can share the subject scheme files with others without having to share preferences or the entire project.
- The subject scheme offers validation support so you are notified if an undefined value is used.

Creating a Subject Scheme Map

To create and configure a subject scheme map, follow this procedure:
1. Use the **New Document wizard** *(on page 368)* to create a new **Subject Scheme** document *(Framework templates > DITA Map > map > Subject Scheme)*.

2. Use the controls in **Author** mode to define the hierarchical tree of values for your *subject scheme* *(see the Author mode example below *(on page 3153))* or switch to **Text** mode and define it there if you prefer *(see the Text mode example below *(on page 3154))*.

**Note:**
The pre-defined subject scheme template includes Navigation Titles *(<navtitle> element)*. This element is not required, but if you use it, the text that you enter for the *(<navtitle>)* will be used instead of the name of the value) in the various places where profiling attributes are presented in Oxygen XML Editor *(on page 3155)*. An example of when this might be helpful is if you want to use abbreviations for the name of a value, but you want to see its full name in Oxygen XML Editor.

3. Bind the particular attribute to the key you define for the tree of values using the **<attributedef>** and **<subjectdef>** elements inside the **<enumerationdef>** element. Notice that in the examples below *(on page 3153)*, the *audience* attribute is bound to the *audienceKey* value.

**Tip:**
By default, attributes can accept multiple values, but you can use `outputclass="single_value"` to specify that a certain attribute only accepts a single value at a time and the attribute will be presented in Oxygen XML Editor with radio buttons instead of checkboxes. For example:

```xml
<enumerationdef outputclass="single_value">
  <attributedef name="audience"/>
  <subjectdef keyref="audienceKey"/>
</enumerationdef>
```

You can also define a specific set of possible attribute values for a specific attribute name that is set on a specific element name. For example, you can define a specific set of *@outputclass* attribute values only for the *<image>* element:

```xml
<enumerationdef>
  <elementdef name="image"/>
  <attributedef name="outputclass"/>
  <subjectdef keyref="imgOutputClassValuesKey"/>
</enumerationdef>
```

4. If you want to filter out (reject) values for certain attributes, bind the attributes to a blank value (as you see for the *props* and *otherprops* attributes in the examples below *(on page 3153)*). This means that those attributes will not appear in the various places where profiling attributes are presented in Oxygen XML Editor *(on page 3155)*.

5. Save your *subject scheme* file.
6. Reference your subject scheme in the highest level of map (main DITA map (on page 3223)) that will use the set of profiling values and set its type to subjectScheme. The easiest way to do this is:
   a. With your subject scheme file opened in the editor, go to the DITA Maps Manager view, right-click the main DITA map, and select Append Child > Reference to the currently edited file.
   b. In the Insert Topic Reference dialog box, go to the Attributes tab and in the Type field, enter or select subjectScheme.
   c. Click the Insert and Close button and save your main DITA map.

Using a Subject Scheme in Conjunction with a DITAVAL File

You can use a subject scheme to customize a hierarchy of profiling attribute values and then use it in conjunction with a DITAVAL file to filter or flag (on page 3155) the entire tree of values. For example, suppose one of the values for the audience attribute in a hierarchical subject scheme is surgeon and it has two subordinate values of neuro-surgeon and plastic-surgeon (see the examples below (on page 3153)). You could create a DITAVAL file with the following content:

```xml
<val>
  <prop action="exclude" att="audience" val="surgeon"/>
</val>
```

That DITAVAL file could then be used for a condition set (on page 3140) to filter content in Author mode or during the transformation stage to filter content in the output (on page 3159) and the neuro-surgeon and plastic-surgeon values would be excluded by the filter since the subject scheme defines them as subordinate values of the surgeon value.

Example: Subject Scheme Map that Defines Custom Values for the Audience Attribute

This example uses typical audience values for medical personnel (therapist, oncologist, physicist, radiologist, surgeon, and so on). The audience attribute is bound to the audienceKey value (which defines the tree of values). You can also see that it filters out all possible values for other attributes (props and otherprops) so that they won't be available in the various places where profiling attributes are presented in Oxygen XML Editor (on page 3155).

Example using Author mode controls:
Figure 805. Subject Scheme Author Mode Controls

Subject Scheme
A scheme that defines audience user values

<table>
<thead>
<tr>
<th>audienceKey</th>
</tr>
</thead>
<tbody>
<tr>
<td>therapist</td>
</tr>
<tr>
<td>oncologist</td>
</tr>
<tr>
<td>physicist</td>
</tr>
<tr>
<td>radiologist</td>
</tr>
<tr>
<td>surgeon</td>
</tr>
</tbody>
</table>

|  neuro-surgeon |
|  plastic-surgeon |

Binding the audience attribute to the values defined in the key

<table>
<thead>
<tr>
<th>Binding</th>
</tr>
</thead>
<tbody>
<tr>
<td>bind attribute audience to audienceKey</td>
</tr>
</tbody>
</table>

Reject all possible values for other profiling attributes

<table>
<thead>
<tr>
<th>Binding</th>
</tr>
</thead>
<tbody>
<tr>
<td>bind attribute props to</td>
</tr>
</tbody>
</table>

| bind attribute otherprops to |

Example code in Text mode:

```xml
<subjectScheme>
  <!-- A scheme that defines audience user values -->
  <subjectdef keys="audienceKey">
    <subjectdef keys="therapist"/>
    <subjectdef keys="oncologist"/>
    <subjectdef keys="physicist"/>
    <subjectdef keys="radiologist"/>
    <subjectdef keys="surgeon">
      <subjectdef keys="neuro-surgeon"/>
      <subjectdef keys="plastic-surgeon"/>
    </subjectdef>
  </subjectdef>
</subjectScheme>
```
<!-- Binding the audience attribute to the values defined in the key -->
<enumerationdef>
  <attributedef name="audience"/>
  <subjectdef keyref="audienceKey"/>
</enumerationdef>

<!-- Reject all possible values for other profiling attributes -->
<enumerationdef>
  <attributedef name="props"/>
  <subjectdef/>
</enumerationdef>
<enumerationdef>
  <attributedef name="otherprops"/>
  <subjectdef/>
</enumerationdef>
</subjectScheme>

Where the Profiling Attributes are Available in Oxygen XML Editor

When you edit a DITA topic in the **Text** or **Author** mode, Oxygen XML Editor collects all the profiling values from the *subject scheme map* (on page 3224) that is referenced in the map that is currently opened in the **DITA Maps Manager** (on page 2897) (or set as the *root map* (on page 2914)). The values of profiling attributes defined in a *Subject Scheme Map* are available in the following places in Oxygen XML Editor (regardless of their mapping in the **Profiling/Conditional Text** preferences page (on page 187)):

- The **Profiling** tab of the **Edit Properties** dialog box (on page 2937).
- The **Edit Profiling Attribute** dialog box (on page 3138).
- The inline profiling controls in **Author** mode (on page 3145).
- The proposals for the attribute values in the **Content Completion Assistant** (on page 3217).

For more information about using a DITA subject scheme map, watch our video demonstration:

https://www.youtube.com/embed/RgkVRg6k6zo

**Related Information:**
- Filtering Profiling Values with a DITAVAL File (on page 3155)
- DITA 1.3 Specifications: Subject Scheme Maps

Filtering Profiling Values with a DITAVAL File

You can use a DITAVAL filter file to control the filtering or flagging of profiled content or to identify which values are to be used for conditional processing during a particular output.
DITAVAL Filtering Use-Case

Suppose that a medical publication uses the audience profiling attribute to profile the content for the following types of users: therapist, physician, and surgeon. Suppose that in the output, you want to exclude any content that is profiled as surgeon value for the @audience attribute.

You could use a DITAVAL filter file to exclude anything that is profiled as surgeon:

```xml
<val>
  <prop action="exclude" att="audience" val="surgeon"/>
</val>
```

If you then transform the main DITA map (on page 3218) and specify the DITAVAL filter file in the transformation scenario, the output will exclude anything that is profiled as surgeon).

DITAVAL Filter File Editor in Author Mode

The Author editing mode in Oxygen XML Editor offer a simple and intuitive editor for creating or modifying DITAVAL files. It provides a series of drop-down menus and text fields that allow you to easily define the filters.

![DITAVAL Filter file in Author Mode](image)

Use the +... button to display a drop-down list that contains elements that you can add at that particular location in the DITAVAL file. Clicking this button at the top (next to the DITAVAL FILTER File title, allows you to insert the following elements:
- **Style Conflict** - Inserts a `<style-conflict>` element that declares behavior to be used when one or more flagging methods collide on a single content element. You can use the simple drop-down menus to select values for the `@foreground-conflict-color` and `@background-conflict-color` attributes.

- **Filter** - Inserts a `<prop>` element that identifies an attribute to apply a filtering action on. The possible actions that you can select are `include`, `exclude`, `passthrough`, and `flag`. If you select the `flag` action, you can use the drop-down menus to select values for the `@style`, `@color`, and `@background` attributes.

- **Flag** - Inserts a `<revprop>` element that identifies a value in the `@rev` attribute that should be flagged in some manner. The allowed actions are `include`, `passthrough`, and `flag`. If you select the `flag` action, you can use the drop-down menus to select values for the `@style`, `@color`, `@background`, and `@changebar` attributes.

See the DITAVAL Element Specifications for more details about the allowed filters and flags.

### How to Create a DITAVAL Filter File

To create a DITAVAL filter file, follow these steps:

1. Go to **File > New**.
2. Scroll to the Framework templates > DITA folder.
3. Select the Filter template file and click **Create**.
4. Define your filters in the DITAVAL file (in Text or Author mode).
5. Save the DITAVAL file.

**Result:** The DITAVAL filter file can now be used for all of the following:

- To apply a reference to the DITAVAL file in a Profiling Condition Set using the **Use DITAVAL File** option in the Condition Set configuration dialog box (on page 3140).
- You can use the Import from DITAVAL option in the Attributes and Condition Sets preferences page (on page 188) to use the DITAVAL file to define profiling attributes.
- You can use the DITAVAL file to apply the filters to the output by specifying the DITAVAL file in the transformation scenario (on page 3115).
- You can use the filter file in the DITA Map Completeness Check dialog box (on page 2942) when validating your DITA map (on page 3218).
- DITAVAL files are also used when working with the DITA 1.3 Branch Filtering mechanism. For more details, see: Working with DITA 1.3 Branch Filtering (on page 3060).
- You can define the colors and styles to be used for rendering profiled condition sets (on page 3158) in Author mode and the DITA Maps Manager (on page 2897) view by using a Flag filter in the DITAVAL file.

### Related information

- DITAVAL Element Specifications
- Working with DITA 1.3 Branch Filtering (on page 3060)
- Customizing Profiling Values with a Subject Scheme Map (on page 3151)
Styling the Rendering of Profiled Content Using a DITAVAL File

If you are using a DITAVAL filter file to control the filtering of profiled content, you can define the colors and styles to be used for rendering profiled condition sets in Author mode and the DITA Maps Manager (on page 2897) by defining the styles in a flag filter that is set in a DITAVAL filter file.

How to Define a Flag for a Condition Set in a DITAVAL Filter File

To define the colors and styles to be used for rendering profiled condition sets by using a flag filter in a DITAVAL filter file, follow these steps:

1. Create or edit your DITAVAL file (on page 3157) to define your profiling condition set (on page 3140).
2. In Author mode, define the filters for your condition set (on page 3156).
3. Select Flag from the drop-down menu on in a particular Filter or Flag Revision to present additional drop-down menus that allow you to configure the colors and styles for the particular condition set.
4. Save the DITAVAL file.

Result: Test your changes by opening profiled content in Author mode or the DITA Maps Manager (on page 2897) and use the options in the Profiling / Conditional Text drop-down menu to see how the changes in your DITAVAL flag are rendered.

EXAMPLE:

Using a Flag on a Filter to define the styling for a condition set like this:

```
DITAVAL Filter file

Filter exclude ∨ @ product ∨ = editor:eclipse
Filter flag ∨ @ product ∨ = editor
    style ∨ color ∨ background
        bold ∨ marcon ∨ silver
```

will render the styling of the profiled content in Author mode to look like this:

```
Shortcut Keys - Many of the shortcut keys
Markdown editor: product [editor]
```

and will render the styling in the DITA Maps Manager view (on page 2897) to look like this:
Publishing Profiled DITA Content

You can create a variety of publications or versions of your documentation from a single map by applying profiling conditions to the build.

Oxygen XML Editor includes preconfigured transformation scenarios for DITA. By default, these scenarios take the current profiling condition set (on page 3142) into account during the transformation, as defined in the Filters tab (on page 3114) when creating a DITA transformation (on page 1490). You can also specify a DITAVAL file (on page 3155) that defines filters for your profiled content.

Figure 807. Profiling Option in the Filters Tab (DITA-OT Transformations)

Conditional Processing to Generate Multiple Deliverables

By default, the content of most elements is included in all output media. Within maps and topics, elements can specify the delivery targets to which they apply.

Within maps, topic references can use the @deliveryTarget attribute to indicate the delivery targets to which they apply. Within topics, most elements can use the @deliveryTarget attribute to indicate the delivery targets.

If a referenced topic should be excluded from all output formats, set the @processing-role attribute to resource-only instead of using the @deliveryTarget attribute. Content within that topic can still be referenced for display in other locations.

@deliveryTarget

The intended delivery target of the content, for example html, pdf, or epub. This attribute is a replacement for the now deprecated @print attribute.
The `<deliveryTarget>` attribute is specialized from the `<props>` attribute. It is defined in the `deliveryTargetAttDomain`, which is integrated into all OASIS-provided document-type shells. If this domain is not integrated into a given document-type shell, the `<deliveryTarget>` attribute will not be available.

The `<deliveryTarget>` attribute is processed the same way as any other conditional processing attribute. For example, `<topicref deliveryTarget="html5 epub" href="example.dita"/>` uses two values for `<deliveryTarget>`. A conditional processing profile can then set rules for `<deliveryTarget>` that determine whether the topic is included or excluded when the map is rendered as HTML5 or EPUB.

**DITA Open Toolkit Support**

The *DITA Open Toolkit* is an open-source publishing engine that can generate various output formats (for example, HTML, PDF, CHM) from DITA content. Oxygen XML Editor includes support for the DITA Open Toolkit. This section includes information about how to install and create a DITA-OT plugin (on page 3222), and how to use an external instance of the DITA Open Toolkit.

**Related Information:**

- DITA Open Toolkit Documentation

**DITA-OT Plugins**

The architecture of the DITA Open Toolkit publishing engine is plugin-based. A plugin can add support for publishing DITA content as a new format or for customizing an existing output format. The DITA Open Toolkit bundled with Oxygen XML Editor already has lots of plugins pre-installed but you can also install additional plugins (on page 3222) or create your own.

**Creating a DITA-OT Plugin**

Oxygen XML Editor provides the ability to install additional DITA Open Toolkit plugins (on page 3164) that can be found from various sources (for example, Oxygen's public GitHub repository includes some DITA-OT plugins). It is also possible to create your own plugin.

To create a DITA-OT plugin, follow these steps:

1. Create a new folder in the `plugins` folder located in your DITA-OT directory (for example, if you are using DITA 3.7.0, the path would look like this: `[OXYGEN_INSTALL_DIR]/frameworks/dita/DITA-OT3.x/plugins/MyNewPlugin`).
2. Create a `plugin.xml` file in that same folder. This file will contain the extension points for the plugin. For example, references to the XSLT stylesheets that will be used to style the output.
Note:
You can easily create this file by using the DITA-OT Plugin new document template that is included in Oxygen XML Editor (from the New document wizard (on page 368) you can find this template in Framework templates > DITA > plugin.

Example:

```xml
<plugin id="org.metadita.specialization.music">
  <feature extension="dita.specialization.catalog.relative"
    file="catalog-dita.xml"/>
  <feature extension="dita.xsl.xhtml" file="xsl/music2xhtml.xsl"/>
  <feature extension="dita.xsl.html5" file="xsl/music2xhtml.xsl"/>
</plugin>
```

Tip:
Oxygen XML Editor includes special editing support when adding extension points in the plugin.xml file. If you place the cursor in the value of the @extension attribute and press Ctrl+Space, a list of possible extension points is presented with links to the DITA-OT documentation. For more information about extension points that are available to use in the plugin.xml file, see: http://www.dita-ot.org/dev/extension-points/extension-points-by-plugin.html.

3. Install the newly created DITA-OT plugin (on page 3164) by running the built-in transformation scenario called Run DITA-OT Integrator (on page 3102) from the Apply Transformation Scenario(s) (on page 1433) or Configure Transformation Scenario(s) dialog box (on page 1548).

Note:
If the integrator is not visible, select the Show all scenarios option in the Settings dropdown menu.

You can share your new plugin with other users who have the same DITA-OT distribution by sending them your newly created folder along with the installation instructions (on page 3164).

Related Information:
DITA Open Toolkit Documentation
Defining the Transformation Type and Allowed Parameters in a DITA-OT Plugin (on page 3165)

Example: Creating a DITA-OT Plugin for Embedding Video Resources

To offer a detailed example of the steps required to create a new DITA Open Toolkit plugin (on page 3222), this topic uses an example of creating an XSLT customization plugin that provides support for referencing
video and audio content using the DITA `<object>` element and then publishing to HTML and PDF output formats. This plugin (`com.oxygenxml.media`) is available in the DITA Open Toolkit distribution that comes bundled with the latest version of Oxygen XML Editor, but these instructions show you how you could create it if it were not included.

The following procedure is meant to help you with creating the plugin:

1. Create a folder for your plugin in the DITA-OT plugins folder (`DITA-OT-DIR/plugins/`).
2. Create a plugin.xml file (in the same plugin folder) that contains the extension points of the plugin.

   **Note:**
   You can easily create this file by using the DITA-OT Plugin template that is included in Oxygen XML Editor (from the New document wizard (on page 368) you can find this template in Framework templates > DITA > plugin).

   **Example: Media Plugin File**

   ```xml
   <plugin id="com.oxygenxml.media">
      <feature extension="package.support.name" value="Oxygen XML Editor Support"/>
      <feature extension="package.support.email" value="support@oxygenxml.com"/>
      <feature extension="package.version" value="21.0"/>
      <feature extension="dita.xsl.xhtml" value="xhtmlMedia.xsl" type="file"/>
      <feature extension="dita.xsl.xslfo" value="pdfMedia.xsl" type="file"/>
   </plugin>
   ``

   The most important extensions in it are the references to the XSLT stylesheets that will be used to style the HTML and PDF outputs.

   You can find other DITA-OT plugin extension points here: [http://www.dita-ot.org/dev/extension-points/extension-points-by-plugin.html](http://www.dita-ot.org/dev/extension-points/extension-points-by-plugin.html).

3. Create an XSLT stylesheet to customize the output types. In this example, to customize the HTML output, it is necessary to create an XSLT stylesheet called `xhtmlMedia.xsl` (in the same plugin folder).

   **Tip:**
   You can use the Find/Replace in Files action (on page 436) to find an XSLT stylesheet with content that is similar to the desired output and use it as a template to overwrite parts of your stylesheet. For example, suppose you want to overwrite HTML content produced from a DITA codeblock element. Since a DITA `<object>` element has the `@class` attribute value `- topic/object`, you can take part of the class attribute value (`topic/object`) and search the DITA-OT resources for a similar stylesheet. The search might find the XSLT stylesheet `DITA-OT-DIR/plugins/org.dita.xhtml/xsl/xslt4h/xsl/xsl4h/dita2htmlImpl.xsl`.
You can use it as a starting point to overwrite the `xhtmlMedia.xsl` stylesheet. For example, the results might be:

```xml
<xsl:template
    match="*:[contains(@class, 'topic/object')][contains(@outputclass, 'video')]">
    <video class="embed-responsive-item">
        <xsl:call-template name="commonattributes"/>
        <xsl:call-template name="setidname"/>
        <xsl:call-template name="copySource"/>
    </video>
</xsl:template>
```

4. Create additional XSLT stylesheets to customize all other desired output types. In this example, to customize the PDF output it is necessary to create an XSLT stylesheet called `pdfMedia.xsl` (in the same plugin folder).

In this case, you might find an appropriate XSLT stylesheet called `DITA-OT-DIR/plugins/org.dita.pdf2/xsl/fo/topic.xsl` to use as a starting point to overwrite the `pdfMedia.xsl` stylesheet, with results looking something like this:

```xml
<!--Treat video, audio or iframe objects as links-->
<xsl:template
    match="*:[contains(@class,'topic/object')][@outputclass = 'video']">
    <xsl:variable name="target" select="@data"/>
    <xsl:variable name="baseDir">
        <xsl:call-template name="substring-before-last">
            <xsl:with-param name="text" select="@xtrf"/>
            <xsl:with-param name="delim" select="'/'/"/>
        </xsl:call-template>
    </xsl:variable>
    <fo:inline xsl:use-attribute-sets="object">
        <xsl:call-template name="commonattributes"/>
        <xsl:if test="exists($target)">
            <fo:basic-link external-destination="url({$target})" xsl:use-attribute-sets="xref">
                <xsl:value-of select="$target"/>
            </fo:basic-link>
        </xsl:if>
    </fo:inline>
</xsl:template>
```

5. To install the created plugin in the DITA-OT, run the built-in transformation scenario called Run DITA-OT Integrator (on page 3102) by executing it from the Apply Transformation Scenario(s) dialog.
box (on page 1433). If the integrator is not visible, select the Show all scenarios option that is available in the Settings drop-down menu. For more information, see Installing a DITA-OT Plugin (on page 3164).

Results of running the integrator using the media plugin example:

XSLT content is applied with priority when publishing to both HTML and PDF outputs.

a. For the HTML output, in the XSLT stylesheet DITA-OT-DIR/plugins/org.dita.xhtml/xsl/dita2html-base.xsl, a new import automatically appeared:

   `<xsl:import href="../plugins/com.oxygenxml.media/xhtmlMedia.xsl"/>

This import is placed after all base imports and thus has a higher priority. For more information about imported template precedence, see: http://www.w3.org/TR/xslt#import.

b. Likewise, for the PDF output, in the top-level stylesheet DITA-OT-DIR/plugins/org.dita.pdf2/xsl/fo/topic2fo_shell_fop.xsl, a new import statement appeared:

   `<xsl:import href="../../../com.oxygenxml.media/pdfMedia.xsl"/>

Now, you can distribute your plugin folder to anyone that has a DITA-OT installation along with some simple installation notes. Your customization will work provided the templates you are overwriting have not changed from one DITA-OT distribution to the other.

Related Information:
DITA Open Toolkit Documentation

Installing a DITA-OT Plugin

Oxygen XML Editor comes bundled with various DITA-OT plugins (on page 3222), but the architecture of the DITA Open Toolkit also allows you to install additional plugins that can be found from various sources (for example, Oxygen's public GitHub repository includes some DITA-OT plugins).

Warning:
Oxygen XML Editor has only been tested with DITA-OT plugins that come bundled with the product. Be aware that installing additional plugins or modifying the bundled plugins may affect the default functionality of the application.

Installing a DITA-OT Plugin

To install a DITA-OT plugin, following this procedure:
1. Copy the additional plugin to the location of the DITA-OT version you are using (by default, DITA-OT-\plugins directory).

2. Select the Configure Transformation Scenario(s) action from the DITA Maps Manager toolbar (you could also use the same action on the main toolbar or open the Transformation Scenarios view).

3. Select the Run DITA-OT Integrator transformation scenario. If the integrator is not visible, select the Show all scenarios option that is available in the Settings drop-down menu.

**Important:**

The folder where the DITA-OT is located needs to have full write access permissions set to it. For example, in Windows, if you are integrating plugins in the DITA-OT folder bundled with Oxygen XML Editor and your application is installed in the Program Files folder, you can start the Oxygen XML Editor main executable with administrative rights for the integrator process to be able to modify resources in the DITA-OT folder.

4. Apply the scenario.

5. Check the Results panel at the bottom of the application to make sure the build was successful.

After the installation, you can open a DITA map and use the Configure Transformation Scenarios dialog box to create a new DITA-OT transformation scenario. Oxygen XML Editor detects that transformation type declaration from the DITA-OT plugin and presents descriptions in the DITA Transformation Type dialog box. Oxygen XML Editor also shows the contributed parameters from the plugin in the transformation scenario's Parameters tab.

**Tip:**

You can declare the transformation type and allowed parameters by following the procedure found in: Defining the Transformation Type and Allowed Parameters in a DITA-OT Plugin.

---

**Defining the Transformation Type and Allowed Parameters in a DITA-OT Plugin**

Custom DITA-OT plugins may contribute new transformation types (transtypes) and each transtype may have a set of allowed configuration parameters. If a DITA-OT plugin declares a transtype, Oxygen XML Editor detects that transformation type declaration and presents descriptions in the DITA Transformation Type...
dialog box (on page 1490) and the contributed parameters in the transformation scenario's Parameters tab (on page 3112).

To define a transformation type and its contributed parameters in a DITA-OT plugin, follow this procedure:

1. If you have not already done so, create a DITA-OT plugin.xml file (on page 3160) (you can easily create this file by using the DITA-OT Plugin new document template in the New document wizard (on page 368)).

2. In the plugin.xml file, define the transformation type details by using the <transtype> element to specify a name, description, and the transtype it extends.

   ```xml
   <transtype name="xhtml" extends="base-html" desc="HTML">
   </transtype>
   ```

3. Define allowed parameters by using the <param> element to specify the name, description, and information about the default and allowed set of values. For more information, see: https://www.dita-ot.org/3.1/topics/plugin-configfile.html.

   ```xml
   <param name="args.indexshow" desc="Specifies whether to show the index" type="enum">
     <val>yes</val>
     <val default="true">no</val>
   </param>
   ```

   Depending on the type declared for a parameter, Oxygen XML Editor will help you pick values for each parameter edited in the Parameters tab of the transformation scenario configuration dialog box. For example, for parameters of type "enum", Oxygen XML Editor will present a combo box for choosing the proper value for the parameter.

4. You can also extend various extension points in your plugin.xml. For more information, see: https://www.dita-ot.org/3.1/extension-points/plugin-extension-points.html.

```xml
<extensionPlugin>
  <transtype name="pdf-css-html5">
    <promotedParams>
      <param name="args.css" promotedName="CSS"/>
      <param name="args.css.param.numbering" promotedName="Numbering"/>
      <param name="args.chapter.layout" promotedName="Chapter layout"/>
    </promotedParams>
  </transtype>
</extensionPlugin>
```
The example above results in the **Parameters** tab looking like this:

**Figure 808. Promoted Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>args.css</td>
<td>$conflicted.css</td>
<td>You can use this to specify a list of CSS URLs to be used...</td>
</tr>
<tr>
<td>args.input</td>
<td>$conflicted.input</td>
<td>Specifies the master file for your documentation project.</td>
</tr>
<tr>
<td>css.processor.type</td>
<td>chemistry</td>
<td>Choose the formatting processor. Allowed values: &quot;ch...</td>
</tr>
<tr>
<td>dita.dir</td>
<td>$conflicted.dita.dir</td>
<td>Specifies where DITA-OT is installed.</td>
</tr>
<tr>
<td>args.chapter.layout</td>
<td>BASIC</td>
<td>Specifies whether chapter level TOCs are generated f...</td>
</tr>
<tr>
<td>args.css-param.numbering</td>
<td>shallow</td>
<td>When set to 'shallow', only the first level topics (the c...</td>
</tr>
<tr>
<td>args.css-param.title-layout</td>
<td>normal</td>
<td>Specifies the layout used for the titles. Allowed values...</td>
</tr>
</tbody>
</table>

5. **Install the plugin (on page 3164).**

**Note:**

If the plugin is installed using an external command line, you may need to restart Oxygen XML Editor to properly re-detect the newly contributed transtypes and parameters.

**Example of a plugin.xml File:**

```xml
<plugin id="com.oxygenxml.pdf.prince">
    <!-- extensions -->
    <feature extension="dita.conductor.transtype.check" value="pdf-prince" type="txt"/>
    <feature extension="dita.conductor.target.relative" value="integrator.xml" type="file"/>
    <feature extension="dita.transtype.print" value="pdf-prince"/>
    <transtype name="pdf-prince" extends="commons" desc="PDF (Prince XML)">
        <param name="princeExecPath" type="file" desc="Path to the Prince executable"/>
    </transtype>
</plugin>
```

For more information, watch this DITA-OT Day 2015 presentation:

https://www.youtube.com/embed/LcrRYUFiq4

**Built-in Third-Party DITA Open Toolkit Plugins**

The DITA Open Toolkit 3.7.0 distribution that is bundled with Oxygen XML Editor includes some pre-installed third-party open-source **plugins (on page 3222)** that add extra publishing formats and functionality.

The **plugins** that come bundled with Oxygen XML Editor include:
• **DITA For Publishers** - These *plugins* allow DITA content to be published to additional formats, such as EPUB 2.0 and Kindle.

• **DITA to Word** - This *plugin* allows users to publish DITA content to MS Word.

• **DITA Community** - These *plugins* allow support for DITA 1.3 with embedded or referenced MathML and SVG images.

### Extra Free Publishing Plugins

The DITA Open Toolkit publishing engine comes with support for predefined output formats such as HTM5, PDF, and Eclipse Help. Since the architecture of the publishing engine is plugin-based, over time, lots of useful *plugins* were developed in the Oxygen XML GitHub account that enhance the publishing and some of them are listed below. The plugins that are already installed within the DITA-OT engine that comes bundled with Oxygen XML Editor are listed with a [Bundled] marker.

#### Plugin that Converts DITA Maps to PDF Using CSS 3 [Bundled]

You can use this very popular *plugin* to publish DITA to PDF output using CSS. As the publishing engine, it can use the Oxygen XML Chemistry processor (freely bundled with Oxygen XML Editor), the Antenna House engine, or the Prince XML engine.

#### DITA Metrics Report [Bundled]

This is a very useful open-source *plugin* can be used to generate an HTML report from an existing DITA project and contains a lot of useful information, including:

- Total number of maps and topics that are part of the project.
- Total number of elements used in topics and maps along with a table presenting all element names and their usage counter.
- The used elements for each DITA domain.
- Total number of attributes used in topics and maps along with a table presenting all attribute names and their usage counter.
- Statistics about the conditional attributes used in the project.
- Information about content reuse.
- Text and content statistics, including both total words (word count) and unique words (vocabulary).
- List of largest and smallest topics and the number of words each one uses.
- Listing of all links to resources outside of the project.
- A metrics evolution report between different versions of your documentation.

#### Export DITA Map Plugin [Bundled]

You can use this free *plugin* to create a ZIP file from your entire DITA project. The plugin also takes filters/profiling into account when including topics.
Publish DITA Content with References to Video and Audio Resources. [Bundled]

A DITA Open Toolkit plugin can be used to convert the DITA `<object>` element to various HTML 5 structures (such as `<video>`, `<audio>`, or `<iframe>`).

Show Consecutive Codeblocks in Multiple Tabs for WebHelp Output

This open-source plugin can be used to display consecutive DITA `<codeblock>` elements in separate tabs.

Add Edit Links in HTML or PDF-based Output [Bundled]

This plugin can be used to add edit links in HTML or PDF-based output that allows subject matter experts to offer feedback for the published content directly in the source using a DITA web editing tool (such as Oxygen XML Web Author).

Create a Single Merged XML Document From an Entire DITA Project [Bundled]

This plugin can be used to produce a merged output from the entire DITA map structure without further processing. It is useful if you want to further process the merged XML document for producing various reports.

Dynamically Publish Excel Content as DITA

A DITA Open Toolkit plugin that can be used to dynamically convert Excel files to DITA (Excel files referenced with `format=excel` in DITA maps).

Dynamically Use JSON Content in DITA Topics

A DITA Open Toolkit plugin that can be used to dynamically convert JSON content to DITA (JSON files referenced with `format=json` in DITA maps).

Dynamically Publish ASCIIDoc Content as DITA

A DITA Open Toolkit plugin that can be used to dynamically convert ASCIIDoc content to DITA (ASCIIDoc files referenced with `format=ant-parser` in DITA maps).

Embed HTML Content in DITA Topics [Bundled]

A plugin that can be used to embed well-formed HTML content in a DITA topic inside a special element.

Embed LaTeX Equations in DITA Content

A DITA Open Toolkit plugin that can be used to publish embedded **LaTeX** mathematical equations to HTML and PDF.

Embed UML Diagrams in DITA Content

A DITA Open Toolkit plugin that can be used to publish embedded UML diagrams equations to HTML and PDF.
**Float Images in HTML and PDF Outputs**

A plugin that can be used to float an image referenced in a DITA topic left or right depending on the specified @outputclass attribute value.

**Embed Referenced MathML and SVGZ Images in HTML Output**

A DITA Open Toolkit plugin that can be used to embed referenced MathML and SVG images in the HTML5 and XHTML output.

**Dynamically Convert DITA Tables to Graphs**

A DITA Open Toolkit plugin that converts DITA tables having a certain structure to SVG graphs.

**Show Oxygen Change Tracking Information in the PDF Output [Bundled]**

This plugin can be used to display Oxygen XML Editor tracked changes (insertions, deletions, or comments) in the PDF output.

**Sample Customization Plugin for Classic PDF (XSL-FO) Output**

This sample DITA Open Toolkit PDF customization plugin is a good starting point if you want to:

- Customize fonts.
- Customize a cover page to provide custom logos and coloring.
- Customize page headers and footers.

**PDF (XSL-FO) - Generate Numbers Before a Topic's Title**

A DITA-OT PDF2 customization plugin that can be installed to generate numbers before each topic's title.

**Presents Chapters With Landscape Orientation in PDF (XSL-FO) output**

A PDF customization folder that can be used to define landscape orientation for a certain chapter.

**Using an External DITA Open Toolkit in Oxygen XML Editor**

Oxygen XML Editor comes bundled with a DITA Open Toolkit, located in the DITA-OT-DIR directory. Starting with Oxygen XML Editor version 17, if you want to use an external DITA-OT for all transformations and validations, you can open the Preferences dialog box (Options > Preferences) (on page 124) and go to the DITA page (on page 269), where you can specify the DITA-OT to be used. Otherwise, to use an external DITA Open Toolkit, follow these steps:
1. Edit your transformation scenarios and in the Parameters tab change the value for the dita.dir parameter to point to the new directory.

2. To make changes in the libraries that come with the DITA Open Toolkit and are used by the Ant process, go to the Advanced tab, click the Libraries button and deselect Allow Oxygen to add high priority libraries to classpath.

3. If there are also changes in the DTDs and you want to use the new versions for content completion and validation, go to the Document Type Association preferences page (on page 138), edit the DITA and DITA Map document types and modify the catalog entry in the Catalogs tab to point to the custom catalog file catalog-dita.xml.

Related Information:
- Editing a Transformation Scenario (on page 1545)
- Creating New Transformation Scenarios (on page 1464)
- DITA Open Toolkit Documentation

DITA Open Toolkit Project

The DITA Open Toolkit project file allows you to define all your DITA map input and filter pairs and to produce the desired output formats by applying the publishing engine over this single project file: https://www.dita-ot.org/dev/topics/using-project-files.html.

Oxygen XML Editor has special support for creating, editing, validating, and publishing DITA Open Toolkit project files represented in XML format. It can also use such files to detect connections between DITA resources in the entire project and to apply root map and filter pairs when editing.

Editing DITA Open Toolkit Project Files

The New Document wizard (on page 368) includes a template to help you create DITA Open Toolkit project files (with an .xml file extension). The template is located in the Framework templates > DITA-OT folder. There is also a sample project file that can be found in the application samples folder: OXYGEN_INSTALL_DIR/samples/dita/mobile-phone/mobilePhoneProjectFile.xml.

When working with a DITA-OT project file in the Author visual editing mode, you can see a compact representation of the file by default. You can switch to the Edit style in the Styles toolbar drop-down menu to edit the file using form controls and inline buttons. The additional View as YAML style can be selected to see a visual representation of the same document in YAML.

DITA-OT Project File Content Completion

Content completion is available according to the associated schema and it is enhanced with proposals for ID references, available transformation types, parameter names, and values.
DITA-OT Project File Validation

The default automatic validation support for DITA-OT project files has enhanced Schematron rules that report invalid references to non-existing contexts. The default validation is based on a validation scenario named **DITA-OT Project** that is included in the DITA-OT project framework.

The DITA-OT Project framework also includes a validation scenario named **DITA-OT Project Validation and Completeness Check**. It contains validation units that automatically validate the project file based on the **DITA-OT Project** scenario and also a manual validation unit based on the **DITA-OT Project Validation and Completeness Check** validation engine that validates all contexts recursively.

When creating a validation scenario (on page 788), or editing an existing scenario (on page 798) for a DITA-OT project file, you can select **DITA-OT Project Validation and Completeness Check** engine in the Validation engine column and clicking the **Settings** button for that engine opens the Configure validation engine dialog box where you can configure options for validating the DITA-OT project.

![Configure Validation Engine Dialog Box for DITA-OT Project Validation](image)

The options available in this dialog box include:

**Batch validate referenced DITA resources**

This option specifies the level of validation that applies to referenced DITA files:
• If the checkbox is left unchecked (default setting), the DITA files will be validated using the rules defined in the DTD or XML Schema declared in the document.
• If the checkbox is selected, the DITA files will be validated using rules defined in their associated validation scenario (on page 787).

Note:
If you want your DITA topics to be validated against a Relax NG schema, this checkbox must be selected.

Check the existence of non-DITA references resources
Extends the validation of referenced resources to non-DITA files.

Include remote resources
Select this option if you want to check that remote referenced binary resources (such as images, movie clips, ZIP archives) exist at the specified location.

Report references to resources outside of the DITA map folder
If selected, it will report any references to DITA resources that are located outside the main DITA map (on page 3223) folder.

Report links to topics not referenced in DITA maps
Checks that all the topics referenced by other topics are also linked in the DITA map.

Report multiple references to the same topic
If selected, it will report warnings when a topic is referenced multiple times in the DITA map, unless a unique @copy-to attribute is used on the <topicref> element for any topic that is referenced multiple times.

For example, it will not report a warning if there is a topic referenced twice, but the second <topicref> has a @copy-to attribute set:

```
<topicref href="topic.dita"/>
......
<topicref href="topic.dita" copy-to="topic2.dita"/>
```

On the other hand, it will report a warning if there is a topic referenced twice and none of the reference-type elements has a @copy-to attribute set or both of them have the @copy-to attribute set to the same value:

```
<topicref href="topic.dita" copy-to="topic2.dita"/>
......
<topicref href="topic.dita" copy-to="topic2.dita"/>
```

Check for duplicate topic IDs within the DITA map context
Checks for multiple topics with the same ID in the context of the entire map.
Report duplicate key definitions

Checks the DITA map for multiple key references with the same key defined for them. This is helpful because if you have two different resources with the same value for the @keys attribute, all references will point to the first one encountered and the other will be ignored.

Note:
This option takes key scopes (on page 3058) into account. For example, if you have something like this:

```xml
<topicref href="t2.dita" keys="k2"/>
<topicgroup keyscope="ks">
  <topicref href="t2.dita" keys="k2"/>
</topicgroup>
```

It will not report the “k2” key as a duplicate because it is defined in a key scope (on page 3058) on the second occurrence.

Report unreferenced key definitions

Checks the entire DITA map and reports any key definitions that are not referenced anywhere. Note that if the Use DITAVAL filters option is selected, this check will search for unreferenced key definitions based upon your selected filter.

Report unreferenced reusable elements

Checks the entire DITA map and reports any detected reusable elements that are not referenced anywhere. It looks for elements that have an ID specified in the following types of topic references:

- Any `<topicref>` that contains a @processing-role attribute set to resource-only.
- Any other referenced topic that contains elements that are reused elsewhere through a @conref or @conkeyref.

Report table layout problems

Looks for table layout problems. The types of errors that may be reported include:

- If a row has fewer cells than the number of columns detected.
- For a CALS table, if a cell has a vertical span greater than the available rows count.
- For a CALS table, if the number of `<colspecs>` is different than the number of columns detected from the table @cols attribute.
- For a CALS table, if the number of columns detected from the table @cols attribute is different than the number of columns detected in the table structure.
- For a CALS table, if the value of the @cols, @rowsep, or @colsep attributes are not numeric.
- For a CALS table, if the @namest, @nameend, or @colname attributes point to an incorrect column name.
Identify possible conflicts in profile attribute values

When the profiling attributes of a topic contain values that are not found in parent topic profiling attributes, the content of the topic is overshadowed when generating profiled output. This option reports these possible conflicts.

Report attributes and values that conflict with profiling preferences

Looks for profiling attributes and values that are not defined in the Profiling / Conditional Text preferences page (on page 187) (you can click the Profiling Preferences button to open this preferences page). It also checks if profiling attributes defined as single-value have multiple values set in the searched topics.

Additional Schematron checks

Allows you to select a Schematron file that Oxygen XML Editor will use for the validation of DITA resources. You can specify the path by using the text field, its history drop-down, the Insert Editor Variables (on page 323) button, or the browsing actions in the Browse drop-down list.

Publishing DITA Open Toolkit Project Files

Once a DITA-OT project file is opened in the application, two predefined publishing scenarios become available in the Configure Transformation Scenario(s) dialog box (on page 1548):

- **Publish DITA-OT Project (all deliverables)** - Runs the publishing engine and produces output for all deliverables defined in the project file.
- **Publish DITA-OT Project (select deliverable)** - Runs the publishing engine and produces output for only one deliverable specified by the end-user.

Some of the allowed transformation parameters that are relevant to the DITA-OT project file include:

- **project.file** - Specifies the path to the project file.
- **dita-ot.dir** - Specifies the directory where DITA-OT, used in transformation is installed.
- **additional.args** - Specifies the additional arguments used in transformation.
- **deliverable.id** - Specifies the id of the deliverable. This parameter is only available in the Publish DITA-OT Project (select deliverable) transformation.

Main Files Support for DITA Open Toolkit Project Files

If you enable main files support at project level (on page 3181), you can choose to detect all top-level DITA Open Toolkit project files and to add them to the Main Files folder. You could also manually add the top-level files for your DITA-OT project in the Main Files folder. The benefit of this is that whenever you rename or move files in the Project view, the references to those resources will automatically be updated.

Tip:
The Resource Hierarchy/Dependencies view (on page 3183) also works for DITA-OT project files.
Editing Contexts Detected from DITA Open Toolkit Project Files

Once a DITA-OT project file is added to the Main Files folder, the Context drop-down menu on the DITA Maps Manager toolbar will contain context DITA maps defined in the project file and the Profiling/Conditional Text menu will contain filter pairs gathered from the project file. When you select one of them in the drop-down menu, the application gathers the keys from the context DITA map and applies the filters specified in that context.

DITA Specialization Support

DITA is designed to let you design new markup and new document types that allow any general-purpose DITA processor to process documents that use the new markup. This in turn enables blind interchange of DITA documents from any source. In particular, in the context of a map, you can combine topics of any type and get usable results from any general-purpose DITA processor. Specialization is the one truly unique and distinguishing aspect of DITA. Even if you have no use for any aspect of DITA modularity or reuse, you still have a use for specialization simply because it enables reliable interchange in a way that no other XML application does.

For detailed information and step-by-step tutorials about DITA specializations, see DITA 4 Practitioners: DITA Configuration and Specialization Tutorials.

In addition, the topics in this section contain information about using DITA specializations in Oxygen XML Editor.

Integrating a DITA Specialization

A DITA specialization can have its document type defined with any of the following:

- **DTD** - For configuration and specialization tutorials, see [http://dita4practitioners.github.io/dita-specialization-tutorials/](http://dita4practitioners.github.io/dita-specialization-tutorials/).
- **XSD** - For configuration and specialization tutorials, see [http://dita4practitioners.github.io/dita-specialization-tutorials/](http://dita4practitioners.github.io/dita-specialization-tutorials/).
- **Relax NG** - For more information, see the following presentation: Creating DITA-OT Constraint/ Specialization Plugins. For Relax NG coding requirements, see [https://www.oxygenxml.com/dita/1.3/specs/archSpec/base/relax-ng-requirements.html](https://www.oxygenxml.com/dita/1.3/specs/archSpec/base/relax-ng-requirements.html).

A DITA specialization may optionally include specialized processing, that is new XSLT template rules that match the extension part of the @class attribute values of the new elements, and thus extend the default processing available in the DITA Open Toolkit.

To integrate a DITA specialization into Oxygen XML Editor, use one of the following methods:

**DITA-OT Plugin Method**

If the DITA specialization is available as a DITA Open Toolkit plugin, follow this procedure:
1. Copy the additional plugin to the location of the DITA-OT version you are using (by default, \DITA-OT\plugins directory).

**Important:**
The application needs to have full write access permissions to the DITA-OT directory.

2. If Oxygen XML Editor was installed in the default location, you may need to restart and run it as an administrator.

3. Select the Configure Transformation Scenario(s) (on page 1548) action from the DITA Maps Manager toolbar (you could also use the same action on the main toolbar or open the Transformation Scenarios view (on page 1555)).

4. Select the Run DITA-OT Integrator transformation scenario (on page 3102).

**Tip:**
If you don’t see that scenario in the Configure Transformation Scenario(s) (on page 1548) dialog box or Transformation Scenarios view (on page 1555), click the Settings button and select the Show all scenarios option, but don’t forget to change it back to Show only the scenarios available for the editor after you are finished with this procedure.

5. Apply the scenario (on page 1548).

6. Check the Results panel at the bottom of the application to make sure the build was successful.

7. Restart Oxygen XML Editor with your normal permissions.

**Tip:**
Oxygen XML Editor detects new document templates (on page 375) contributed by the DITA-OT plugin as long as you do the following:

- Create a new folder called template_folders inside your DITA OT plugin's folder. For example: \DITA-OT-DIR\plugins\my_custom_plugin\template_folders.
- Create one or more subfolders inside the template_folders directory that contain the new document templates. The new document templates found in those subfolders will be available in the New document wizard.

### Alternative Methods

If the DITA specialization is not available as a DITA-OT plugin, you have the following options:

- If the DTDs that define the extension elements are located in a folder outside the DITA Open Toolkit folder, add new rules to the DITA-OT catalog file. These rules are meant for resolving the DTD references from the DITA files that use the specialized elements to that folder. This allows for correct resolution of DTD references to your local DTD files and is needed for both validation and
transformation of the DITA maps or topics. The DITA-OT catalog file is called `catalog-dita.xml` and is located in the root folder of the DITA Open Toolkit.

- If there is specialized processing provided by XSLT stylesheets that override the default stylesheets from DITA-OT, these new stylesheets must be called from the DITA-OT Ant build scripts.

**Important:**

If you are using DITA specialization elements in your DITA files, it is recommended that you activate the Enable DTD/XML Schema processing in document type detection option in the Document Type Association preferences page (on page 138).

- You could create your own document templates (on page 375), store them in a custom directory, then add that directory to the list of template directories that Oxygen XML Editor uses by adding the directory to the list in the Document Templates Preferences (on page 167) page.

### Editing DITA Map Specializations

In addition to recognizing the default DITA map (on page 3218) formats (`<map>` and `<bookmap>`), the DITA Maps Manager view (on page 2897) can also be used to open and edit specializations of DITA maps.

All advanced editing actions available for the map (such as insertion actions or editing properties) allow you to specify the element in an editable combo box. The elements that initially appear in the combo box are all the elements that are allowed to appear at the insert position for the given specialization.

The topic titles rendered in the DITA Maps Manager view (on page 2897) are collected from the target files by matching the `@class` attribute and not a specific element name.

When editing DITA specializations of maps in the main editor, the insertions of topic reference, topic heading, topic group and conref actions should work without modification. For the table actions, you have to modify each action manually to insert the correct element name at the cursor position. You can go to the DITA Map document type from the Document Type Association preferences page (on page 138) and edit the table actions to insert the element names as specified in your specialization. See Creating a Framework through the Configuration Dialog (on page 2133) for more details.
Editing DITA Topic Specializations

In addition to recognizing the default DITA topic formats, topic specializations can also be edited in Author mode.

The content completion should work without additional modifications and you can choose the tags that are allowed at the cursor position.

The CSS styles used for rendering the elements should also work on the specialized topics without additional modifications.

The toolbar/menu actions should be customized to insert the correct element names. You can go to the DITA document type from the Document Type Association preferences page (on page 138) and edit the actions to insert the element names, as specified in your specialization. See Creating a Framework through the Configuration Dialog (on page 2133) for more details.

Related Information:
DITA Configuration and Specialization Tutorials
Integrating a DITA Specialization (on page 3176)

Translating DITA Projects Overview

This topic contains some general information about translating DITA content and is meant to help those who do not store their DITA projects through a Content Management System (CMS) or other type of service that already includes their own translation support.

Choosing a Translation Agency

To minimize translation costs, it is recommended to choose a translation agency that is able to handle DITA content directly, without requiring you to convert the content to some intermediary format. This means that you benefit from the DITA reusable content features (on page 3031).

If you plan to translate your DITA project, it is also recommended that you contact a DITA-aware translation agency as early in your process as possible because translation agencies who translate DITA content directly usually need to have a preliminary discussion about how your project is structured, which terms need to be skipped when translating, how various measuring units are translated, how content is reused, your metadata strategy, and how screenshots are handled. Those discussions may influence the way that you organize and write your DITA content.

Note:
If your translation agency does not directly handle DITA content, there are commercial tools that can be used to convert DITA to XLIFF (for example, https://www.maxprograms.com/products/fluenta.html).
Optimizing Content for Translation

In general, there are three main principles to take into account when writing DITA content that will be translated:

1. Use a controlled vocabulary (for example, the Simplified Technical English vocabulary).
2. Avoid reusing inline elements other than product names. The following DITA Users List discussion describes the reasons for this: https://lists.oasis-open.org/archives/dita/201301/msg00029.html.
3. Avoid profiling/filtering content at inline level, for similar reasons.

General DITA Project Structure

It is usually considered best practice to organize your DITA maps/topics in a separate folder for each language. One folder that contains the English version of all of your DITA resources and a separate folder for each of the other languages you will translate with equivalent DITA resources translated in that specific language.

General Translation Workflow

When translating DITA content, the most common workflow involves these steps:

1. Create your content in the primary language.
2. Before each release, you gather all the DITA files that have been changed and need to be translated. The DITA Translation Package Builder Add-on (on page 2548) could be handy for this.
3. Send a copy of the relevant DITA files to the translation agency (known also as "localization service provider").
4. Receive translated DITA content back from the translation agency and integrate it in each language-specific project folder (on page 3180).

Publishing Translated Content

All of your translated DITA maps and topics should have the xml:lang attribute set with the appropriate value on the root element. Along with the actual translated content, the published output may also contain static text (such as the word Table followed by the table number, Figure following by the number, or Note appearing before the content of each DITA <note> element). The DITA Open Toolkit includes support for various languages for HTML-based output and PDF-based output. You can also add support for other languages: Globalizing DITA Content: Customizing Generated Text. For information about how to add a new language to the Oxygen WebHelp Responsive output, see Adding a New Language (on page 1683).

Liability

Translation agencies usually do not assume any liability for incorrectly translated content. If possible, it is recommended to have someone who is familiar with the particular language be responsible for reviewing and accepting the translated content. For example, if your company has regional headquarters located in various countries, perhaps someone from each headquarters could review the translated content.
Other Resources

Here are some links to other resources that might help you with translating DITA projects:

- DITA Translation: Organizing Your DITA Files
- DITA Translation: Using XLIFF to Translate DITA Projects
- WhP Localization Services Blog Page

Main Files Support in DITA

Oxygen XML Editor includes a feature that allows you to define Main Files on page 3220 at project level. This feature is typically used in Oxygen XML Editor for XML documents to determine the context for operations such as validation, content completion, refactoring, searches, or displaying components collected from various modules. For DITA projects, this feature has a more limited purpose in Oxygen XML Editor since it is mainly used to provide the means for updating references to moved or renamed resources. Since you can move or rename DITA resources (such as topics and maps) in the DITA Maps Manager on page 2897, the root map on page 3223 is used as the scope to update all the references to the moved or renamed resources. However, you do not have this option for non-DITA resources (such as folders, images, HTML files, audio, video, text files, Markdown documents) since they do not appear in the DITA Maps Manager. Also, when moving DITA resources in the DITA Maps Manager, you have to do it one at a time.

You can use the Main Files support in DITA to update all the references to moved or renamed resources in the scope of the Main Files, and since the root map on page 3223 will be set as the Main File, you achieve the same result as if you were moving or renaming them in the DITA Maps Manager. It also allows you to move multiple DITA resources (or entire folders) at once in the Project view on page 402, instead of the DITA Maps Manager, while still giving you the option of updating all the references.

How to Enable Main Files Support in DITA

To use the Main Files support in DITA, follow these steps:

1. Go to the Project view on page 402 and enable Main Files support with one of the following methods:
   ◦ Select Enable Main Files Support from the Settings menu in the top-right corner.
   ◦ Select Enable Main Files Support from the contextual menu of the project root folder. If a disabled Main Files folder exists, you can also select that option from its contextual menu.
   ◦ Click the Enable button in the tooltip located at the bottom. This tooltip window is displayed when the Main Files support is disabled. Clicking the Read more link takes you to the user guide. Clicking the Enable button opens the Enable Main Files dialog box. This dialog box contains general information about the Main Files Support and allows you to enable it.

   Warning:
   Once you close this window tip, Oxygen XML Editor hides it for all projects. You can make the window tip reappear by resetting Oxygen XML Editor to its default settings on page 314. However, doing so will result in you losing your customized options.
2. Add the main DITA map (root map) (on page 3223) to the Main Files folder by doing one of the following:
   ◦ Right-click the project root folder and select Detect Main Files.
   ◦ Right-click the Main Files folder and select Detect Main Files from Project.
   ◦ If you enabled the Main Files support from the tooltip at the bottom of the Project view, you can also use the Detect and Enable button in the resulting dialog box to detect the main files from the current project.
   ◦ Manually add the root map (on page 3223) to the Main Files folder by doing one of the following:
     • Right-click a file from your project and select Add to Main Files from the contextual menu (or simply drag and drop it into the Main Files folder).
     • Select Add Files or Add Edited File from the contextual menu of the Main Files folder.

   Tip:
   You can set multiple maps in the Main Files folder and all of them will automatically be added to the list of root maps you can select from the drop-down menu in the DITA Maps Manager toolbar (on page 2901).

3. [Alternative] If you have a defined DITA Open Toolkit project XML file (on page 3175) you can add it to the Main Files folder. Once you do that the application will know the dependencies between all resources directly and indirectly referenced from the project file, including DITA maps, topics, binary resources and DITAVAL filter files.

Moving or Renaming Non-DITA Resources and Updating the References to Them

With the Main Files support enabled, you can move or rename non-DITA resources (such as folders, images, HTML files, audio, video, text files, Markdown documents) or move multiple normal DITA resources (or entire folders) in the Project view (on page 402) and Oxygen XML Editor will offer the option of updating all the references to the moved or renamed resources in the scope of the Main Files (in this case, the main DITA map (root map) (on page 3223)).

To move or rename non-DITA resources (or move multiple DITA resources) and update the references to them, follow these steps:

1. Enable Main Files support and add your root DITA map (on page 3223) to the Main Files folder as described in the How to Enable Main Files Support in DITA (on page 3181) section above.

2. Go to the Project view (on page 402), and use one of the following methods to move or rename the resources:

   Moving Resources

   To move resources in the Project view (on page 402), do one of the following:
Simply drag and drop the resource to the new location in the tree structure
(the Enable drag-and-drop in Project view option must be selected in the View
preferences page (on page 306)).

- Use the Cut, Copy, and Paste actions from the contextual menu.
- Right-click the resource and select Refactoring > Move resource action from the
  contextual menu. Note that this method also allows you to specify a new name and
destination path in the Move resource dialog box.

**Result:** In all cases, a Move resource dialog box will be presented.

Renaming Resources

To rename resources in the Project view (on page 402), do one of the following:
- Select the resource and press F2, or simply left-click again, until the in-place editor
  allows you to change the file name.
- Right-click the resource and select Rename or Refactoring > Rename resource.

**Result:** In all cases, a Rename resource dialog box will be presented.

3. Make sure the Update references of the moved resource(s) option is selected in the resulting Move or
   Rename dialog box and keep the scope as main files to make sure all the references to the moved or
   renamed resource are updated.

DITA Resource Hierarchy/Dependencies View

The Resource Hierarchy/Dependencies view displays the hierarchy (on page 3219) or dependencies
(on page 3217) for resources included in an XML document. For DITA resources, it will only show direct
references, so resources that are indirectly referenced through keys are not presented in the hierarchy or
dependencies tree.

To see the hierarchy or dependencies for a DITA resource (maps or topics), right-click a resource in the Project
view (on page 402) and either select Resource Hierarchy or Resource Dependencies.

If you want to view the dependencies for a media resource (such as images) directly referenced in a DITA
topic, click the Show Dependencies button on the toolbar of the Resource Hierarchy/Dependencies view,
select the All files filter in the file browser, find the particular resource, and double-click it.
The following actions are available on the toolbar of the Resource Hierarchy/Dependencies view:

- **Refresh**
  Refreshes the Hierarchy/Dependencies structure.

- **Stop**
  Stops the hierarchy/dependencies from computing.

- **Show Hierarchy**
  Compute the hierarchy structure.

- **Show Dependencies**
  Compute the dependencies structure.

- **Configure dependencies search scope**
  Allows you to configure a scope to compute the dependencies structure. You can restrict the scope to the current project or to one or multiple working sets (on page 3224). If the **Use only Main Files, if enabled** checkbox is selected, the scope of the search is restricted to the Main Files directory (on page 3181).

- **History**
  Provides access to the list of previously computed dependencies. Use the **Clear history** button to remove all items from this list.

The contextual menu for a resource listed in the Resource Hierarchy/Dependencies view contains the following actions:
Open

Opens the resource. You can also double-click a resource in the Hierarchy/Dependencies structure to open it.

Go to reference

Opens the source document where the resource is referenced.

Copy location

Copies the location of the resource.

Move resource

Opens the Move resource dialog box where the following fields are available:

- **Destination** - Presents the path to the current location of the resource you want to move and gives you the option to introduce a new location.
- **New name** - Presents the current name of the moved resource and gives you the option to change it.
- **Update references of the moved resource(s)** - As long as Main Files support is enabled (on page 3181), you can select this option to update the references to the resource you are moving, in accordance with the new location and name. A Preview option is available that allows you to see what will be updated before selecting Move to process the operation.

Rename resource

Opens the Rename resource dialog box where the following fields are available:

- **New name** - Presents the current name of the edited resource and allows you to modify it.
- **Update references of the renamed resource(s)** - As long as Main Files support is enabled (on page 3181), you can select this option to update the references to the resource you are renaming. A Preview option is available that allows you to see what will be updated before selecting Rename to process the operation.

Resource Hierarchy

Shows the hierarchy for the selected resource.

Resource Dependencies

Shows the dependencies for the selected resource.

Add to Main Files

Adds the currently selected resource in the Main Files directory.

Expand More

Expands more of the children of the selected resource from the Hierarchy/Dependencies structure.
Collapse All

Collapses all children of the selected resource from the Hierarchy/Dependencies structure.

Tip:
When a recursive reference is encountered in the Hierarchy view, the reference is marked with a special icon.

Search and Rename Actions for IDs in DITA

Oxygen XML Editor allows you to search references to ID attributes (either direct references using the @href and @conref attributes or indirect references using @keyref or @conkeyref attributes) or to rename the id attribute in all the declared and referenced locations. The main benefit of this feature is the fact that it allows you to rename @id attributes (or search for their references) in the scope of the entire project. It also works for IDs defined inside DITA maps and then referenced in maps and topics.

In Author mode, these operations are available for DITA documents in the contextual menu (grouped in the Manage IDs submenu). In Text mode, these actions are also available in the Quick Assist menu. To access it, place the cursor inside the value of an @id attribute and click the yellow light bulb icon.

The possible actions include:

Rename in

Renames the ID and all of its occurrences. Selecting this action opens a dialog box where you insert the new ID value and choose the scope of the rename operation. For a preview of the changes you are about to make, click Preview. This opens the Preview dialog box, which presents a list with the files that contain changes and a preview zone of these changes.

Rename in File (Available in the Text mode only)

Renames the ID you are editing and all its occurrences in the current file.

Search References

Searches for the references of the ID. By default, the scope of this action is the current project.

Search References in

Searches for the references of the ID and you can choose the scope of the operation or configure working sets to use for the scope.

Search Declarations (Available in the Text mode only)

Searches for the declaration of the ID reference. By default, the scope of this action is the current project.

Search Declarations in (Available in the Text mode only)

Searches for the declaration of the ID reference and you can choose the scope of the operation or configure working sets to use for the scope.

Search Occurrences in file
Searches for the declaration and references of the ID in the current document and presents the results in the message panel at the bottom of the application.

**Change scope (Available in the Quick Assist menu in Text mode only)**

Opens a dialog box where you can choose the scope of the operation or configure working sets to use for the scope.

**Tip:**

A quick way to go to the declaration of an ID in Text mode is to move the cursor over an ID reference and use the **Ctrl + Single-Click (Command + Single-Click on macOS)** navigation.

Selecting an ID that you use for search or refactor operations differs between the Text and Author modes. In the Text mode, you position the cursor inside the declaration or reference of an ID. In the Author mode, Oxygen XML Editor collects all the IDs by analyzing each element from the path to the root. If more IDs are available, you are prompted to choose one of them.

**Metadata**

Metadata is a broad concept that describes data that explains or identifies other data. Metadata can be used for many purposes, from driving automation of document builds to enabling authors and readers to find content more easily. DITA provides numerous types of metadata, each of which is used and created differently. Some of the most important forms of metadata in DITA are topic and taxonomy.

**Topic Metadata**

Topic metadata describes the topic and what it is about. Topic metadata can be inserted in the `<prolog>` element of a topic or inside the `<topicref>` element that points to a topic from a map. In other words, metadata about the topic can be asserted by the topic itself, or can be assigned to it by the map that includes it in the build. This allows multiple maps to assign different metadata to the same topic. This may be appropriate when you want to describe a topic differently in various documents.

**Taxonomy and Subject Scheme**

A taxonomy is a controlled vocabulary. It can be used to standardize how many things in your content and metadata are named. This consistency in naming can help ensure that automated processes work correctly, and that consistent terminology is used in content, and in metadata. In DITA, taxonomies are created using **subject scheme maps (on page 3224)**. When you are authoring, many of the values you choose from have been defined in **subject scheme maps**.
Migrating MS Office Documents to DITA

Oxygen XML Editor integrates the entire DITA for Publishers plugins suite and provides some possibilities for migrating content from Microsoft Office® (and other Office-type formats) to DITA. There are also possibilities for migrating various other types of formats. For more information, see Migrating Various Document Formats to and from DITA (on page 3190).

Migration from Office-type formats to XML is rarely perfect and manual changes may need to be made to the converted content, but the methods described below should help you find the best approach for your particular case.

Oxygen XML Editor Batch Documents Converter Add-on (Multiple Documents)

The provided Word to DITA conversion of the Oxygen Batch Documents Converter add-on (on page 2533) contains an option named Create DITA maps from Word documents containing multiple sections. When this option is selected, all sections from your Word document marked by titles or headings will be separated into individual DITA topics and referenced in a DITA map.

Smart Paste (Single Document)

1. Open the document in MS Office (or other similar application), select all the content, and copy it.
2. Open Oxygen XML Editor and create a new DITA topic.
3. Paste the selected content in Author mode. The Smart Paste functionality (on page 613) will attempt to convert the content to DITA structure.

HTML to DITA (Single Document)

1. Save your document as HTML.
2. Once you have converted it to HTML, you have several possibilities:
   ◦ In Oxygen XML Editor, select File > Import/Convert > HTML File to XHTML to import it as XHTML. Then, open the XHTML in Oxygen XML Editor and use one of the XHTML to DITA transformation scenarios (on page 1375) to convert the content to DITA structure.
   ◦ Open the HTML file in any web browser, select all of its content, and copy it. Then, open Oxygen XML Editor, create a new DITA topic, and paste the selected content in Author mode. The Smart Paste functionality (on page 613) will attempt to convert the HTML content to DITA structure.

Word to LibreOffice to DITA (Single Document)

1. Open the document in the LibreOffice application and save it as DocBook.
2. Open the DocBook document in Oxygen XML Editor.
3. Run the built-in DocBook to DITA transformation scenario (on page 1461).
4. You may need to make some manual adjustments for elements that could not be mapped.
Word to DITA using DITA For Publishers (Single Document)

1. Save the document in the MS Word DOCX format.
2. Open it in the Archive Browser view (on page 2008) in Oxygen XML Editor and then open the document.xml file contained in the archive.
3. Run the built-in DOCX DITA transformation scenario. This scenario runs a build file over the DOCX archive and should produce a DITA project that contains a DITA map and multiple topics.
4. You may need to do some manual reconfiguration to map DOCX styles to DITA content. The XSLT conversion is part of the DITA For Publishers plugin and there is documentation for it available here: http://www.dita4publishers.org/d4p-users-guide/user_docs/d4p-users-guide/word2dita/word2dita-intro.html.

Word to DocBook to DITA (Multiple Documents)

1. Use a tool to convert the documents to DocBook. For example, Pandoc is a free document converter engine that can convert DOCX documents to DocBook and according to Pandoc's manual, you can specify multiple input files and use wildcards in the commands.
2. Save the newly converted DocBook documents somewhere in your project.
3. Perform a batch transformation (on page 1553) on all the newly converted DocBook documents:
   a. Select all the DocBook documents in the Project view (on page 402).
   b. Right-click the selected files and choose Transform > Configure Transformation Scenario(s).
   c. Apply the built-in DocBook to DITA transformation scenario (on page 1461).
4. You may need to make some manual adjustments in the resulting documents for elements that could not be mapped.

Word to HTML/Markdown to DITA (Multiple Documents)

1. Use a tool to convert the documents to HTML or Markdown. For example, Pandoc is a free document converter engine that can convert DOCX documents to those formats.
2. Use Oxygen's Batch Converter add-on (on page 2533) to convert the documents to DITA.
3. You may need to make some manual adjustments in the resulting documents for elements that could not be mapped.

Migrating Excel and Other Types of Spreadsheets to DITA

There are two possibilities for converting Microsoft Excel (or other similar types of documents) to DITA:

- Copy the spreadsheet content and paste it in a DITA topic in Author mode. The Smart Paste functionality (on page 613) will attempt to convert the content to DITA.
- Use Oxygen's Batch Converter add-on (on page 2533) to convert one or more spreadsheet documents to DITA.
Migrating Various Document Formats to and from DITA

When organizations decide to use DITA for structuring, developing, managing, or publishing content, they usually already have content written in other formats and need to convert it to DITA. There are a variety of possibilities for a conversion to DITA, depending on the original format of the content.

Migration from other formats to DITA is rarely perfect and manual changes may need to be made to the converted content, but the methods described below should help you find the best approach for your particular case.

### Migrating Microsoft Office and Other Similar Types of Documents to DITA

There are various possibilities for migrating content from Microsoft Office® (and other Office-type formats) to DITA. For details, see Migrating MS Office Documents to DITA (on page 3188).

### Migrating DocBook Content to DITA

The Oxygen Batch Documents Converter add-on (on page 2533) can be used for migrating one or multiple DocBook documents to DITA.

The provided DocBook to DITA conversion contains an option named Create DITA maps from DocBook documents containing multiple sections. When this option is selected, all sections from your DocBook document will be separated into individual DITA topics and referenced in a DITA map.

### Migrating Google Docs to DITA

There are several possibilities to convert Google Docs to DITA:
• Copy the content from Google Docs and paste it in an open DITA topic in Author mode. The Smart Paste functionality (on page 613) will attempt to convert the content to DITA.
• Save the Google document as OpenDocumentFormat (ODF), then open it in the free LibreOffice application and save it as DocBook. Next, open the DocBook document in Oxygen XML Editor and run the built-in transformation scenario called DocBook to DITA (on page 1461).
• If you want to convert multiple Google documents at once, save the documents as HTML, then use Oxygen's Batch Documents Converter add-on (on page 2533) to convert the documents to DITA.

In all cases, you may need to make some manual adjustments in the resulting documents for elements that couldn't be mapped.

**Migrating Markdown Content to DITA**

There are several possibilities to convert Markdown content to DITA:

• The DITA Open Toolkit publishing engine bundled with Oxygen XML Editor allows you to reference Markdown files directly in a DITA map and either publish them directly or export the Markdown files to DITA one by one. For details, see Working with Markdown Documents in DITA (on page 3024).
• If you want to convert multiple Markdown files at once, you can use Oxygen's Batch Documents Converter add-on (on page 2533) to convert the documents to DITA.

**Migrating HTML Content to DITA**

There are several possibilities to convert HTML content to DITA:

• Copy the HTML content and paste it in an open DITA topic in Author mode. The Smart Paste functionality (on page 613) will attempt to convert the content to DITA.
• Convert the HTML file to XHTML by selecting File > Import/Convert > HTML File to XHTML. Then, open the XHTML file and use one of the XHTML to DITA Transformation Scenarios (on page 1375) to convert the content to DITA.
• If you want to convert multiple HTML files at once, you can use Oxygen's Batch Converter add-on (on page 2533) to convert the documents to DITA.

**Migrating Unstructured FrameMaker to DITA**

There is a FrameMaker plugin that can be used for this type of conversion: https://leximation.com/tools/info/fm2dita.php.

**Migrating MadCap Content to DITA**

Some recent MadCap versions seem to have facilities to export content directly to DITA. Otherwise, you will need to convert XHTML content to DITA with a custom XSLT stylesheet to preserve variable references.
Migrating Confluence to DITA

To migrate Confluence content to DITA, first export the content to HTML. For this, log in to your Confluence account and navigate to the specific space that you want to export. Then go to **Space Settings > Export space** and choose to export it as HTML.

You can then use Oxygen's Batch Documents Converter add-on (on page 2533), selecting the **Confluence to DITA** action, to convert the exported `index.html` file into a DITA map with topics.

Migrating Other Formats to DITA

You may find third-party applications (such as Pandoc) that can convert your content to HTML or to some kind of XML format like DocBook. Once you have HTML or DocBook content, you can convert them to DITA using one of the methods described above.

Migrate from DITA to Confluence and Other Formats

There are various possible methods available for converting DITA content to Confluence and other formats (such as Microsoft Word or HTML). For details and ideas for some of the possible methods, see the [DITA to Confluence] blog post.

Resources

For more information about migrating to DITA, see our webinar: Working with DITA in Oxygen - Migrating to DITA and Refactoring.

How to Count Words in DITA Topics or Maps

There are various ways to count words in Oxygen XML Editor:

- Open a DITA topic in the **Author** visual editing mode, right-click anywhere in the editor, and select **Text > Count Words**.
- Open a DITA map in the **DITA Maps Manager** view. Click the **Open map in editor with resolved topics** toolbar button. In the newly opened DITA map, right-click anywhere in the editor and select **Text > Count Words**.
- Open a DITA map in the **DITA Maps Manager** view. Click the **Configure Transformation Scenarios** toolbar button and choose the **DITA Map Metrics Report** transformation scenario.

**Tip:**

Along with the word count, this **DITA Map Metrics Report** transformation provides additional information (such as the number of processed maps and topics, content reuse percentage, the number of elements, attributes, words, and characters used, and more).

DITA 1.3 Support

Starting with version 17.1, Oxygen XML Editor includes support for some DITA 1.3 features.
The Oxygen XML Editor publication of the full DITA 1.3 specifications can be found at https://www.oxygenxml.com/dita/1.3/specs/index.html#introduction/dita-release-overview.html.

The following table is a list of DITA 1.3 features and their implementation status in Oxygen XML Editor:

**Table 54. DITA 1.3 Features Implementation Status**

<table>
<thead>
<tr>
<th>Feature</th>
<th>Editing</th>
<th>Publishing [DITA Open Toolkit 3.7.0 is used]</th>
</tr>
</thead>
<tbody>
<tr>
<td>DITA 1.3 DTD, XML Schema, and Relax NG-based maps/topics/tasks/references, etc.</td>
<td>New DITA 1.3 document templates. By default, DITA topics and maps that do not specify version in the DOCTYPE declaration are also considered to be DITA 1.3</td>
<td>N/A</td>
</tr>
<tr>
<td>Learning Object and Group maps</td>
<td>New document templates</td>
<td>No specific support implemented</td>
</tr>
<tr>
<td>Troubleshooting specialization</td>
<td>Create and edit new troubleshooting topics</td>
<td>No specific support implemented</td>
</tr>
<tr>
<td>XML markup domain</td>
<td>Validation and Content Completion</td>
<td>Special rendering in PDF and XHTML-based outputs</td>
</tr>
</tbody>
</table>
| Equation and MathML domain | Validation and content completion
Display and Insert equations | Special rendering in PDF and XHTML-based outputs |
| SVG domain | Validation and content completion
Display referenced SVG content | Special rendering in PDF and XHTML-based outputs |
| Other new DITA 1.3 elements (div, strike-through, overline, etc.) | Validation and Content Completion | Special rendering in PDF and XHTML-based outputs |
| Release management domain | Validation and Content Completion | No specific support implemented |
| **Scoped keys (on page 3058)** | Define key scopes
Validate and check for completeness | Partially implemented (Various issues may still be encountered) |
Table 54. DITA 1.3 Features Implementation Status (continued)

<table>
<thead>
<tr>
<th>Feature</th>
<th>Editing</th>
<th>Publishing [DI- TA Open Toolk- it 3.7.0 is used]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Resolve keyrefs and conkeyrefs taking key scopes into account</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Key scope information is displayed in a tooltip when hovering over an item in the DITA Maps Manager</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Branch filtering (on page 3060)</td>
<td>Display, create, and edit &lt;ditavalref&gt; elements</td>
<td>Partially implemented (Various issues may still be encountered)</td>
</tr>
<tr>
<td>Key-based cross deliverable addressing</td>
<td>Special display for references to DITA maps with <code>scope=&quot;peer&quot;</code> and a defined keyscope</td>
<td>Not implemented.</td>
</tr>
<tr>
<td>Gather and present keys from peer maps</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Shorthand to address syntax that identifies elements in the same topic</td>
<td>Properly resolved for validation, links, and conrefs</td>
<td>Implemented</td>
</tr>
<tr>
<td>Various table attributes (orientation, rotation, scope, and headers on cells)</td>
<td>Not implemented in the Table Properties action support. However, attributes can be changed from the Attributes view</td>
<td>Not implemented</td>
</tr>
<tr>
<td>New Map topicref attributes (cascade, deliveryTarget)</td>
<td>Allow setting new attributes, propose proper values for them</td>
<td>Implemented</td>
</tr>
</tbody>
</table>

Related information

Watch our DITA 1.3 video tutorial for more information about key scopes and branch filtering.

DITA 2.0 Support

Starting with version 23, Oxygen XML Editor includes support for some DITA 2.0 features. To enable this support, go to the Options > Preferences > DITA page and select the Enable DITA 2.0 Editing Support (Experimental) checkbox. Enabling it results in DITA 2.0 topic and map templates being available in the New document wizard (on page 368).

The following table is a list of DITA 2.0 features and their implementation status in Oxygen XML Editor. The list of proposed DITA 2.0 changes is published here: https://www.oasis-open.org/committees/download.php/65626/DITA-2.0-proposals.pdf.
### Table 55. DITA 2.0 Features Implementation Status

<table>
<thead>
<tr>
<th>Feature</th>
<th>Editing</th>
<th>Publishing [DITA Open Toolk-it 3.7.0 is used]</th>
</tr>
</thead>
<tbody>
<tr>
<td>DITA 2.0 DTD, and Relax NG-based maps/topics/tasks/references, etc.</td>
<td>New DITA 2.0 document templates.</td>
<td>Also supported by the publishing engine.</td>
</tr>
<tr>
<td>Other new DITA 2.0 elements (include, etc.)</td>
<td>Validation and Content Completion.</td>
<td>Special rendering in PDF and XHTML-based outputs: <a href="https://www.dita-ot.org/dev/reference/dita-v2-0-support.html">https://www.dita-ot.org/dev/reference/dita-v2-0-support.html</a>.</td>
</tr>
<tr>
<td>Profiling attributes defined using the new <code>&lt;specializations&gt;</code> attribute.</td>
<td>Profiling attributes defined using the new <code>&lt;specializations&gt;</code> attribute are recognized by the application.</td>
<td>Also supported by the publishing engine.</td>
</tr>
</tbody>
</table>
23. Scripting Oxygen

Although Oxygen XML Editor is mostly intended to be a visual editing tool, the all platforms distribution is bundled with a scripts subfolder that contains scripts to automate and run various utilities from a command line.

To run any of these scripts, you are required to purchase a special scripting commercial license. Trial scripting licenses are also available, by request, for clients who are interested in testing the scripts for their particular workflows. Once you have a scripting license key available, you should copy the license key to a file named scriptinglicensekey.txt and save it in the main application directory (the parent directory of the "scripts" directory).

For more information about the Oxygen Scripting support, watch our Webinar: Automate XML processing with Oxygen XML Scripting.

DITA Validate and Check For Completeness

Attention: This script is bundled with the all platforms distribution of Oxygen XML Editor. To run the script, you are required to purchase a special scripting commercial license.

The Validate and Check For Completeness (on page 2941) action that is available on the toolbar of the DITA Maps Manager view provides the ability to validate a DITA map or a DITA Open Toolkit project file with a large array of settings. The settings dialog box has an Export settings option that can be used to export the settings to an XML configuration file. Once the settings are exported, you can use the validateCheckDITA.sh script to run a validation on a DITA map or DITA Open Toolkit project file and report the results in a separate XML document.

Sample Command-Line Arguments for the Validate and Check for Completeness Script:

```
sh scripts/validateCheckDITA.sh -i inputFile [-c contextId] [-s settingsFile] [-r reportFile]
```

A public example of using such a script as a GitHub action for reporting errors in pull requests on DITA project can be found here: https://github.com/oxygenxml/blog/blob/master/.github/workflows/workflow.yml. The GitHub action calls a Gradle script target named runValidation: https://github.com/oxygenxml/blog/blob/master/build/build.gradle.
Transform

Attention:

- This script is bundled with the all platforms distribution of Oxygen XML Editor. To run the script, you are required to purchase a special scripting commercial license.
- To execute a scenario based on WebHelp using this script, in addition to the scripting commercial license, you are required to purchase an Oxygen XML WebHelp license or an Oxygen Publishing Engine license.
- To execute a scenario based on Chemistry using this script, in addition to the scripting commercial license, you are required to purchase an Oxygen PDF Chemistry license or an Oxygen Publishing Engine license.

The Transform script (transform.sh) helps you to execute a transformation scenario. You can run the scenarios for the existing document types (frameworks) (on page 3219) without setting a scenarios file, but for others, you have to specify a specialized scenarios file or a project file that contains scenarios.

You can export transformation scenarios from Oxygen XML Editor into a specialized scenarios file by using the Export selected scenarios action from the Transformation Scenarios view or using the Export Global Transformation Scenarios action from the Options menu.

Arguments for the Transform Script

```
sh scripts/transform.sh -i inputFile -sn scenarioName [-s scenariosFile] [-v]
```

- **-i inputFile**
  
  The input file that the transformation scenario is applied to.

- **-sn scenarioName**
  
  The name of the transformation scenario to be executed.

- **-s scenariosFile**
  
  The name of a file that contains additional scenarios. It can be a specialized scenarios file or a project file that contains project transformation scenarios.

  The scenarios from this file are merged with the scenarios from the document types (frameworks) (on page 3219).

- **-v**
  
  This argument can be specified to activate verbose logging for DITA-OT and ANT scenarios. It is useful for debugging.
XML Refactoring

Attention:

- This script is bundled with the all platforms distribution of Oxygen XML Editor. To run the script, you are required to purchase a special scripting commercial license.
- To execute an XQuery refactoring operation using this script, in addition to the scripting commercial license, you are required to purchase a Saxon EE license.

The XML Refactoring script (xmlRefactoring.sh) can be used to execute XML refactoring operations (on page 840). You can run a refactoring operation by specifying the operation id of the operation. If, in addition to the refactoring operations provided by Oxygen XML Editor in the OXYGEN_INSTALL_DIR/refactoring folder and in framework configurations, you want to run a custom refactoring operation, you have to specify the directory that contains it, using the od (operations directory) argument.

Arguments for the XML Refactoring Script

```
sh scripts/xmlRefactoring.sh  -id operationId -i inputFilesOrDirectories [-f filesFilter]
[-od operationsDirectory] [-p param1=value1...] [-v]
```

- **-id operationId**
  
  The ID of the refactoring operation to be executed.

- **-i inputFilesOrDirectories**
  
  A list of space-separated input files or directories that the refactoring operation is applied to.

- **-f filesFilter**
  
  Specifies a filter for the input files by using a file pattern. For example, to restrict the operation to only analyze build files, you could use `build*.xml` for the file pattern.

- **-od operationsDirectory**
  
  A directory that contains additional refactoring operations.

- **-p param1=value1...**
  
  A list of space-separated pairs of a parameter’s name and value used by the refactoring operation.

- **-v**
  
  This argument can be specified to activate verbose logging.
DITA Translation Package Builder

Attention:
This script is bundled with the all platforms distribution of Oxygen XML Editor. To run the script, you are required to purchase a special scripting commercial license.

The DITA Translation Package Builder (translationPackageBuilder.sh) script helps you to build a translation package for DITA files that can be sent to translators. You can also extract the changed files back into your project once you receive the package back from the translators.

This script requires the DITA Translation Package Builder add-on to be installed in the all platforms distribution of Oxygen XML Editor. To install it the add-on, follow these instructions:

1. Go on the DITA Translation Package Builder plugin Releases page and download the latest translation-package-builder-{version}-plugin.jar package.
2. Unzip it inside {oxygenInstallDir}/plugins.

Note:
Do not create any intermediate folders. Afterwards, the file system should look like this:
{oxygenInstallDir}/plugins/translation-package-builder-{version}/plugin.xml

Examples for the DITA Translation Package Builder Script

Example: Generating a Milestone File

```
sh scripts/translationPackageBuilder.sh -gm -i ditamapFile [-m milestoneFile] [-verbose]
```

This action is the first one to use. It will generate a unique hash for each documentation resource. This information will be used by the second action to detect which files have been modified. A milestone file should be generated the first time you install this plugin and henceforth, after each package is sent to translators.

-gm
Requests the generation of a milestone file.

-i ditamapFile
The main DITA map file.

-m milestoneFile
The path to the milestone file. If missing, it is assumed that the milestone will be saved in the DITA map parent folder with the following name:
{ditamapName}_translation_milestone.xml.

-verbose
Generates a console log about the performed steps. It is useful for debugging.
Example: Creating a Package with the Modified Files to Send to Translation

```
sh scripts/translationPackageBuilder.sh -gp -i ditamapFile [-m milestoneFile] -p package.zip
[-verbose]
```

This action detects which files have been changed since the last generated milestone. These files are packed inside a ZIP file that can be sent to translators. After doing this, you can also generate a new milestone so that the next package will only contain new changes.

- **-gp**
  Requests the generation of a package with the modified files.

- **-i ditamapFile**
  The main DITA map file.

- **-m milestoneFile**
  The path to the milestone file. If missing, it is assumed that the milestone will be located in the DITA map parent folder with the following name: `{ditamapName}_translation_milestone.xml`.

- **-p package.zip**
  The path to the zip archive where all the modified files are collected.

- **-verbose**
  Generates a console log about the performed steps. It is useful for debugging.

Example: Applying a Translation Package Over a DITA Map

```
sh scripts/translationPackageBuilder.sh -ap -i ditamapFile -p package.zip [-verbose]
```

When the translated files arrive from the translator, you should open the DITA map that corresponds to the received language (e.g. open `dita-map-french.ditamap` if the package contains the french translation). Invoking this action will extract the changed files inside the map's directory.

- **-ap**
  Requests the application of a translation package over a DITA map.

- **-i ditamapFile**
  The main DITA map file that matches the received package language. For example, if the package contains topics translated into French, then this map is the French version of your DITA map.

- **-p package.zip**
  The path to the archive with all the translated files.

- **-verbose**
  Generates a console log about the performed steps. It is useful for debugging.
Batch Converter

Attention:
This script is bundled with the all platforms distribution of Oxygen XML Editor. To run the script, you are required to purchase a special scripting commercial license.

The Batch Converter script (batchConverter.sh) helps you to convert between the following formats:

- HTML to XHTML
- HTML to DITA
- HTML to DocBook4 / DocBook5
- Markdown to XHTML
- Markdown to DITA
- Markdown to DocBook4 / DocBook5
- Word to XHTML
- Word to DITA
- Word to DocBook4 / DocBook5
- Excel to DITA
- Confluence to DITA
- DocBook to DITA
- JSON to XML
- JSON to YAML
- XML to JSON
- YAML to JSON

This script requires the Oxygen Batch Documents Converter add-on to be installed in the all platforms distribution of an Enterprise edition of Oxygen XML Editor.

To install the add-on, follow these instructions:

1. Go on the Oxygen Batch Documents Converter plugin Releases page and download the latest oxygen-batch-converter-(version)-plugin.jar package.
2. Unzip it inside {oxygenInstallDir}/plugins.

Note:
Do not create any intermediate folders. Afterwards, the file system should look like this: {oxygenInstallDir}/plugins/oxygen-batch-converter-(version)/plugin.xml

Arguments for the Batch Converter Script

```
sh scripts/batchConverter.sh -i inputFiles -if inputFormat -o outputDirectory
-of outputFormat [-ss (true|false)] [-csd (true|false)] [-cs converterSettingsFile]
```

-i inputFiles
A list of space-separated input files or directories in file syntax form.

**-if inputFormat**

The format of the input files. The possible values are: html, markdown, word, confluence, docbook, excel, json, yaml, or xml.

**-o outputDirectory**

The output directory in file syntax form.

**-of outputFormat**

The format of the output files. The possible values are: xhtml, dita, docbook4, docbook5, json, yaml, or xml.

**-ss (true|false) [only for Word to DITA, HTML to DITA, Markdown to DITA, and DocBook to DITA conversions]**

Splits sections marked by titles or headings to separate files and create a DITA map. The possible values are true or false (default).

**-csd (true|false) [only for Markdown to DITA conversions]**

Creates short description elements from the first paragraph before the headings. Possible values are true or false (default).

**-cs converterSettingsFile**

A file that contains the Batch Documents Converter add-on preferences settings. It can be an xpr file that contains project options or an xml file that contains global options. If not specified, the operation uses the application's default settings.

**Confluence to DITA**

The Confluence to DITA conversion processes the HTML content generated by the Confluence export process. For exporting, login to your Confluence account and navigate to the specific space that you want to export. Go to Space Settings > Export space and choose to export it as HTML. The index.html file resulting from this process has to be provided in the inputFiles argument.

**Compile Framework Script**

⚠️ Attention:

This script is bundled with the all platforms distribution of Oxygen XML Editor.

A custom framework (on page 3219) (document type) can be created using a special XML descriptor file (on page 2134), either from scratch or by extending an existing built-in framework (such as DITA or DocBook) and then making modifications to it.

The Compile Framework Script (compileFrameworkScript.sh) helps you to compile the script into the *.framework file that represents the framework configuration. Although Oxygen XML Editor is now able
to automatically compile and load scripts, you might want to compile it yourself to obtain the resulting *.framework file if your framework runs on a different version of Oxygen XML Editor.

Arguments for the Compile Framework Script

```
sh scripts/compileFrameworkScript.sh -i "script1.exf" "script2.exf" "frameworksDir"
```

-i inputFiles

A list of space-separated Framework Extension Script Files (on page 2136) (*.exf) or directories in file syntax form. If a directory is specified, the Framework Extension Script Files (on page 2136) are searched for inside it and in its child directories.

XSLT Stylesheets Documentation

Attention:
This script is bundled with the all platforms distribution of Oxygen XML Editor. To run the script, you are required to purchase a special scripting commercial license.

You can generate documentation for XSLT Stylesheets from Oxygen XML Editor by using the Tools > Generate Documentation > XSLT Stylesheet Documentation main menu action. The settings dialog box has an Export settings option that can be used to export the settings to an XML configuration file. Once the settings are exported, you can use the scripts/stylesheetDocumentation.sh script to generate XSLT stylesheets documentation from the command line.

Sample Command-Line Arguments for the Generate XSLT Stylesheet Documentation Script

```
sh scripts/stylesheetDocumentation.sh xslFile [-cfg:configFile] | [-out:outputFile]
```

Related information
XML Schema Documentation (on page 3203)
WSDL Documentation (on page 3204)

XML Schema Documentation

Attention:
This script is bundled with the all platforms distribution of Oxygen XML Editor. To run the script, you are required to purchase a special scripting commercial license.

You can generate documentation for XML Schemas from Oxygen XML Editor by using the Tools > Generate Documentation > XML Schema Documentation main menu action. The settings dialog box has an Export settings option that can be used to export the settings to an XML configuration file. Once the settings are exported, you can use the scripts/schemaDocumentation.sh script to generate XML Schema documentation from the command line.
Sample Command-Line Arguments for the Generate XML Schema Documentation Script

```
```

Related information

XSLT Stylesheets Documentation (on page 3203)
WSDL Documentation (on page 3204)

WSDL Documentation

⚠️ **Attention:**

This script is bundled with the all platforms distribution of Oxygen XML Editor. To run the script, you are required to purchase a special scripting commercial license.

You can generate documentation for WSDL documents from Oxygen XML Editor by using the **Tools > Generate Documentation > WSDL Documentation** main menu action. The settings dialog box has an **Export settings** option that can be used to export the settings to an XML configuration file. Once the settings are exported, you can use the `scripts/wsdlDocumentation.sh` script to generate XML Schema documentation from the command line.

Sample Command-Line Arguments for the Generate WSDL Documentation Script

```
sh scripts/wsdlDocumentation.sh wsdlFile [-cfg:configFile] [-out:outputFile]
```

Related information

XSLT Stylesheets Documentation (on page 3203)
XML Schema Documentation (on page 3203)

XML Instance Generator

⚠️ **Attention:**

This script is bundled with the all platforms distribution of Oxygen XML Editor. To run the script, you are required to purchase a special scripting commercial license.

You can generate multiple XML documents from an XML Schema from Oxygen XML Editor by using the **Tools > Generate Sample XML Files** main menu action. The settings dialog box has an **Export settings** option that can be used to export the settings to an XML configuration file. Once the settings are exported, you can use the `scripts/xmlGenerator.sh` script to generate multiple XML instance files from the command line.

Sample Command-Line Arguments for the Generate Sample XML Files Script

```
sh scripts/xmlGenerator.sh path/to/config/file [-verbose]
```
Extended Version of the Script and its Arguments

```
sh scripts/xmlGenerator.sh [[path_to_config_file] [-s XML_schema_path -r root [-n ns] [-o output_dir]
[-f instance_name] [-i num_of_instances]] [-verbose]]
```

**path_to_config_file**

The path to the file that contains the configuration to be used.

**-s XML_schema_path**

The path to the XML schema to be used for generating the XML file(s).

**-n ns**

The namespace used for the XML namespace declaration.

**-r root**

The root element for the generated file(s).

**-o output_dir**

The output directory to be used for storing the generated file(s).

**-f instance_name**

The pattern name to be used for the generated file(s). It is usually the name plus extension.

**-i num_of_instances**

The number of XML files to be generated.

**-verbose**

This argument can be specified to activate verbose logging.

---

**Note:**

Any value specified by the -s, -n, -r, -o, -f, or -i arguments overrides the corresponding value from the configuration file, if that file is specified in the path_to_config_file argument.

---

**Flatten XML Schema**

---

**Attention:**

This script is bundled with the all platforms distribution of Oxygen XML Editor. To run the script, you are required to purchase a special scripting commercial license.

You can flatten an XML schema that contains multiple includes and redefines to a single schema file from Oxygen XML Editor by using the Tools > Flatten Schema main menu action. You can use the equivalent scripts/flattenSchema.sh script to flatten an XML schema from the command line.
Sample Command-Line Arguments for the Flatten Schema Script

```
  [-flattenImports:<boolean_value>]
  | [-useCatalogs:<boolean_value>]
  [-flattenCatalogResolvedImports:<boolean_value>] [-verbose]
```

Compare Directories

⚠️ **Attention:**

This script is bundled with the all platforms distribution of Oxygen XML Editor. To run the script, you are required to purchase a special scripting commercial license.

The **Compare Directories** script (compareDirs.sh) can be used to compare two directories and get the comparison results in various formats.

Arguments for the Compare Directories Script

```
```

**firstDirPath**

Mandatory argument that specifies the first directory path (it can also be provided as a URL using 'file://' protocol).

**secondDirPath**

Mandatory argument that specifies the second directory path (it can also be provided as a URL using 'file://' protocol).

**baseDirPath**

Optional argument that specifies the path of the base directory that the other two directories will be compared against in a 3-way comparison (it can also be provided as a URL). If present, it must appear immediately after the first two mandatory arguments.

**-if includeFilesFilter**

Use this argument to only include files that match the specified pattern in the comparison (e.g. .xml, .json). Default value = *.

**-ia includeArchives**

If set to true, files from archives are included in the comparison. Default value = false.

**-ef excludeFilesFilter**
Use this argument to exclude files that match the specified pattern from the comparison (e.g. *.jpg).

**-ed excludeSubdirsFilter**

Use this argument to exclude sub-directories that match the specified pattern from the comparison (e.g. .svn, _svn, .git).

**-cm comparisonMode**

Specifies the comparison mode. There are three modes available: **content**, **binary**, and **timestamp**. Default value = **content**.

**-alg comparisonAlg**

Specifies the algorithm to be used for the comparison. Possible values: **auto**, **chars**, **words**, **lines**, **syntax_aware**, **xml_fast**, and **xml_accurate**. Default value = **auto**.

**-als algStrength**

Specifies the strength of the algorithm to be used for the comparison. Possible values: **low**, **medium**, **high**, and **very_high**. Default value = **medium**.

**-iws ignoreWS**

If set to **true**, whitespaces are ignored if differences consist only of whitespaces. Default value = **false**.

**-ipi ignorePI (only for the XML-aware algorithms)**

If set to **true**, processing instructions are ignored in the comparison. Default value = **false**.

**-icm ignoreComments (only for the XML-aware algorithms)**

If set to **true**, comments are ignored in the comparison. Default value = **false**.

**-idt ignoreDocType (only for the XML-aware algorithms)**

If set to **true**, DOCTYPE sections are ignored in the comparison. Default value = **false**.

**-itn ignoreText (only for the XML-aware algorithms)**

If set to **true**, text content is ignored in the comparison. Default value = **false**.

**-ins ignoreNS (only for the XML-aware algorithms)**

If set to **true**, namespaces are ignored in the comparison. Default value = **false**.

**-ind ignoreNSDecl (only for the XML-aware algorithms)**

If set to **true**, namespace declarations are ignored in the comparison. Default value = **false**.

**-inp ignorePrefixes (only for the XML-aware algorithms)**

If set to **true**, prefixes are ignored in the comparison. Default value = **false**.

**-iao ignoreAttrOrder (only for the XML-aware algorithms)**

If set to **true**, the order of attributes is ignored in the comparison. Default value = **false**.

**-iee ignoreExpStateForEmptyElems (only for the XML-aware algorithms)**
If set to `true`, the expansion state for empty elements is ignored in the comparison. Default value = `false`.

**-merge mergeOperation**

If set to `true`, a merge operation is invoked after the comparison. Default value = `false`.

**Notes:**

- This argument is considered only for 3-way comparisons (i.e. only if the `baseDirPath` argument is provided).
- The merge operation is similar to the same process in any versioning system. Following the comparison between the first and second directories (relative to the base folder), all the differences of the type **incoming** are considered and the content of the first directory is updated accordingly.
- Conflicting changes are not addressed.
- After the comparison, a report is created that provides details about the changes that were made.

**-mergeout outputDirPathForMerge**

Invokes a merge operation after the comparison and also allows you to specify the output directory path for the merge operation. For example, it allows you to specify a specific existing or new directory where the results of the merge operation is saved, other than the first directory path for the comparison (which is what happens when using only the `-merge` argument). The path of the directory can also be provided as a URL using `file://` protocol. This argument and the `-merge` argument are not dependent on each other.

**-out outputFormat**


**Notes:**

- If you choose to save/redirect the console output to a file, this argument establishes the type of the output file and its content is formatted accordingly. If you skip specifying the "/grouped" or the "/raw" qualifiers, "/grouped" takes precedence.
- If you choose the `html` or `htm` output format, it is recommended that you also choose to save/redirect the console to the specified HTML file to view the comparison result in your preferred browser.
- The "ifcr" qualifier for the `html` or `htm` values is considered only if the `-outfile` argument is also present. IFCR is an acronym for Include File Comparison Reports and it means that, along with generating the directory comparison report,
separate file comparison reports will be generated for all modified file pairs. These reports are available through links from the main report and are saved to a specific directory based on the value provided by the `outfile` argument. It will have the same parent directory and the same name as the `outputFile` plus `-OXY-FC-REPORTS` added to the end of its name.

- The `html` value, as well as the `grouped`, `raw`, or `ifcr` qualifiers, are not considered if the `--merge` argument is present.

```
(outfile outputFile)
```

Specifies the path for an output file to save the comparison results, instead of presenting them in the console. The content of the output file is formatted according to the `-out` argument. The output file path can also be provided as a URL using `file://` protocol.

```
-help | --help | -h | --h
```

Displays help text.

**Notes:**

- For boolean arguments, it is not necessary to provide the "true" value. Their presence in the argument list is equivalent to setting their value to "true" (and their absence from the argument list is equivalent to setting their value to "false"). However, constructs of the form `bool_option true|false` are accepted and interpreted accordingly.
- File markers used in reports are as follows: M = modified, O1 = only found in 1st directory, O2 = only found in 2nd directory.

**Examples of Compare Directories Script**

**Example 1: Compare Directories and Include Archives While Excluding JPEGs**

The following command results in archives being included in the comparison, while JPEGs are excluded:

```
sh scripts/compareDirs.sh dir1 dir2 -ia true -ef *.jpg
```

**Example 2: Compare Directories Only Including XML Files While Excluding Comments and the Attribute Order**

The following command only includes XML files (even from archives) in the comparison, while ignoring the comments and attribute order:

```
sh scripts/compareDirs.sh dir1 dir2 -if *.xml -ia -iao -icm
```

**Example 3: Compare Directories Only Including XML Files While Excluding Comments and the Attribute Order**
The following command redirects the comparison results to a JSON file named "results.json", with "raw" mode formatting:

```
sh scripts/compareDirs.sh dir1 dir2 -out json/raw > results.json
```

**Example 4: Compare Directories and Generate Comparison Report**

It is possible to generate a report in the form of an HTML file that contains the results of the comparison. The following command compares the directories and redirects the console to the specified HTML file to view the comparison results:

```
sh scripts/compareDirs.sh dir1 dir2 -out html -outfile results.html
```

**Figure 811. Example of an HTML Report for Directory Comparison**

![Example of an HTML Report for Directory Comparison](image)

**Resources**

For more information about the file comparison script and how to generate comparison reports in various formats, see the following resources:

- **Webinar:** The New Oxygen Compare and Merge Scripts.
- **Video:** Generating Directory Comparison Reports Using Command-Line Scripts.

**Related information**

- Compare Directories Tool (on page 494)
- Compare Files Script (on page 3210)

**Compare Files**

⚠️ **Attention:**

This script is bundled with the all platforms distribution of Oxygen XML Editor. To run the script, you are required to purchase a special scripting commercial license.
The **Compare Files** script (`compareFiles.sh`) can be used to compare files (2-way or 3-way) and get the comparison results in various formats.

### Arguments for the Compare Files Script

```bash
sh scripts/compareFiles.sh firstFilePath secondFilePath [baseFilePath] [-ct contentType]
[-alg comparisonAlg] [-als algStrength] [-lws ignoreWS] [-ipi ignorePI] [-icm ignoreComments]
[icd ignoreCDATA] [-idt ignoreDocType] [-itn ignoreText] [-ins ignoreNS] [-ind ignoreNSDecl]
[-inp ignorePrefixes] [-iao ignoreAttrOrder] [-iee ignoreExpStateForEmptyElems] [-enx
XPathExprToExcludeNodes] [-out outputFormat]] [-help | --help | -h | --h]
```

**firstFilePath**

Mandatory argument that specifies the first file path (it can also be provided as a URL).

**secondFilePath**

Mandatory argument that specifies the second file path (it can also be provided as a URL).

**baseFilePath**

Optional argument that specifies the path of the base file that the other two files will be compared against in a 3-way comparison (it can also be provided as a URL).

**-ct contentType**

Specifies the content type of the files to be compared. Possible values (based on known extensions of some of the most common file types): `.xml`, `.dtd`, `.css`, `.rnc`, `.xquery`, `.json`, `.yaml`, `.java`, `.js`, `.c`, `.cpp`, `.pl`, `.py`, `.php`, `.sql`, `.bat`, `.sh`, `.properties`, `.txt`. The option is used to force the file handling to the specific type of file. Otherwise, the file extension is auto-detected.

**-alg comparisonAlg**

Specifies the algorithm to be used for the comparison. Possible values: `auto`, `chars`, `words`, `lines`, `syntax_aware`, `xml_fast`, and `xml_accurate`. Default value = `auto`.

**-als algStrength**

Specifies the strength of the algorithm to be used for the comparison. Possible values: `low`, `medium`, `high`, and `very_high`. Default value = `medium`.

**-lws ignoreWS**

If set to `true`, whitespaces are ignored if differences consist only of whitespaces. Default value = `false`.

**-ipi ignorePI** (only for the XML-aware algorithms)

If set to `true`, processing instructions are ignored in the comparison. Default value = `false`.

**-icm ignoreComments** (only for the XML-aware algorithms)

If set to `true`, comments are ignored in the comparison. Default value = `false`.

**-idt ignoreDocType** (only for the XML-aware algorithms)

If set to `true`, doctype declarations are ignored in the comparison. Default value = `false`. 

If set to `true`, DOCTYPE sections are ignored in the comparison. Default value = `false`.

- **-itn ignoreText (only for the XML-aware algorithms)**
  If set to `true`, text content is ignored in the comparison. Default value = `false`.

- **-ins ignoreNS (only for the XML-aware algorithms)**
  If set to `true`, namespaces are ignored in the comparison. Default value = `false`.

- **-ind ignoreNSDecl (only for the XML-aware algorithms)**
  If set to `true`, namespace declarations are ignored in the comparison. Default value = `false`.

- **-inp ignorePrefixes (only for the XML-aware algorithms)**
  If set to `true`, prefixes are ignored in the comparison. Default value = `false`.

- **-iao ignoreAttrOrder (only for the XML-aware algorithms)**
  If set to `true`, the order of attributes is ignored in the comparison. Default value = `false`.

- **-iee ignoreExpStateForEmptyElems (only for the XML-aware algorithms)**
  If set to `true`, the expansion state for empty elements is ignored in the comparison. Default value = `false`.

- **-enx XPathExprToExcludeNodes**
  Specifies an XPath expression to exclude certain nodes from the comparison.

- **-merge mergeOperation**
  If set to `true`, a merge operation is invoked after the comparison. Default value = `false`.

---

**Notes:**

- This argument is considered only for 3-way comparisons (i.e. only if the `baseFilePath` argument is provided).
- The merge operation is similar to the same process in any versioning system. Following the comparison between the first and second files (relative to the base file), all the differences of the type `incoming` are considered and the content of the first file is updated accordingly.
- If conflicting changes are detected, the merge operation is aborted and the first file remains unchanged.
- After the comparison and merge, a report is created that provides some details about the changes that were made.

---

- **-mergeout outputDirPathForMerge**
  Invokes a merge operation after the comparison and also allows you to specify the output directory path for the merged file. Instead of directly affecting the first compared file (which is what happens when using only the `-merge` argument), a new file is created with the same name
as the first file and it is saved in the specified directory. The path of the output directory can also be provided as a URL. This argument and the -merge argument are not dependent on each other.

-out outputFormat

Specifies the format of the output. Possible values: yaml, json, xml, html, htm, html/inlineCSS, or htm/inlineCSS. Default value = yaml.

Notes:

• If you choose to save/redirect the console output to a file, this argument establishes the type of the output file and its content is formatted accordingly.
• If you choose any of the html, html/inlineCSS, htm, or htm/inlineCSS output formats, it is recommended that you also choose to save/redirect the console to the specified HTML file to view the comparison result in your preferred browser.
• The inlineCSS qualifier for the html and htm values implies that the CSS-based generated HTML code is more suitable to be directly inserted in emails (as most email clients only accept inline CSS styling for HTML emails.
• The html and htm values (with or without the inlineCSS qualifier) are not considered if the -merge argument is present.

-outfile outputFile

Specifies the path for an output file to save the comparison results, instead of presenting them in the console. The content of the output file is formatted according to the -out argument. The output file path can also be provided as a URL.

-help | --help | -h | --h

Displays help text.

Note:

For boolean arguments, it is not necessary to provide the "true" value. Their presence in the argument list is equivalent to setting their value to "true" (and their absence from the argument list is equivalent to setting their value to "false"). However, constructs of the form bool_option true|false are accepted and interpreted accordingly

Examples of Compare Files Script

Example 1: Compare Files and View Results in XML Format

The following command compares the files (ignoring the namespaces and prefixes) and redirects the console output to the results.xml file (XML-formatted):

```
sh scripts/compareFiles file1 file2 -ins -inp -ind -out xml > results.xml
```

Example 2: Compare Files with Line by Line Algorithm
The following command compares the files using the **lines** algorithm and ignores whitespaces:

```
sh scripts/compareFiles.sh file1 file2 -alg lines -iws
```

**Example 3: Compare Files and Generate Comparison Report**

It is possible to generate a report in the form of an HTML file that contains the results of the comparison. The following command compares the files and redirects the console to the specified HTML file to view the comparison results:

```
sh scripts/compareFiles.sh file1 file2 -out html -outfile outFileName.html
```

**Figure 812. Example of File Comparison Report in HTML Format**

![Comparison Report](image.png)

**Resources**

For more information about the file comparison script and how to generate comparison reports in various formats, see the following resources:

- **Webinar**: The New Oxygen Compare and Merge Scripts.
- **Video**: Generating File Comparison Reports Using Command-Line Scripts.

**Related information**

Compare Directories Script (on page 3206)
**Format and Indent Files**

**Attention:**
This script is bundled with the all platforms distribution of Oxygen XML Editor. To run the script, you are required to purchase a special scripting commercial license.

The **Format and Indent Files** script (batchFormatAndIndent.bat/batchFormatAndIndent.sh) can be used to format and indent multiple files at once.

**Arguments for the Format and Indent Files Script**

```
batchFormatAndIndent -i inputFilesAndDirs [-f filesFilter] [-s formattingSettingsFile] [-r] [-ih] [-v]
```

- **-i inputFilesAndDirs**
  The input files and directories.

- **-f filesFilter**
  A filter for the input files, specified by using a file pattern (e.g. *.xml, t_*.dita).

- **-s formattingSettingsFile**
  A file that contains formatting settings. It can be an .xpr file that contains project options or an .xml file that contains global options. If not specified, the operation uses the application's default settings.

- **-r**
  Use this argument if the operation should be performed recursively for the specified input directories.

- **-ih**
  Use this argument if you want the operation to also format and indent hidden files.

- **-v**
  Activates verbose logging. It is useful for debugging purposes.
Glossary

Active Cell

Active cell refers to the selected cell where data is entered when you begin typing. Only one cell is active at a time. The active cell is bounded by a heavy border.

Alternate CSS Style

The Alternate CSS Style refers to the choices in the bottom half of Styles drop-down menu (on the toolbar) that makes it easy to apply style changes to your documents as they appear in Author mode and the output without having to edit the CSS stylesheets. By default, the alternate styles are applied like layers, they are merged sequentially with the main CSS style (on page 3220), and you can activate any number of them. However, if you deselect the Enable multiple selection of alternate CSSs option (on page 146) in the CSS subtab of the Document Type configuration dialog box (on page 140), the alternate styles are treated like main CSS styles (on page 3220) and you can only select one at a time.

For more information, see Configuring and Managing Multiple CSS Styles for a Framework (on page 2145).

Anchor

An Anchor is used in various types of links to take the user to a specific location within the target document. It is designated in a URL or in the value of the @href attribute with a # symbol followed by the anchor that is defined in a target ID (for example href="MyTopic.dita#anchor").

Apache Ant

Apache Ant (Another Neat Tool) is a software tool for automating software build processes.

Block Element

A block element is intended to be visually separated from its siblings, usually vertically. For instance, paragraphs and list items are block elements. It is distinct from a inline element, which has no such separation.

Bookmap

A bookmap is a specialized DITA map used for creating books. A bookmap supports book divisions such as chapters and book lists such as indexes.
Callout

A **callout** is a string of text inside a graphic and is connected to a specific location in a document by a line. Oxygen XML Editor uses callouts to present comments and other types of review modifications.

Canonicalize

To **canonicalize** something means to convert it to a standard format that everyone generally uses. When using the term with regard to XML, it refers to the process of converting data that has more than one possible representations into a standardization that conforms to the specification of an XML document or document subset. It is helpful for applications that require the ability to test whether or not the content of an XML document or subset has been changed.

Content Completion Assistant

The **Content Completion Assistant** refers to a very helpful mechanism in Oxygen XML Editor that offers a list of proposed items that could be inserted at the current location, depending on the current context, editing mode, and type of document. It also tries to determine the most logical choice in the current editing context and displays that proposal at the beginning of the list.

For more information about this feature and how to invoke it, depending on your editing context, see the following:

- Content Completion Assistant in Author Mode *(on page 616)*
- Content Completion Assistant in Text Mode *(on page 532)*
- Content Completion Assistant in Grid Mode *(on page 586)*
- Content Completion in XSLT Stylesheets *(on page 892)*
- Content Completion in Ant Build Files *(on page 937)*
- Content Completion in XML Schema *(on page 992)*
- Content Completion in XQuery *(on page 1035)*
- Content Completion Assistance in WSDL Documents *(on page 1053)*
- Content Completion in CSS Stylesheets *(on page 1077)*
- Content Completion in LESS Stylesheets *(on page 1081)*
- Content Completion in Relax NG Schemas *(on page 1089)*
- Content Completion in NVDL Schemas *(on page 1105)*
- Content Completion in JavaScript Documents *(on page 1187)*
- Content Completion in Schematron Documents *(on page 1209)*
- Content Completion in SQF *(on page 1248)*

Dependencies

The word **dependencies** in XML documents refers to resources or references that are relied upon in a particular scope (e.g. the scope of the entire project or a particular DITA map) for a selected resource. This is helpful if you want to see where the resource is used or referenced in the entire hierarchy of the project or map. For example, you might want to look for the dependencies of an XML document to find all the instances where it is referenced throughout the hierarchy of a project or DITA map.
Dockable

A Dockable window is one that can be moved and resized, and either floated or pinned to a location, allowing you to configure the workspace according to your preferences.

Document Fragment

A document fragment represents a portion of an XML document's tree of nodes or content.

Document Type Association

In general terms, a Document Type Association is a set of rules that associate a document type with a framework (on page 3219). In Oxygen XML Editor, Document Type Association also specifically refers to a preferences page (on page 138) where you can create new custom frameworks or edit existing ones. Note that frameworks (document types) that come built-in with Oxygen XML Editor are read-only, but you can Extend (on page 138) or Duplicate (on page 138) them to configure them as custom frameworks.

DITA Map

A DITA map is a component of the DITA framework (on page 3219) that provides the means for a hierarchical collection of DITA topics that can be processed to form an output. Maps do not contain the content of topics, but only references to them. These are known as topic references. Usually, the maps are saved on disk or in a CMS with the extension .ditamap.

Maps can also contain relationship tables that establish relationships between the topics contained within the map. Relationship tables are also used to generate links in your published document.

You can use your map or bookmap (on page 3216) to generate a deliverable using an output type such as XHTML, PDF, HTML Help, or Eclipse Help.

DITA Open Toolkit

DITA Open Toolkit is an open-source publishing engine for content authored in the Darwin Information Typing Architecture. It is a vendor-independent, open-source implementation of the DITA standard, released under the Apache License, Version 2.0.

The toolkit supports all versions of the OASIS DITA specification, including 1.0, 1.1, 1.2, and 1.3.

DITA-OT

Related Information:

http://www.dita-ot.org/
DITA-OT-DIR

*DITA-OT-DIR* refers to the default directory that is specified for your DITA Open Toolkit distribution in the Options > Preferences > DITA preferences page (on page 269).

For example, if you are using DITA-OT 3.7.0 that comes bundled with Oxygen XML Editor, the default directory is: `[OXYGEN_INSTALL_DIR]/frameworks/dita/DITA-OT3.x`. You can also specify a custom directory.

Foldable Element

A *foldable element* refers to elements that can be collapsed and expanded in Oxygen XML Editor. Foldable elements are marked with a small triangle (▼ / ▲) on the left side of the editor panel and you can use that triangle to quickly collapse or expand them. This feature is helpful when you are working with large documents and you want to temporarily hide blocks of content. You can right-click the triangle to access additional collapse and expand actions (Collapse Other Folds, Collapse Child Folds, Expand Child Folds, Expand All).

Framework

A *framework* refers to a package that contains resources and configuration information to provide ready-to-use support for an XML vocabulary or document type. A framework is associated to an XML document type according to a set of rules. It also includes a variety of settings that improve editing capabilities for its particular file type. Oxygen XML Editor includes a Document Type Configuration Dialog Box (on page 140) that allows you to define the set of rules and customize various authoring mechanisms for new or existing frameworks.

For advanced details about customizing your own framework, see the Creating and Configuring Custom Frameworks (on page 2133) section.

Global Options

*Global Options* refers to the storage option (on page 311) in the Oxygen XML Editor preference pages (Options > Preferences). If you select Global Options (on page 312), the options in that particular preferences page are stored locally on your computer and are not accessible to other users (unless you export them into an XML options file (on page 313) that can then be shared).

Hierarchy

In the Oxygen Resource Hierarchy view, the word *hierarchy* refers to a set of components (outgoing references) that are displayed as a tree, recursively, starting from the selected XML file (shown as the root). It makes it easy to navigate through the hierarchy (tree) to find specific data.

IDML

*IDML* is an abbreviation for Adobe InDesign Markup files.
Inline Element

An inline element is intended to be displayed in the same line of text as its siblings or the surrounding text. For instance, strong and emphasis in HTML are inline elements. It is distinct from a block element, which is visually separated from its siblings.

Java Archive

Java Archive (JAR) is an archive file format. JAR files are built on the ZIP file format and have the .jar file extension. Computer users can create or extract JAR files using the jar command or an archive tool.

Key Space

The concept of a Key Space in DITA refers to a set of all possible keys that can be used in a DITA map structure. A Key Space is established when a root map (on page 3223) defines a set of effective key bindings. When Oxygen XML Editor processes key references, it determines the effective binding of a given key to a resource in the context of the specified root map (on page 2914).

Keystore

A Keystore is an encrypted file that contains private keys and certificates. There are two types of keystores that are supported in Oxygen XML Editor:

- Java Key Store (JKS)
- Public-Key Cryptography Standards version 12 (PKCS-12)

Main CSS Style

The Main CSS Style refers to the selection in the top half of the Styles drop-down menu (on the toolbar) that makes it easy to quickly change the look of your documents as they appear in Author mode and the output without having to edit the CSS stylesheets. The main CSS applies to the whole document and you can also select one or more alternate styles (on page 3216) (listed in the bottom half of the drop-down menu) that behave like layers and are merged sequentially with the main CSS style.

For more information, see Configuring and Managing Multiple CSS Styles for a Framework (on page 2145).

Main File

A Main File typically refers to the root of an imported or included tree of modules and this support helps you simplify the configuration and development of XML projects. For more information, see the Contextual Project Operations Using 'Main Files' Support (on page 418) section.
Named User

**Named User** is defined as an individual full or part-time employee who is authorized by You (the individual or entity who owns the rights for Oxygen XML Editor) to use the software regardless of whether or not the individual is actively using the software at any given time. To avoid any doubt, **Named User** licenses cannot be shared among multiple individuals and separate **Named User** licenses must be purchased for each individual user.

A **Named User** license may not be reassigned to another employee except in the following circumstances:

- (a) Upon termination of the **Named User’s** employment with your company.
- (b) Permanent reassignment of a **Named User** to a position that does not involve the use of the Software.

For example, suppose Jane has been assigned an **Oxygen** license and she leaves your company. When she leaves, you can simply reassign her license to John, her replacement. In the event that you do reassign the **Named User** license in accordance with the restrictions above, you do not need to notify Syncro of such a reassignment.

**Note:**
This definition is taken from the Oxygen XML Editor **End-User License Agreement**.

Oxygen Publishing Template

**Oxygen Publishing Template** defines all the aspects related with the **look and feel** (**layout and styles**) for the **WebHelp Responsive** output.

The template is self-contained and packed as a ZIP archive making it easy to share with others. It represents the main method for customizing the **WebHelp Responsive** output.

**Related Information:**
[Publishing Template Package Contents for WebHelp Responsive Customizations](on page 1606)

Perspective

In Oxygen XML Editor, a **perspective** refers to an interface layout geared towards a specific editing environment. Each **perspective** includes a unique set of interface objects, toolbars, views, and features.

You can change the **perspective** by selecting the respective icon ( ![perspectives icon](image.png) ) in the top-right corner of Oxygen XML Editor or by selecting the **perspective** from the **Window > Open Perspective** menu.

The **perspectives** that are available in Oxygen XML Editor are:
• Editor (on page 344) - The most commonly used perspective and it is used to edit XML documents.

• DITA (on page 346) - Provides an editing environment with default side-views and other interface components that are optimal for working with DITA projects.

• XSLT Debugger (on page 347) - Used to detect problems in an XSLT transformation by executing the process step by step in a controlled environment.

• XQuery Debugger (on page 349) - Used to detect problems in an XQuery transformation process by executing the process step by step in a controlled environment

• Database (on page 350) - Used to browse and manage databases.

Plugin

In Oxygen XML Editor, a plugin is a component that adds extended functionality using a series of extension points and can be installed as an add-on. For more information, along with a full list of add-ons that are officially supported for Oxygen XML Editor, see Oxygen XML Add-on Repositories.

For more information, see the following topics:

• Installing and Updating Add-ons (on page 120)
• Automatic and Manual Methods for Installing Plugins (on page 2413)
• Packing and Deploying Plugins as Add-ons (on page 2445)
• Add-ons Preferences (on page 136)
• Extending Oxygen XML Editor with Plugins (on page 2411)
• General Configuration of an Oxygen XML Editor Plugin (on page 2411)

Pretty-Print

Pretty-print refers to formatting and indenting the source code in Text mode to make the content easier to view and analyze. The formatting actions that are available in Oxygen XML Editor include:

• Format and Indent Element - Available in the Source submenu of the contextual menu for the current element.

• Format and Indent - Available on the toolbar for the entire current document.

• Format and Indent Files - Available in the contextual menu of the Project view (on page 402) for one or more selected files.

Project Options

Project Options refers to the storage option (on page 311) in the Oxygen XML Editor preference pages (Options > Preferences). If you select Project Options (on page 312), the options in that particular preferences page are stored at project level in the project file (.xpr), which can easily be shared with other users (on page 312).
QName

*QName* stands for "qualified name" and defines a valid identifier for elements and attributes. *QNames* are used as URI references to reference particular elements or attributes within XML documents.

Quick Assist

The *Quick Assist* feature gives you easy access to some of the most commonly used actions for the specific type of document you are editing. If one or more actions are available in the current context, they are accessible via a yellow bulb help (💡) placed at the current line in the stripe on the left side of the editor in *Text* mode. You can also invoke the quick assist menu by using the `Alt + 1` (`Meta + Alt + 1` on macOS) keyboard shortcuts.

Quick Fix

The *Quick Fix* support in Oxygen XML Editor helps you resolve errors that appear in an XML document by offering proposals to fix problems such as missing required attributes or invalid elements. *Quick Fixes* are available in *Text* mode and *Author* mode and they can be presented and activated in several ways.

- When hovering over an area of text where a validation error or warning occurs, the *Quick Fix* proposals can be presented as links in a tooltip pop-up window.
- When hovering over an error or warning in *Author* mode, the *Quick Fix* proposals are presented in a small drop-down menu.
- If you place the cursor in the highlighted area where a validation error or warning occurs, a *Quick Fix* icon (💡) is displayed in the stripe on the left side of the editor. Clicking that icon will allow you to select from the available proposals.
- If you place the cursor in the highlighted area where a validation error or warning occurs, you can also access the *Quick Fix* menu by pressing `Alt + 1` (`Command + Alt + 1` on macOS) on your keyboard.

Oxygen XML Editor also provides support for defining and customizing a library of *Quick Fixes* using the Schematron language (on page 1236).

Root Map

A *Root Map* (or main map) specifies a *DITA map* (on page 3218) that defines a hierarchical structure of submaps that are contained within the root map. Essentially, the root map defines a scope and provides the mechanism to allow your defined keys to be propagated throughout the entire map structure (this mechanism is also known as a *key space* (on page 3220)).

In Oxygen XML Editor, the *DITA Maps Manager* includes an option on its toolbar where you can easily specify the root map (on page 2901), but there are also several other ways to select or change the root map (on page 2914).
**Space-Preserved Element**

A spaced-preserved element refers to elements that require white spaces and line endings to be preserved (for example, DITA `<codeblock>` and `<pre>` elements).

**Subject Scheme Map**

A Subject Scheme Map allows you to create custom controlled attribute values and to manage metadata. Subject scheme maps use a key definition to define a collection of controlled values rather than a collection of topics. The highest level of map that uses the set of controlled values must reference the subject scheme map where those controlled values are defined.

A controlled value is a keyword that can be used as a value in a metadata attribute. For example, the `@audience` metadata attribute may take a value that identifies the user group associated with a particular content unit (for medical equipment, that might include therapist, oncologist, surgeon, radiologist, and so on). In a subject scheme map, you can define a list of these audience values and you can then use these values to profile your content. For more information, see Customizing Profiling Values with a Subject Scheme Map (on page 3151).

**Track Changes**

The Track Changes feature allows you to review changes that you or other authors have made and then accept or reject them. You can also manage the visualization mode of the tracked changes, add comments to changes, and mark them as being done. These actions are easily accessible from contextual menus, the toolbar, or the Review view (on page 664).

For more information about this feature, see Managing Tracked Changes (on page 643).

**WebHelp Output Directory**

`WebHelp_OUTPUT_DIR` refers to the output directory where WebHelp transformation files will be generated.

The output directory can be specified using the Output Directory text field in the Output tab of the transformation scenario dialog box.

When running the WebHelp transformation from a command line, the output directory can be specified using the `-o` or `--output` option.

**Working Set**

A Working Set refers to a set of files that will be used for the scope of search and refactoring operations. Many of the search and refactoring wizards include a step where you can specify the scope for the operation and you can choose one or more working sets to restrict the scope to that specified set of files.
**XML Catalog**

An *XML Catalog* maps a system ID or a URI reference for a resource (stored either remotely or locally) to a local copy of the same resource. Whenever XML processing relies on external resources (such as referenced schemas and stylesheets), the use of an *XML Catalog* becomes a necessity when Internet access is not available or the connection is slow.

Oxygen XML Editor includes default global catalogs as well as default catalogs for each of the built-in frameworks *(on page 3219)*, and you can also create your own. Oxygen XML Editor uses these *XML Catalogs* to resolve references for document validation and transformations. For more information, see *Working with XML Catalogs* *(on page 826)*.
Index

Special Characters
:after pseudo-element  2409
:before pseudo-element  2409

A
Add fonts to built-in FO processor  1526, 1528
Add hyphenation libraries to built-in DITA-OT FO processor  1996
Add hyphenation libraries to built-in FO processor  1529
Add PDF image support to built-in DITA-OT FO processor  1996
Add PDF image support to built-in FO processor  1529
Add Schematron Quick Fix  1236
Add-on  3222
Add-on preferences  136
Add-ons  2514, 3222
Batch converter  2533
CGM image support  2579
Check for add-on updates  120
Content Fusion Connector  2527
DITA Outgoing References View  2540
DITA prolog updater  2538
DocBook checker  2578
Emmett  2557
Git support  2515
ICU4J library  2564
Install new add-ons  120
Manage add-ons  120
Saxon XSLT and XQuery transformer  2563
Smart Autocomplete  2550
Technical Writer Helper  2561
Terminology checker  2541
Translation package builder  2548
Translator Helper  2547
Vale linter  2546
XSpec Helper View  2562

Adding custom views  2417
Adding media resources in Author Mode  747
Additional Framework plugin extension  2422
Additional languages plugin extension  2425
Additional XProc engine plugin extension  2423
Addons  2514
AI image rendering in Author mode  722
Associate schema to an XML document
816
Attributes view in Author Mode
627
Attributes view in Design mode
995
Attributes view in Text mode
542
Author Action dialog box
148
Author editing mode
354, 588
  Adding media resources
  747
Apply profiling attributes
  673
Apply profiling condition sets
  677
Attributes view
  627
Bidirectional text
  752
Content completion
  616
Contextual menu actions
  759
Create/Edit profiling attributes
  670
Create/Edit profiling condition sets
  675
Displaying referenced content
  595
Displaying the markup
  594
Drag and Drop
  612
Editing attributes
  610
Editing content
  599
Editing XML markup
  601
Elements view
  633
Entities view
  547, 634
Folding
  611
Form Controls
  758
Generating IDs
  755
Handling whitespaces
  598
Image Map Editor
  725
DITA
  725, 2980
DocBook
  731
TEI
  736
XHTML
  742
Image rendering
  719
AI images
  722
CGM images
  721
EMF images
  723
EPS images
  722
JAI images
  722
PSD images
  721
Inserting images
  719
MathML
  749
MathML equations in HTML output
Auto recovery
384
AutoCorrect
460
Add dictionaries
461
AutoCorrect preferences
193
Automated tests
2446
Automatic Spell Check
456
Automatic validation
775
Automatic validation in JSON
1115
Automatically correct misspelled words
460
Available memory
343
Backup file
384
BaseX database connection
2063
BaseX database contextual menu actions
2064
BaseX XQJ connection
2066
Batch converter add-on
2533
Batch transformation
414, 1553
Batch validation
414
Berkeley DB XML database connection
2037
Berkeley DB XML database contextual menu actions
2039
Berkeley DB XML debugging

Bidirectional text in Author mode
752
Bidirectional text in Grid mode
586
Bidirectional text in Text mode
564
Browsing for remote files with SharePoint
2088
Built-in frameworks
  DITA Map
    1361
  DITA Topic
    1337
  DocBook 4
    1291
  DocBook 5
    1312
  DocBook 5.1
    1312
  DocBook 5.1 Assembly
    1334
  DocBook 5.1 Topic
    1335
  DocBook Targetset Map
    1336
  EPUB
    1426
  JATS
    1414
  jTEI
    1413
  TEI ODD
    1401
  TEI P5
    1388
  XHTML
    1374
  Built-in XML refactoring operations
    844, 2583
  Callout preferences

C
Configure Calabash with XEP
1534
Configure transformation scenario
1545
Configure Transformation Scenario dialog box
1548
Configuring content completion proposals
2192, 2200, 2207
Configuring options
308
Configuring Oxygen
124
Configuring Toolbars
320, 365
Confluence
3190
Content Completion Assistant in Grid Mode
586
Content completion configuration file
(cc_config.xml)
2192, 2200, 2207
Content completion helper views
536, 621
Content completion in Author Mode
616
Content completion in HTML
1259
Content completion in JSON
1126
Content completion in Text Mode
532
Content completion preferences
211
Content Fusion Connector add-on
2527
Contextual menu actions in Author mode
759
Contextual menu actions in JSON
1138
Contextual menu actions in Text Mode
567
Contextual menu of current editor tab
395
Convert database to XML Schema
1024, 2620
Convert JSON to XML
1131, 2632
Convert JSON to YAML
1137, 2631
Convert schema to another schema language
1021, 2618
Convert XML to JSON
1134, 2636
Convert XSD to JSON Schema
1172, 2564, 2639
Convert YAML to JSON
1137, 2632
Copy/Paste in Grid Mode
584
Create DTD from learned document structure
826
Create new transformation scenario
1464
Create new validation scenario
788
Create profiling attributes in Author Mode
670
Create profiling condition sets in Author Mode
675
Create Schematron Quick Fix
1236
Creating frameworks
2133
Creating Markdown documents
1265
Creating new documents
368
Creating publishing templates
1642, 1791
CSS :has relational pseudo class
2319
CSS @font-face rule
2311
CSS @media rule
CSS attr() function
CSS extensions
Additional CSS properties
- oxy-append-content CSS property
- oxy-collapse-text property value
- oxy-display-tags property
- oxy-editable property
- oxy-floating-toolbar CSS property
- oxy-foldable property
- oxy-folded property
- oxy-link property
- oxy-link-activation-trigger property
- oxy-lower-cyrillic property values
- oxy-morph property value
- oxy-not-foldable-child property
- oxy-placeholder-content property
- oxy-prepend-content CSS property
- oxy-show-placeholder property
- oxy-style property
- oxy-tags-background-color property
- oxy-tags-color property

display property
list-style-type property
visibility property
Additional CSS selectors
Built-in CSS selectors
Custom CSS pseudo-classes
Custom form controls
Custom functions
oxy_action
oxy_action_list
oxy_add
oxy_attributes
oxy_base-uri
oxy_capitalize
oxy_compound_action
oxy_concat
oxy_divide
oxy_getSomeText
oxy_indexof
oxy_label
oxy_lastindexof
<table>
<thead>
<tr>
<th>2046</th>
<th>2068</th>
</tr>
</thead>
<tbody>
<tr>
<td>Generic JDBC</td>
<td>Contextual menu actions</td>
</tr>
<tr>
<td>2061</td>
<td>2069</td>
</tr>
<tr>
<td>IBM DB2</td>
<td>Data Source Explorer view</td>
</tr>
<tr>
<td>2021</td>
<td>2015</td>
</tr>
<tr>
<td>Contextual menu actions</td>
<td>SQL support</td>
</tr>
<tr>
<td>2024</td>
<td>2071</td>
</tr>
<tr>
<td>JDBC-ODBC</td>
<td>Table Explorer view</td>
</tr>
<tr>
<td>2062</td>
<td>2017</td>
</tr>
<tr>
<td>MarkLogic</td>
<td>XQuery</td>
</tr>
<tr>
<td>2049</td>
<td>2074</td>
</tr>
<tr>
<td>Contextual menu actions</td>
<td>Debugging</td>
</tr>
<tr>
<td>2057</td>
<td>2078</td>
</tr>
<tr>
<td>Debugging</td>
<td>Berkeley DB XML</td>
</tr>
<tr>
<td>2054, 2078</td>
<td>2044, 2081</td>
</tr>
<tr>
<td>MarkLogic development</td>
<td>MarkLogic</td>
</tr>
<tr>
<td>2053</td>
<td>2054, 2078</td>
</tr>
<tr>
<td>Microsoft SQL Server</td>
<td>Drag and drop from Data Source Explorer</td>
</tr>
<tr>
<td>2025</td>
<td>2074</td>
</tr>
<tr>
<td>Contextual menu actions</td>
<td>Transformations</td>
</tr>
<tr>
<td>2028</td>
<td>2075</td>
</tr>
<tr>
<td>MySQL</td>
<td>Validation</td>
</tr>
<tr>
<td>2059</td>
<td>2075</td>
</tr>
<tr>
<td>Oracle</td>
<td>Debug PDF transformation</td>
</tr>
<tr>
<td>2029</td>
<td>1996</td>
</tr>
<tr>
<td>Contextual menu actions</td>
<td>Debugger preferences</td>
</tr>
<tr>
<td>2032</td>
<td>256</td>
</tr>
<tr>
<td>PostgreSQL</td>
<td>Debugging CSS stylesheets</td>
</tr>
<tr>
<td>2034</td>
<td>2409</td>
</tr>
<tr>
<td>Contextual menu actions</td>
<td>Debugging XQuery</td>
</tr>
<tr>
<td>2037</td>
<td>2102</td>
</tr>
<tr>
<td>SharePoint</td>
<td>Breakpoints</td>
</tr>
<tr>
<td>2081</td>
<td>2125</td>
</tr>
<tr>
<td>Browsing for remote files</td>
<td>Debugging Java extensions</td>
</tr>
<tr>
<td>2088</td>
<td>2131</td>
</tr>
<tr>
<td>Contextual menu actions</td>
<td>Identify expressions</td>
</tr>
<tr>
<td>2085</td>
<td>2122</td>
</tr>
<tr>
<td>MS Azure authentication</td>
<td>Information views</td>
</tr>
<tr>
<td>2089</td>
<td>2108</td>
</tr>
<tr>
<td>SharePoint Browser view</td>
<td>Breakpoints view</td>
</tr>
<tr>
<td>2083</td>
<td>2109</td>
</tr>
<tr>
<td>WebDAV</td>
<td>Context view</td>
</tr>
</tbody>
</table>
Steps in a typical debugging process
2121

Supported processors
2132

Debugging Java extensions
2131

Design editing mode
Attributes view
995

Components
961

Profiling
2126

Toolbar
2105

Debugging XSLT that call Java extensions
2131

Debugging XSLT
2102

Breakpoints
2109

Context view
2110

Messages view
2112

Nodes/Values Set view
2118

Output Mapping Stack view
2114

Stack view
2113

Templates view
2117

Trace view
2116

Variables view
2119

XPath Watch view
2111

Variables view
2119

XPath Watch view
2111

Layout
2103

Performance profiling
2126

Profiling
2126

Hotspots view
2129

Invocation Tree view
2128

Steps in a typical debugging process
2121

Breakpoints view
2109

Context view
2110

Messages view
2112

Nodes/Values Set view
2118

Output Mapping Stack view
2114

Stack view
2113

Templates view
2117

Trace view
2116

Variables view
2119

XPath Watch view
2111

Variables view
2119

XPath Watch view
2111

Layout
2103

Performance profiling
2126

Profiling
2126

Hotspots view
2129

Invocation Tree view
2128

Steps in a typical debugging process
2121

Supported processors
2132

Toolbar
2105

Debugging XSLT that call Java extensions
2131

Design editing mode
Attributes view
995

Components
961
Grouping components

xs:all
xs:alternative
xs:any
xs:anyAttribute
xs:assert
xs:attribute
xs:attributeGroup
xs:choice
xs:complexType
xs:element
xs:field
xs:group
xs:import
xs:include
xs:key
xs:keyRef
xs:notation
xs:openContent
xs:override
xs:redefine
xs:schema
xs:selector
xs:sequence
xs:simpleType
xs:unique

Contextual menu actions

Editing actions
Facets view
Navigation
Outline view
Palette view
Design mode preferences
Detect Main Files
Detect Main Files from Project
Diagram preferences
Diff Directories tool
Diff Files tool
Diff tool
Digital Signature
Canonicalize files
Certificates
Sign files
Change order of topics
Chunking
Create bookmark
Create map
Create subject scheme
Create submap
Create table of contents
Define keys
DITA Maps Manager
Edit Properties dialog box
Find unreferenced resources
Insert Reference dialog box
Insert references
Insert topic groups
Insert topic headings
Managing maps
Move resources
Remove topics
Rename resources
Root map
Validate and Check for Completeness
XML catalogs
Metadata
Migrate Excel to DITA
Migrate Office documents to DITA
Migrate various formats to DITA
Migrate Word to DITA
Open Toolkit
Output
Customizing Transformations
Custom build file
DITA-OT Transformation Scenario
Syntax highlights in codeblocks
Profiling
Applying profiling attributes
Applying profiling condition sets
Attribute groups
Creating or editing profiling attributes
Creating or editing profiling condition sets
Customizing colors and styles for profiling attributes
DITAVAL filter file
Filtering attribute values
Flagging content with DITAVAL file
Publishing profiling
Showing and filtering profiling attributes
Using subject scheme map
Projects
Publishing
Relationship tables
reltable
Resource Hierarchy/Dependencies view
Reusing content
Branch Filtering
Code References
Conkeyref
Conref
Content Reference Push mechanism
Content References
DITA Reusable Components view
Edit Content References
Key Scopes
Reference Topics in multiple maps
Reusable components
Create Reusable Component
Insert Reusable Component
Reuse Content dialog box
Variable text
Specialization
Add images
Add media resources
Content completion
Convert topic types
Create new topic
Edit topics
Embed HTML Content
Fast Create multiple topics
Image maps
LaTeX equations
MathML equations
Tables
696, 2986
Translations
3179
Use external DITA-OT
3170
Your first DITA document
39, 2887
DITA and Markdown documents
1275, 3024
DITA Authoring and Publishing
2886
DITA Logging preferences
275
DITA Map document type
1361
Author mode actions
1362, 2947
Open DITA map with content resolved
2958
DITA Map menu actions
1362, 2947
DITA Map PDF - based on HTML5 & CSS
1449, 3095
Abbreviated-form element
1974
Accessibility
1928, 1928
Add
Strings
1981
Appendices
1919
Page breaks
1920
Back matter
1876
Style topics
1877
Bookmap styling
1977
Bookmarks
1908
Change labels
1908
Depth
1909
Initial state
1909
Remove numbering
1910
Sections display
1909
Changing the cover page
1976
Changing the page size
1976
Changing the TOC depth
1976
Code Blocks
1971
Command Line
1781
Comments
1934
Styling
1940
Cover page
1852
Add empty pages
1860
Add second cover
1858
Add text
1857
Background image
1854
Copyright page
1861
Place cover on left side
1858
Place cover on right side
1858
Title styling
Strings
1980

Notes
1974

Numbering
1879, 1883
Remove prefix
1889
Reset page numbering
1888

Page breaks
1849
Add a blank page after a topic
1851
Avoid in lists and tables
1849
Enforce number of lines
1852
Force before or after topic
1850

Page size
Change setting for an element
1827
Changing
1827
Orientation
1827

Permissions
1982
Security
1982
Styling
1946

Table of contents
1890
Change header
1895
Display short description in TOC
1896
Display topic before TOC
1896
Increase depth

1892
Remove TOC entries
1897
Start on odd page
1896
Style
1894
Tables
1962
Centering
1970
Layout
1964
Rotate
1963
Small images
1965
Split Cell
Borders
1971
Stripes
1970
Text bleeding
1965
Wide
1963
Zebra stripes
1970
Tracked changes
1934
Hide footnotes
1941
Styling
1940
Trademarks
1975
Translation
1979
Troubleshooting
1985
Abbreviated-form
Cell borders missing
1986 Date formatting issues
1983 Disappearing lines
1986 Error parsing
1985 Failed to run pipeline error
1983 Glossentry
1985 Glossgroup
1985 Highlights span off page
1942 I/O exception
1985 PDF file is damaged
1982 Unknown host
1985 DITA Map to CHM (Compiled Help) transformation
1453, 3099 DITA Map to Kindle transformation
1455, 3101 DITA Map to MS Office Word transformation
1452, 3098 DITA Map to PDF transformation
1451, 3098 DITA Map to WebHelp Responsive transformation
1435, 3082 DITA map to XHTML output customization
3122 DITA Map to Zendesk transformation
3101 DITA Map toolbar actions
1362, 2947 DITA Map transformations
1434, 3081 DITA Maps
Transformation scenarios
1434, 3081
DITA maps preferences
270 DITA new topics preferences
273 DITA Outgoing References View add-on
2540 DITA PDF - based on HTML5 & CSS
Add
Strings
1981
Changing the cover page
1976
Changing the page size
1976
Changing the TOC depth
1976
Command Line
1781
Comments
Styling
1940
Custom transformation parameters
1975
Customization CSS
1814
Customize
Strings
1979
Debugging
1816
Figures
Figure numbering
1958
i18n
1979
Integration Server
Jenkins
1782
Localization
1979
Modify
Strings
<table>
<thead>
<tr>
<th>1980 Permissions</th>
<th>1502, 3116 Feedback tab</th>
</tr>
</thead>
<tbody>
<tr>
<td>1982 Security</td>
<td>1500, 3113 Filters tab</td>
</tr>
<tr>
<td>1982 Tracked changes</td>
<td>1501, 3114 FO Processor tab</td>
</tr>
<tr>
<td>Styling</td>
<td>1497, 3111 Output tab</td>
</tr>
<tr>
<td>1940 Translation</td>
<td>1505, 3118 Parameters tab</td>
</tr>
<tr>
<td>1979 DITA perspective</td>
<td>1499, 3112 Skins tab</td>
</tr>
<tr>
<td></td>
<td>1491 Templates tab</td>
</tr>
<tr>
<td></td>
<td>1493, 3106 DITA-OT Transformation Scenario</td>
</tr>
<tr>
<td>346 DITA preferences</td>
<td>1291 DocBook 4 document type</td>
</tr>
<tr>
<td>269 DITA prolog updater add-on</td>
<td>1293 Author mode actions</td>
</tr>
<tr>
<td>2538 DITA publishing preferences</td>
<td>1293 DocBook 4 menu actions</td>
</tr>
<tr>
<td>275 DITA tables in Author Mode</td>
<td>1293 DocBook 4 toolbar actions</td>
</tr>
<tr>
<td>696, 2986 DITA to Confluence</td>
<td>1312 DocBook 5 document type</td>
</tr>
<tr>
<td>3190 DITA to Word</td>
<td>1314 DocBook 5 menu actions</td>
</tr>
<tr>
<td>1452, 3098 DITA topic</td>
<td>1314 DocBook 5 toolbar actions</td>
</tr>
<tr>
<td>Transformation scenarios</td>
<td>1334 DocBook 5.1 Assembly</td>
</tr>
<tr>
<td>1457, 3103 DITA Topic document type</td>
<td>1334 DocBook 5.1 document type</td>
</tr>
<tr>
<td>1337 Author mode actions</td>
<td>1335 DocBook 5.1 Topic document type</td>
</tr>
<tr>
<td>1338, 3001 DITA Topic menu actions</td>
<td>1490 DITA-OT preferences</td>
</tr>
<tr>
<td>1338, 3001 DITA Topic toolbar actions</td>
<td>269 DITA-OT transformation scenario</td>
</tr>
<tr>
<td>1338, 3001 DITA topic transformations</td>
<td>1490 DITA-OT Transformation Scenario</td>
</tr>
<tr>
<td>1457, 3103 DITA-OT preferences</td>
<td>1490 DITA-OT Transformation Scenario</td>
</tr>
<tr>
<td>269 DITA-OT transformation scenario</td>
<td>1490 DITA-OT Transformation Scenario</td>
</tr>
<tr>
<td>1490 Advanced tab</td>
<td>2578 DocBook checker add-on</td>
</tr>
<tr>
<td></td>
<td>2578 DocBook olink</td>
</tr>
</tbody>
</table>
DocBook tables in Author Mode 1309, 1331

DocBook Targetset Map document type 686

DocBook to DITA 1336

DocBook to DITA transformation 3190

DocBook to DITA transformation 1461

DocBook to EPUB transformation 1462

DocBook to PDF output customization 1997

DocBook to PDF transformation 1462

DocBook to WebHelp transformation 1458

DocBook transformations 1458

Document plugin extension 2438

Document template preferences 167

Document templates
  Creating 375
  Customizing 377
  Sharing 381

Document type association preferences 138

Document type configuration dialog box 140
  Association rules tab 142
  Author tab 145
    Actions subtab 147
    Content Completion subtab 160

Contextual Menu subtab 157

CSS subtab 146

Menu subtab 157

Toolbar subtab 160

Catalogs tab 164

Classpath tab 144

Extensions tab 167

Schema tab 144

Templates tab 144

Transformation tab 165

Validation tab 166

Document Types 1291, 1430

Documents with long lines 471

Drag and Drop in Author Mode 612

Drag and Drop in Grid Mode 583

Drag and Drop in Text Mode 529

DTD Entities 835, 835

Duplicate transformation scenario 1547

Edit mode preferences 171

Edit profiling attributes in Author Mode 670

Edit profiling condition sets in Author Mode
Edit validation scenario
Editing actions in Grid Mode
Editing attributes in Author Mode
Editing content in Author Mode
Editing Modes
Author
Design
Grid
Text
Editing publishing templates
Editing XML markup in Author Mode
Editing XML markup in Text Mode
Editing XSLT
  Compile XSL Stylesheet for Saxon tool
  Encoding preferences
Editor perspective
Editor preferences
Editor variables
Custom editor variables
Elements view in Author Mode
Elements view in Text mode
Elements view preferences
EMF images in Author mode
Emmet add-on
Encoding preferences
Entities view
EPS image rendering in Author mode
EPUB document type
Error: 1248 WARN
org.apache.fop.apps.FOUserAgent
Error: Anttask - Error rendering fo file
Error: Cannot find files.log file
Error: Could not create MSXML object
Error: java.io.FileNotFoundException
Error: Navigation to the web page was canceled
Error: OutOfMemory
Error: OutofMemoryError
Error: stack overflow
Error: Startup crash - Fault Module Name
Executing SQF in Other Documents
eXist database
  Connection wizard
  Manual configuration
eXist database connection
<table>
<thead>
<tr>
<th>Page</th>
<th>Section</th>
</tr>
</thead>
<tbody>
<tr>
<td>2044</td>
<td>eXist database contextual menu actions</td>
</tr>
<tr>
<td>2047</td>
<td>Export actions</td>
</tr>
<tr>
<td>147</td>
<td>Export color themes</td>
</tr>
<tr>
<td>129</td>
<td>Export Global Options</td>
</tr>
<tr>
<td>314</td>
<td>Export Global Transformation Scenarios</td>
</tr>
<tr>
<td>323</td>
<td>Export Global Validation Scenarios</td>
</tr>
<tr>
<td>314</td>
<td>Export Layout</td>
</tr>
<tr>
<td>360</td>
<td>Export to Excel</td>
</tr>
<tr>
<td>587</td>
<td>Exporting Markdown documents</td>
</tr>
<tr>
<td>1263</td>
<td>Extending frameworks</td>
</tr>
<tr>
<td>2133</td>
<td>Extending Oxygen with plugins</td>
</tr>
<tr>
<td>2411</td>
<td>External DITA-OT plugin extension</td>
</tr>
<tr>
<td>2426</td>
<td>External tool configuration</td>
</tr>
<tr>
<td>292</td>
<td>External tool preferences</td>
</tr>
<tr>
<td>2852</td>
<td>External Tools</td>
</tr>
<tr>
<td></td>
<td>F</td>
</tr>
<tr>
<td>951</td>
<td>Facets view in Design mode</td>
</tr>
<tr>
<td>2527</td>
<td>Feedback Comments Manager view</td>
</tr>
<tr>
<td>288</td>
<td>File comparison appearance preferences</td>
</tr>
<tr>
<td>286</td>
<td>File comparison preferences</td>
</tr>
<tr>
<td>396</td>
<td>File properties</td>
</tr>
<tr>
<td>565</td>
<td>File related actions in Text Mode</td>
</tr>
<tr>
<td>297</td>
<td>File type preferences</td>
</tr>
<tr>
<td>679</td>
<td>Filtering profiled content in Author mode</td>
</tr>
<tr>
<td>444</td>
<td>Find actions</td>
</tr>
<tr>
<td>442</td>
<td>Find All Elements action</td>
</tr>
<tr>
<td>442</td>
<td>Find All Elements dialog box</td>
</tr>
<tr>
<td>444</td>
<td>Find and invoke action</td>
</tr>
<tr>
<td>432</td>
<td>Find/Replace action</td>
</tr>
<tr>
<td>432</td>
<td>Find/Replace dialog box</td>
</tr>
<tr>
<td>436</td>
<td>Find/Replace in Files action</td>
</tr>
<tr>
<td>436</td>
<td>Find/Replace in multiple files</td>
</tr>
<tr>
<td>446</td>
<td>Finding/Replacing text</td>
</tr>
<tr>
<td></td>
<td>Find All Elements dialog box</td>
</tr>
<tr>
<td></td>
<td>Find/Replace dialog box</td>
</tr>
<tr>
<td></td>
<td>Find/Replace in multiple files</td>
</tr>
<tr>
<td></td>
<td>Navigating Find/Replace matches</td>
</tr>
<tr>
<td></td>
<td>Quick Find toolbar</td>
</tr>
<tr>
<td>1025</td>
<td>Flatten Schema tool</td>
</tr>
<tr>
<td>106</td>
<td>Floating license server</td>
</tr>
<tr>
<td>261</td>
<td>FO processor preferences</td>
</tr>
<tr>
<td>611</td>
<td>Folding elements in Author Mode</td>
</tr>
<tr>
<td></td>
<td>Folding in Text Mode</td>
</tr>
</tbody>
</table>
Font preferences

Font size configuration

Font size configuration in Text mode

FOP Error

Form Control

Editing processing instructions

Form Controls

Audio file player

Browser

Button

Button group

Checkbox

Combo box

Date picker

HTML content

Pop-up

Render HTML frames

Render SVG

Text area

Text field

URL chooser

Video player

Form Controls in Author mode

Format and Indent Files tool

Format and Indent in Text Mode

Format/Indent multiple files

Formatting preferences

Formatting Schematron Quick Fix Content

Framework customization

Configuring an extensions bundle

Configuring content completion proposals

Configuring document templates

Configuring transformation scenarios

Configuring validation scenarios

Configuring XML catalogs

Creating

Creating a basic document type association

Customizing annotations for the Content Completion Assistant

Customizing Smart Paste mapping

Customizing the Content Completion Assistant

Deploying as Add-ons

Extending

2133

2190, 2191

2289

2212

2188

2192, 2200, 2207

2227

2216

2218

2228

2133

2297

2212

2188

2289

2133
### Extensions

<table>
<thead>
<tr>
<th>Extension</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Author Action Event Handler</td>
<td>2231</td>
</tr>
<tr>
<td>Author Edit Properties Handler</td>
<td>2286</td>
</tr>
<tr>
<td>Author Extension State Listener</td>
<td>2245</td>
</tr>
<tr>
<td>Author Image Decorator</td>
<td>2236</td>
</tr>
<tr>
<td>Author Persistent Highlighter</td>
<td>2237</td>
</tr>
<tr>
<td>Author Reference Resolver</td>
<td>2241</td>
</tr>
<tr>
<td>Author Schema Aware Editing Handler</td>
<td>2283</td>
</tr>
<tr>
<td>Author Table Cell Separator Provider</td>
<td>2258</td>
</tr>
<tr>
<td>Author Table Cell Span Provider</td>
<td>2254</td>
</tr>
<tr>
<td>Author Table Column Width Provider</td>
<td>2249</td>
</tr>
<tr>
<td>Content Completion Handler</td>
<td>2238</td>
</tr>
<tr>
<td>CSS Styles Filter</td>
<td>2262</td>
</tr>
<tr>
<td>Custom Attribute Value Editor</td>
<td>2260</td>
</tr>
<tr>
<td>Custom Drag and Drop Listener</td>
<td>2241</td>
</tr>
<tr>
<td>Custom Object Insertion Handler</td>
<td>2285</td>
</tr>
<tr>
<td>Element Locator Provider</td>
<td>2263</td>
</tr>
<tr>
<td>Link Target Element Finder</td>
<td>2263</td>
</tr>
<tr>
<td>Profiling Conditional Text Provider</td>
<td>2263</td>
</tr>
<tr>
<td>Unique Attributes Recognizer</td>
<td>2238</td>
</tr>
<tr>
<td>XML Node Renderer Customizer</td>
<td>2269</td>
</tr>
</tbody>
</table>

### Sharing

<table>
<thead>
<tr>
<th>Sharing</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Text to markup shortcut patterns</td>
<td>2187</td>
</tr>
<tr>
<td>Translating annotations for the Content Completion Assistant</td>
<td>2212</td>
</tr>
<tr>
<td>Framework directory locations</td>
<td>139</td>
</tr>
<tr>
<td>Frameworks</td>
<td>1291, 1430</td>
</tr>
<tr>
<td>FTP connection settings</td>
<td>304</td>
</tr>
</tbody>
</table>

### G

| General plugin extension                                               | 2435 |
| Generate Documentation tools                                           | 2627, 2646 |
| Generate HTML report for directory comparison tool                     | 509, 2715 |
| Generate IDs in Author mode                                            | 755  |
| Generate Java classes from XSD                                         | 1027, 2568, 2623 |
| Generate Java classes from XSD tool                                   | 1027, 2568, 2623 |
| Generate JSON Schema documentation                                    | 1167, 2569, 2659 |
| Generate JSON Schema tool                                             | 1165, 2629 |
| Generate OpenAPI documentation                                         | 2575, 2662 |
| Generate Sample JSON Files tool                                        | 1170, 2627 |
| Generate Sample XML Files tool                                         | 1006, 2612 |
| Generate WSDL Documentation tool                                       | 1067, 2656 |
| Generate XML Schema Documentation tool                                 | 1011, 2647 |
| Customize PDF output for DITA                                          |      |
HTML documents

Content completion
1259

Editing features
1255

Folding
1260

Minification
1260

Minifying
1260

Outline view
1261

Syntax highlighting
1259

Validation
1258

HTML grammar checker
2546

HTML to DITA
3190

HTTP authentication schemes
392

HTTP connection settings
303

HTTP floating license server

Allowed users list
117

Replacing
115

Upgrading
116

HTTP license server
107

Management and Statistics
112

HTTPS troubleshooting
390

Huge file editor
471

Huge files
471

IBM DB2 database connection
2021

IBM DB2 database contextual menu actions
2024

ID generation
755

Ignore validation problems
812

Ignored validation problems preferences
229

Image Map Editor in Author Mode
725

Image Map Editor in DITA
725, 2980

Image Map Editor in DocBook
731

Image Map Editor in TEI
736

Image Map Editor in XHTML
742

Image maps
1959

Image not found
1959

Image preview pane
469

Image rendering in Author mode
719

Import color themes
129

Import data dynamically
2099

Import Global Options
314

Import Global Transformation Scenarios
323

Import Global Validation Scenarios
323

Import preferences
266
Import text files
2091
Importing data
2091
Importing data from a database
2096
Importing data from Excel
2093
Importing data from HTML files
2098
Importing Data from text files
2091
Increase memory
343, 1772
Indenting Schematron Quick Fix Content
1243
Information view
512
Information view preferences
306
Insert File in Text mode
565
Insert images in Author Mode
719
Install new add-ons
120
Installing Oxygen
85
Floating license servers
Server signature does not match
117, 2878
Group deployment
101
Licensing
101
Linux Installation
92
Linux/Unix Server Installation
98
macOS Installation
90
Transfer license key
106
Upgrading
118
Windows Installation
86
Windows Server Installation
97
Integrate Feedback with Oxygen
2527
Integrate Schematron Quick Fixes in a framework
1245
Integrate Schematron rules in a framework
1207
Integrating external tools
2852
Introduction
28
J
JAI images in Author mode
722
JATS document type
1414
Author mode actions
1415
JATS menu actions
1415
JATS tables in Author Mode
714
JATS toolbar actions
1415
Java system properties
333
Java VM parameters
339
JavaScript documents
1184
Content Completion
1187
Editing features
1185
Outline view
1189
Syntax highlighting 1188
Validation 1187
JCGM library 721, 1430, 2579
JDBC-ODBC database connection 2062
JSON documents 1110
    Associate schema 1122
    Associate schema directly in JSON documents 1125
    Associate schema through a validation scenario 1122
    Content completion 1126
    Contextual menu actions 1138
    Folding 1128
    JSON to XML converter 1131, 2632
    JSON to YAML converter 1137, 2631
    Navigating references 1113
    Outline view 1128
    Presenting validation errors 1116
    Schema annotations 1127
    Syntax highlighting 1128, 1177
    Validating JSON schema 1176
    Validation 1113
        Against a schema 1114
        Automatic validation 1115
        Check Well-formedness 1114
        Manual validation 1115
        Resolving references to remote schemas 1122
        Sharing scenarios 1121
        Validation scenarios 1118
        XML to JSON converter 1134, 2636
        XSD to JSON Schema converter 1172, 2564, 2639
        YAML to JSON converter 1137, 2632
        JSON schema 1114, 1176
        JSON Schema 1146
        JSON Schema Design mode 355, 1149
        Components 1155, 1156
        Contextual menu actions 1152
        Editing actions 1152
        JSON Schema Design Mode 1149
        Validation 1164
        JSON Schema diagram editor 1149
        Components 1155, 1156
        Contextual menu actions 1152
        Editing actions 1152
        Validation 1164
1164
JSON Schema Diagram Editor
1150
Navigation
1151
Palette view
1151
JSON Schemas
Design mode editing
Navigation
1150
Palette view
1151
JSON to XML tool
1131, 2632
JSON to YAML tool
1137, 2631
JSON validation scenarios
1118
jTEI
document type
1413
K
Kerberos
392
L
Large documents
469
Large File Viewer tool
2674
Large files
470
Layout configuration
314, 360
Layout preferences
135
LESS stylesheets
Compile to CSS
1082
Content completion
1081
Editing features
1080
Syntax highlighting
1082
Validation
1081
Licensing Oxygen
101
Floating license
Reserve floating license
106
Floating licenses
103
Named-user license
102
Release a floating license
105
Request a floating license
HTTP license server
104
Load Layout
314, 360
Localizing frameworks
2229
Localizing SQF messages
1245
Localizing the user interface
338
Localizing XML refactoring operations
876, 2611
Lock Handler plugin extension
2427
Lock/Unlock XML tags in Text Mode
564
Locking a floating license
106
M
Mac accented characters
2877
Mac Function Keys
2877
Mac restart
2877
Mac text input freeze
Overview 419
Transformation 422
Validation 422
Manage add-ons 120
Manage highlighted content in Text Mode 562
Manual validation actions 775
Manual validation in JSON 1115
Markdown documents 1262
Actions in Markdown Editor 1266
Creating Markdown documents 1265
DITA 1275, 3024
Markdown Editor 1263
Rules and specifications 1277
Syntax highlighting 1274
Validation 1274
Markdown Editor 1263
Markdown grammar checker 2546
Markdown preferences 275
Markdown to DITA 3190
MarkLogic database connection 2049
MarkLogic database contextual menu actions 2057
MarkLogic debugging 2054, 2078
MarkLogic for the developer 2053
Markup transparency in Text Mode 563
MathML equations in HTML output in Author mode 1709
MathML equations in WebHelp output 1676, 1754
MathML notations in Author mode 749
MathML preferences 191
Media query 2310
Memory availability 343
Memory issues 1772
Menu Shortcut Keys 294
Message preferences 307
Microsoft SQL Server database connection 2025
Microsoft SQL Server database contextual menu actions 2028
Model view
545, 631
Modular XML files
829
Move file tabs
393
Move XML resources
835
MS Azure authentication
2089
MSXML 3.0 and 4.0 preferences
250
MSXML.NET preferences
251
MySQL database connection
2059

Navigating in Text Mode
517
Navigating in Author Mode
591
Network connection preferences
301
New DITA project
2894
New document from templates
375
New Document Wizard
368
New from Templates Wizard
375
New project
47, 399
NISO Journal Article Tag Suite document type
1414
Non-XML files
397
NTLM
392
NVDL schemas
1102
  Component Dependencies view
  1107

Content completion
1105
Diagram Editor
1102
  Contextual menu actions
  1104
  Full Model view
  1103
  Introduction
  1102
  Logical Model view
  1104
Outline view
1107
Refactoring actions
1108
Search actions
1108
Syntax highlighting
1106
Validation
1105

O
Olink element
1309, 1331
Open DITA map with content resolved
2958
Open File at Cursor in Text mode
565
Open file at specific location
383
Open file at start-up
382
Open file in system application
382
Open preferences
199
Open Redirect plugin extension
2428
Open remote document
386
Open URL dialog box
<table>
<thead>
<tr>
<th>Menu Shortcut Keys</th>
<th>Network Connection Settings</th>
</tr>
</thead>
<tbody>
<tr>
<td>Menu Shortcut Keys</td>
<td>Network Connection Settings</td>
</tr>
<tr>
<td>Messages</td>
<td>Open</td>
</tr>
<tr>
<td>MSXML XSLT</td>
<td>MSXML.NET XSLT</td>
</tr>
<tr>
<td></td>
<td>Network Connection Settings</td>
</tr>
<tr>
<td></td>
<td>Project Level Settings</td>
</tr>
<tr>
<td></td>
<td>Proxy</td>
</tr>
<tr>
<td></td>
<td>Relax NG parser</td>
</tr>
<tr>
<td></td>
<td>Review</td>
</tr>
<tr>
<td></td>
<td>Sample XML Files Generator</td>
</tr>
<tr>
<td></td>
<td>Save</td>
</tr>
<tr>
<td></td>
<td>Saxon-HE/PE/EE XQuery</td>
</tr>
<tr>
<td></td>
<td>Saxon-HE/PE/EE XQuery Advanced</td>
</tr>
<tr>
<td></td>
<td>Saxon-HE/PE/EE XSLT</td>
</tr>
<tr>
<td></td>
<td>Saxon-HE/PE/EE XSLT Advanced</td>
</tr>
<tr>
<td></td>
<td>Saxon6 XSLT</td>
</tr>
<tr>
<td></td>
<td>Schema Design mode</td>
</tr>
<tr>
<td></td>
<td>Schema-Aware</td>
</tr>
<tr>
<td></td>
<td>Schematron parser</td>
</tr>
<tr>
<td></td>
<td>Spell Check</td>
</tr>
<tr>
<td></td>
<td>Spell Check Dictionaries</td>
</tr>
<tr>
<td></td>
<td>SVN</td>
</tr>
<tr>
<td></td>
<td>SVN Diff</td>
</tr>
<tr>
<td></td>
<td>SVN Messages</td>
</tr>
<tr>
<td></td>
<td>SVN Working Copy</td>
</tr>
<tr>
<td></td>
<td>Syntax Highlight</td>
</tr>
<tr>
<td></td>
<td>Text mode</td>
</tr>
<tr>
<td></td>
<td>Trusted Hosts</td>
</tr>
<tr>
<td></td>
<td>Views</td>
</tr>
<tr>
<td></td>
<td>Whitespaces</td>
</tr>
</tbody>
</table>
Overview
419
Transformation
422
Validation
422
Managing resources
402
Move resources
412
Project view
402
Rename resources
412
Sharing
415
Protect PDF
1982
Proxy auto-config script
302
Proxy preferences
301
PSD image rendering in Author mode
721
Publishing
1432
Q
Quick Assist feature in Text Mode
566
Quick Find toolbar
445
Quick fix support in XML documents
813
R
RDP performance issues
2857
Read-only files
472
Rectangular Selection feature
529
Refactoring Operations plugin extension
2432
Refactoring XML Documents
840, 2580
Refreshing content in Author mode
754
Regular expressions syntax
446
Relax NG schemas
1083
Component Dependencies View
1097
Content completion
1089
Custom datatype library
1102
Diagram Editor
1084
Contextual menu actions
1087
Full Model view
1084
Introduction
1084
Logical Model view
1085
Symbols
1086
Editing in Main Files context
1083
Moving resources
1097
Outline view
1091
Quick Assist feature
1100
Refactoring actions
1098
Renaming resources
1097
Resource Hierarchy/Dependencies view
1094
Search actions
1098
<table>
<thead>
<tr>
<th>Syntax highlighting</th>
<th>1090</th>
</tr>
</thead>
<tbody>
<tr>
<td>Validation</td>
<td>1088</td>
</tr>
<tr>
<td>Release a floating license</td>
<td>105</td>
</tr>
<tr>
<td>Remote Desktop performance issues</td>
<td>2857</td>
</tr>
<tr>
<td>Rename XML resources</td>
<td>835</td>
</tr>
<tr>
<td>Rendering documents in Author Mode</td>
<td>590</td>
</tr>
<tr>
<td>Request a floating license from an HTTP license server</td>
<td>104</td>
</tr>
<tr>
<td>Reserving a floating license</td>
<td>106</td>
</tr>
<tr>
<td>Reset Global Options</td>
<td>314</td>
</tr>
<tr>
<td>Reset Layout</td>
<td>314, 360</td>
</tr>
<tr>
<td>Resolve schemas through XML catalogs</td>
<td>828</td>
</tr>
<tr>
<td>Resource Hierarchy/Dependencies view for DITA documents</td>
<td>3183</td>
</tr>
<tr>
<td>Resource Hierarchy/Dependencies view for XML documents</td>
<td>832</td>
</tr>
<tr>
<td>Resource Locking plugin extension</td>
<td>2430</td>
</tr>
<tr>
<td>Restrict access to PDF content</td>
<td>1982</td>
</tr>
<tr>
<td>Restricting Schematron Quick Fix operations</td>
<td>1242</td>
</tr>
<tr>
<td>Results view</td>
<td>548, 635</td>
</tr>
<tr>
<td>Make persistent copy of results</td>
<td>553, 639</td>
</tr>
<tr>
<td>Retina/HiDPI Images in Author mode</td>
<td>724</td>
</tr>
<tr>
<td>Review preferences</td>
<td>183</td>
</tr>
<tr>
<td>Review tools in Author Mode</td>
<td>642</td>
</tr>
<tr>
<td>Review view</td>
<td>664</td>
</tr>
<tr>
<td>Run DITA-OT Integrator transformation</td>
<td>1455, 3102</td>
</tr>
</tbody>
</table>

**S**

- S1000D document type | 1430 |
- Sample XML File Generator preferences | 242 |
- Save file | 384 |
- Save preferences | 201 |
- Save remote document | 386 |
- Save template as button | 1495, 3109 |
- Saxon 6 XSLT preferences | 246 |
- Saxon HE/PE/EE advanced XQuery preferences | 256 |
- Saxon HE/PE/EE advanced XSLT preferences | 249 |
- Saxon HE/PE/EE XQuery preferences | 254 |
- Saxon HE/PE/EE XSLT preferences | 247 |
- Saxon XQuery Transformer plugin extension | 2435 |
- Saxon XSLT and XQuery transformer add-on | 2563 |
- Saxon XSLT Transformer plugin extension | 2434 |
- Schema annotations in Author Mode | 619 |
- Schema annotations in JSON | 1127 |
- Schema annotations in Text Mode | 535 |
Schema Design mode
355, 948

Schema Diagram Editor
355, 948

Schematron
  Editing features
  1192
  Editing in Main Files context
  1206
  Examples
  1194, 1223
  Integrating in a framework
  1207
  Sharing
  1207
  Unit test (XSpec)
  1221
  Validation
  Against Schematron
  1206

Schematron examples
1194, 1223

Schematron Quick Fix examples
1194, 1223

Schematron Quick Fix Operations
  Add
  1237
  Delete
  1237
  Replace
  1237
  String Replace
  1237

Schematron Quick Fixes
815, 1223
  Content Completion
  1248
  Customizing
  1236
  Defining
  1236
  Embedded

Examples
1194, 1223

Executing SQF in Other Documents
1244

Formatting and Indenting
1243

Highlight occurrences
1248

Integrating in a framework
1245

Localization
1245

Multiple similar quick fixes
1244

Refactoring actions
1248

Restricting operations
1242

Search actions
1248

Sharing
1245

SQF Operations
1237

Use-When Condition
1242

User Entry operation
1242

Validation
1247

Schematron schemas
  Content completion
  1209
  Syntax highlighting
  1210

Schematron Schemas
  Embedded
  810, 1211
  Highlight occurrences
  1217
  Moving resources

aq
<table>
<thead>
<tr>
<th>1207</th>
<th>Sharing settings</th>
</tr>
</thead>
<tbody>
<tr>
<td>312</td>
<td>Sharing transformation scenarios</td>
</tr>
<tr>
<td>415</td>
<td>Sharing validation scenarios</td>
</tr>
<tr>
<td>415</td>
<td>Sharing XML refactoring operations</td>
</tr>
<tr>
<td>875, 2610</td>
<td>Shortcut actions in Text Mode</td>
</tr>
<tr>
<td>521</td>
<td>Shortcut key configuration</td>
</tr>
<tr>
<td>294</td>
<td>Shortcut Keys</td>
</tr>
<tr>
<td>55</td>
<td>Show change tracking/comments in DocBook PDF</td>
</tr>
<tr>
<td>1463</td>
<td>Sign files tool</td>
</tr>
<tr>
<td>882, 2667</td>
<td>Signing XML document preferences</td>
</tr>
<tr>
<td>268</td>
<td>Simple text editor</td>
</tr>
<tr>
<td>397</td>
<td>Single sign-on</td>
</tr>
<tr>
<td>392</td>
<td>Smart Autocomplete</td>
</tr>
<tr>
<td>2550</td>
<td>Smart Editing in Text Mode</td>
</tr>
<tr>
<td>520</td>
<td>Smart paste preferences</td>
</tr>
<tr>
<td>180</td>
<td>Smart Paste support in Author Mode</td>
</tr>
<tr>
<td>613</td>
<td>Sorting a table in Author Mode</td>
</tr>
<tr>
<td>714</td>
<td>Sorting exotic languages</td>
</tr>
<tr>
<td>2564</td>
<td>Sorting list items in Author Mode</td>
</tr>
<tr>
<td>717</td>
<td>Sorting selected table rows in Author Mode</td>
</tr>
<tr>
<td>715</td>
<td>Sorting tables with merged cells in Author Mode</td>
</tr>
<tr>
<td>717</td>
<td>Special character preferences</td>
</tr>
<tr>
<td>199</td>
<td>Special characters</td>
</tr>
<tr>
<td>462</td>
<td>Character Map</td>
</tr>
<tr>
<td>464</td>
<td>Fallback Font Support</td>
</tr>
<tr>
<td>465</td>
<td>Inserting symbols</td>
</tr>
<tr>
<td>464</td>
<td>Unrecognized characters</td>
</tr>
<tr>
<td>464</td>
<td>Unsupported characters</td>
</tr>
<tr>
<td>752</td>
<td>Special characters in Author mode</td>
</tr>
<tr>
<td>752</td>
<td>Special characters in Grid mode</td>
</tr>
<tr>
<td>586</td>
<td>Special characters in Text mode</td>
</tr>
<tr>
<td>564</td>
<td>Spell check preferences</td>
</tr>
<tr>
<td>229</td>
<td>Spell checking</td>
</tr>
<tr>
<td>447</td>
<td>Add dictionaries</td>
</tr>
<tr>
<td>451</td>
<td>Add term lists</td>
</tr>
<tr>
<td>453</td>
<td>Custom list of terms</td>
</tr>
<tr>
<td>453</td>
<td>Customizing dictionaries and term lists</td>
</tr>
<tr>
<td>450</td>
<td>Dictionaries and term lists</td>
</tr>
<tr>
<td>449</td>
<td>Forbidden words</td>
</tr>
<tr>
<td>453</td>
<td>Ignored words</td>
</tr>
<tr>
<td>456</td>
<td>Learned words</td>
</tr>
<tr>
<td>456</td>
<td>Multiple files</td>
</tr>
<tr>
<td>752</td>
<td>Special characters in Author mode</td>
</tr>
<tr>
<td>752</td>
<td>Special characters in Grid mode</td>
</tr>
<tr>
<td>586</td>
<td>Special characters in Text mode</td>
</tr>
<tr>
<td>564</td>
<td>Spell check preferences</td>
</tr>
<tr>
<td>229</td>
<td>Spell checking</td>
</tr>
<tr>
<td>447</td>
<td>Add dictionaries</td>
</tr>
<tr>
<td>451</td>
<td>Add term lists</td>
</tr>
<tr>
<td>453</td>
<td>Custom list of terms</td>
</tr>
<tr>
<td>453</td>
<td>Customizing dictionaries and term lists</td>
</tr>
<tr>
<td>450</td>
<td>Dictionaries and term lists</td>
</tr>
<tr>
<td>449</td>
<td>Forbidden words</td>
</tr>
<tr>
<td>453</td>
<td>Ignored words</td>
</tr>
<tr>
<td>456</td>
<td>Learned words</td>
</tr>
<tr>
<td>456</td>
<td>Multiple files</td>
</tr>
</tbody>
</table>
Replace dictionaries

Spell Checking
  Automatic spell check

SQF
  815, 1223
  Content Completion
  1248
Customizing
  1236
Defining
  1236
Embedded
  1250
Examples
  1194, 1223
Executing SQF in Other Documents
  1244
Formatting and Indenting
  1243
Highlight occurrences
  1248
Integrating in a framework
  1245
Localization
  1245
Multiple similar quick fixes
  1244
Operations
  1237
Refactoring actions
  1248
Restricting operations
  1242
Search actions
  1248
Sharing
  1245
Use-When Condition
  1242

User Entry operation
  1242
Validation
  1247
SQF attribute: use-for-each
  1244
SQF examples
  1194, 1223
SQF multi-lingual support
  1245
SQL transformation scenario
  1544
SSH connection settings
  305
StackOverflowException error
  228

Startup parameter
  Application launcher parameters
  339
  Command-line script parameters
  342
  Custom startup parameters file
  342
  Startup parameters
  339
  Status information
  512
  Storing global options
  311
  Storing project level options
  311
  Storing XML refactoring operations
  875, 2610
  StratML documents
  Editing features
  1183
  Styles Filter plugin extension
  2430
  Styles menu
  590
  Supported document types
  1428
Supported frameworks

- OpenAPI
  1431
- S1000D
  1430

Supported processors for XSLT/XQuery debugging

- 2132

SVG

- Syntax Diagrams
  1962

SVG files

- 1251
  - SVG Viewer
  - 1252, 2677
  - SVG viewer in Results panel
  - 1254

SVG Viewer in Results panel

- 1254

SVG Viewer tool

- 1252, 2677

SVN Client

- 2718
  - Authentication
  - 2735
  - Branches/Tags
  - 2768
    - Create branch/tag
    - 2768
    - Exporting resources
    - 2803
    - Importing resources
    - 2802
    - Manage resources
    - 2806
  - Merging
    - 2770
  - Patching
    - 2789
  - Relocate working copy
  - 2789
  - Switch repository location
  - 2787

Checking out a working copy

- 2739

Define a working copy

- 2738

Entering local paths and URLs

- 2848

History dialog box

- 2741

Manage repository locations

- 2734

Menus

- 2719

Preferences

- 2848

Properties

- 2767

Request history for a resource

- 2767

Request status information for a resource

- 2766

Resource History view

- Directory Change Set view
  - 2833

Revision Graph

- 2843

Share projects

- 2737

Sparse checkouts

- 2807

Status bar

- 2733

Synchronize with the SVN repository

- 2750
  - Conflicts
  - 2753
    - Integrating bug tracking tools
  - 2765
    - Send changes to repository
  - 2761
    - Update working copy
  - 2760
    - View differences
Technical issues
2849
Toolbar
2732
Use an existing working copy
2742
Views
2718
Annotations view
2835, 2843
Compare Images view
2841
Compare view
2837
Dynamic Help view
2843
Editor view
2834
History view
2828
Image Preview panel
2841
Properties view
2841
Repositories view
2808
Working Copy view
2813
Working copy resource management
2743
Add resources
2743
Copy resources
2747
Delete resources
2746
Edit files
2743
Ignore resources
2745
Lock/Unlock resources
2748
Move resources
2748
Rename resources
2748
SVN message preferences
285
SVN preferences
281
Swagger
1431
Switch file tabs
393
Syntax highlight preferences
224
Syntax highlighting in Text Mode
554
Syntax highlights in codeblocks
1663, 3123
Syntax highlights in Text mode
554
System properties
333
Table editing features in Author Mode
683
Table Explorer view
2017
Tables in Author Mode
683
Tags Display Mode
594
Tags hide text
2884
Tags transparency selector
563
Technical Writer helper
2561
TEI ODD document type
1401
Author mode actions
1402
TEI ODD menu actions 1402
TEI ODD toolbar actions 1402
TEI P5 document type 1388
  Author mode actions 1389
TEI P5 menu actions 1389
TEI P5 toolbar actions 1389
TEI tables in Author Mode 713
Templates tab 1493, 3106
Terminology checker 2541
Testing plugins and Java extensions 2446
Text editing mode 353, 517
  Attributes view 542
  Bidirectional text 564
  Content completion 532
  Contextual menu actions 567
Drag and Drop 529
Editing XML markup 523
Elements view 546
Entities view 547, 634
File related actions 565
Folding 528
Format and indent 555
Format and indent multiple files 561, 2645
Highlight ID occurrences 566
Lock/Unlock XML tags 564
Manage highlighted content 562
Markup transparency 563
Model view 545, 631
Navigation 517
Rectangular selection 529
Schema annotations 535
Selecting content 529
Set indent to zero 560
Shortcut actions 521
Smart Editing 520
Special characters 564
Syntax highlighting 554
Syntax highlights 554
Validation errors 777
Views 539
Text flows out of code blocks 2884
Text mode preferences 172
Text rendering issues 2884
Text to markup shortcut patterns 2187
Three-way directory comparison and merge tool 500, 2706
Three-way file comparison 474, 2679
Toolbar configuration 320, 365
Touch Bar on Mac 2877
Track Changes feature in Author Mode 643
Trackpad scroll function doesn't work on Lenovo Thinkpad 2882
Transform DITA document to MS Word 1452, 3098
Transformation parameters 1773
Transformation Scenarios 1432
Ant 1506
Apply transformation scenarios 1548
Batch transformation 1553
Built-in transformation scenarios 1433
Configure 1545
Configure Transformation Scenario dialog box 1548
Configure XSLT processor extension paths 1479, 1526
Custom XSLT processor 1479, 1525
Debugging PDF transformations 1996
DITA map 1434, 3081

DITA Map PDF - based on HTML5 & CSS 1449, 3095
DITA Map to CHM (Compiled Help) 1453, 3099
DITA Map to Kindle 1455, 3101
DITA Map to MS Office Word 1452, 3098
DITA Map to PDF 1451, 3098
DITA Map to WebHelp Responsive 1435, 3082
DITA Map to Zendesk 3101

DITA topic 1457, 3103
DITA-OT 1490, 3104
DocBook 1458
DocBook to DITA 1461
DocBook to EPUB 1462
DocBook to PDF 1462
DocBook to WebHelp 1458
DOCX DITA 3188
Duplicate 1547
Integrate external XProc engine 1534
New 1464
Run DITA-OT Integrator 1455, 3102
Sharing 1555
Show Change Tracking and Comments
Supported XSLT processors
1475, 1522
XML to PDF with CSS
1488
XML with XQuery
1480
XML with XSLT
1464
XProc
1530
XQuery
1536
XSLT
1510
Transformation Scenarios view
1555
Transformation types
1545
Transforming Documents
1432
Translating annotations for the Content Completion Assistant
2212
Translating DITA content
3179
Translation package builder add-on
2548
Translator helper
2547
Tree editor preferences
306
Tree Editor tool
2679
Trusted hosts connection settings
304
Two-way file comparison
474, 2679
U
Unicode support
463
Uninstalling Oxygen
122
Upgrading Oxygen
118
Uppercase plugin
2436
URL Stream Handler plugin extension
2431
Use-When SQF Condition
1242
User Entry SQF operation
1242
User roles in Author Mode
589
Using a floating license offline
106
UTF-8 BOM handling
168
V
Validating JSON Documents
Automatic validation
1115
Check Well-formedness
1114
Manual validation
1115
Resolving references to remote schemas
1122
Sharing scenarios
1121
Validating JSON Documents against a schema
1114
Validating JSON schema
1176
Validating XML Documents
772
Against a schema
775
Against a schema with embedded Schematron
810, 1211
Against Schematron
WebDAV connection
2068
WebDAV connection settings
303
WebDAV contextual menu actions
2069
WebDAV over HTTPS
390
WebHelp Classic
1734
Add custom HTML content
1744
Add Favicon
1751
Add logo image
1752
Add videos in DocBook
1753
Adding additional resources
1753
Changing icons in the table of contents
1749
Comments component
1740
Customize highlights in the table of contents
1751
Customize ordered lists
1749
Disable caching
1766
Edit scoring in search results
1755
Indexing Japanese content
1756
Integrate Facebook
1757
Integrate Google Analytics
1762
Integrate Google Search
1763
Integrate Twitter
1760
Layout and features
1734
Localizing the interface for DocBook transformations
1756
Publishing on SharePoint Site
1767
Right-to-Left languages
1757
Skin Builder
1742
Use custom CSS
1743
WebHelp feature matrix
1559
WebHelp Responsive
1560
  Add link to PDF
  1712
  Adding audio objects
  1674
  Adding favicon
  1673
  Adding logo
  1672
  Adding publishing templates to gallery
  1644
  Adding videos
  1674
  Adding welcome message
  1667
Built-in templates
1605
Changing numbers for ordered lists
1662
Changing scoring values in search results
1676
Comments section
1640
Context-sensitive help
1574
Converting old publishing templates
Workspace Access (JavaScript-based) plugin extension
2418
Workspace Access plugin extension
2415
WSDL documents
1051
  Component Dependencies view
  1062
  Content completion
  1053
  Editing in Main Files context
  1052
  Generate documentation for WSDL documents
  1067, 2656
    Custom format
    1072
    HTML format
    1071
  Highlight occurrences
  1063
  Moving resources
  1062
  Outline view
  1055
  Quick Assist feature
  1066
  Refactoring actions
  1064
  Renaming resources
  1062
  Resource Hierarchy/Dependencies view
  1059
  Search actions
  1064
  SOAP analyzer
  1072, 2670
    Test remote files
    1075, 2672
  Syntax highlighting
  1054
  Validation
  1053
  WSDL SOAP Analyzer tool
  1072, 2670
  Test remote files
  1075, 2672

XHTML document type
1374
  Author mode actions
  1375
XHTML documents
  Validation
  1375
  XHTML menu actions
  1375
  XHTML tables in Author Mode
  710
  XHTML toolbar actions
  1375
  XInclude
  835, 837
  XInclude 1.1 features
  837
XLIFF documents
  Editing features
  1184
XML catalog preferences
235
XML Catalogs
826
XML documents
516
  Associate schema
  816
  Associate schema directly in XML documents
  823
  Associate schema in a framework configuration
  825
  Associate schema in a validation scenario defined in framework
  821
  Associate schema through a validation scenario
  821
<table>
<thead>
<tr>
<th>Page</th>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>818</td>
<td>Author Mode editing</td>
<td>731</td>
</tr>
<tr>
<td>588</td>
<td>Adding media resources</td>
<td>736</td>
</tr>
<tr>
<td>747</td>
<td>Apply profiling attributes</td>
<td>742</td>
</tr>
<tr>
<td>673</td>
<td>Apply profiling condition sets</td>
<td>719</td>
</tr>
<tr>
<td>677</td>
<td>Attributes view</td>
<td>719</td>
</tr>
<tr>
<td>627</td>
<td>Content completion</td>
<td>719</td>
</tr>
<tr>
<td>616</td>
<td>Contextual menu actions</td>
<td>719</td>
</tr>
<tr>
<td>759</td>
<td>Create/Edit profiling attributes</td>
<td>719</td>
</tr>
<tr>
<td>670</td>
<td>Create/Edit profiling condition sets</td>
<td>719</td>
</tr>
<tr>
<td>675</td>
<td>Drag and Drop</td>
<td>719</td>
</tr>
<tr>
<td>612</td>
<td>Editing attributes</td>
<td>719</td>
</tr>
<tr>
<td>610</td>
<td>Editing content</td>
<td>719</td>
</tr>
<tr>
<td>599</td>
<td>Editing XML markup</td>
<td>719</td>
</tr>
<tr>
<td>601</td>
<td>Elements view</td>
<td>719</td>
</tr>
<tr>
<td>633</td>
<td>Entities view</td>
<td>719</td>
</tr>
<tr>
<td>547</td>
<td>Folding</td>
<td>719</td>
</tr>
<tr>
<td>611</td>
<td>Form Controls</td>
<td>719</td>
</tr>
<tr>
<td>758</td>
<td>Generating IDs</td>
<td>719</td>
</tr>
<tr>
<td>755</td>
<td>Image Map Editor</td>
<td>719</td>
</tr>
<tr>
<td>725</td>
<td>DITA</td>
<td>719</td>
</tr>
<tr>
<td>725</td>
<td>DocBook</td>
<td>719</td>
</tr>
</tbody>
</table>

- TEI
- XHTML
- AI images
- CGM images
- EMF images
- EPS images
- JAI images
- PSD images
- Inserting images
- MathML
- MathML equations in HTML output
- Model view
- Navigation
- Profiling
- Profiling colors and styles
- Profiling/Conditional Text menu
- Refreshing content
- Rendering documents
- Review tools
- Callouts
Resolve schemas through XML catalogs

Resource Hierarchy/Dependencies view

Search actions for IDs

Text Mode editing

Attributes view

Content completion

Contextual menu actions

Drag and Drop

Editing XML markup

Elements view

Entities view

File related actions

Folding

Format and indent

Format and indent multiple files

Highlight ID occurrences

Lock/Unlock XML tags

Manage highlighted content

Markup transparency

Model view

Navigation
Text Mode
777
XInclude
835, 837
XML catalogs
826
XML documents without a schema
826
XML parser preferences
237
XML refactoring preferences
269
XML Refactoring tool
840, 2580
Built-in operations
844, 2583
Custom operations
856, 2596
Localizing operations
876, 2611
Sharing operations
875, 2610
Storing operations
875, 2610
XML Schema Design mode
355, 948
XML Schema Diagram Editor
355, 948
Attributes view
995
Components
961
Contextual menu actions
953
Edit namespaces
989
Editing actions
952
Facets view
951
Navigation
948
Outline view
993
Palette view
949
Validation
988
XML Schema Regular Expression Builder tool
1028, 2673
XML Schemas
947
Attributes view
995
Component Dependencies view
1000
Content completion
992
Convert DB Structure to XML Schema tool
1024, 2620
Design mode editing
355, 948
Components
961
Grouping components
986
xs:all
978
xs:alternative
974
xs:any
979
xs:anyAttribute
981
xs:assert
984
xs:attribute
967
xs:attributeGroup
969
xs:choice
978
xs:complexType
969
bg
Text mode editing
989
Validation
990
XML Schema 1.1 support
1030
XML to JSON tool
1134, 2636
XML to PDF with CSS transformation scenario
1488
CSS tab
1488
Output tab
1489
XML transformation with XQuery
1480
XML transformation with XSLT
1464
XPath Expressions
1999
Catalogs
2007
Prefix mapping
2007
Toolbar
2000
XPath Builder view
2002
XPath Results view
2005
XProc Scripts
Editing features
1190
XProc transformation scenario
1530
Inputs tab
1531
Options tab
1534
Outputs tab
1532
Parameters tab
1532
XProc tab
1531
XQuery
1033
Content completion
1035
Folding
1037
Format and Indent
1036
Generate HTML documentation
1045, 2654
Input view
1043
Outline view
1037
Sequence view
1046
Syntax highlighting
1036
Transformations
1046
Sequence view
1046
Updating XML documents
1050
XQJ
2077
Unit test (XSpec)
1050
Updating XML documents
1050
Validation
1034
XQuery Builder view
1039
XQuery Debugger perspective
349, 2102
Breakpoints
2125
Debugging Java extensions

Identify expressions

Information views

Breakpoints view
Context view
Messages view
Nodes/Values Set view
Output Mapping Stack view
Stack view
Templates view
Trace view
Variables view
XPath Watch view

Layout

Performance profiling

Profiling
Hotspots view
Invocation Tree view
Steps in a typical debugging process

Toolbar

XQuery Documentation tool

XQuery Profiler preferences
XQuery transformation scenario
FO Processor tab
Output tab
XQuery tab
XQuery Transformer plugin extension
XSD to JSON Schema tool
XSL-FO processor
xsl:sort for exotic languages

XSLT
Component Dependencies view
Component documentation
Content completion
Content completion in XPath expressions
Editing features
Editing in Main Files context
Generate documentation for XSLT stylesheets
Custom format
HTML format
Highlight occurrences
Input view
Moving resources
Outline view
Quick Assist feature
Quick fix support
Refactoring actions
Renaming resources
Resource Hierarchy/Dependencies view
Search declarations
Search references
Syntax highlighting
Text Value Templates
Unit test (XSpec)
Validating XSLT that call Java extensions
Validating XSLT with custom engines
Validation
XPath Tooltip Helper
XSLT Debugger perspective
Breakpoints
Debugging Java extensions
Debugging XSLT that call Java extensions
Identify expressions
Information views
Breakpoints view
Context view
Messages view
Nodes/Values Set view
Output Mapping Stack view
Stack view
Templates view
Trace view
Variables view
XPath Watch view
Layout
Performance profiling
Profiling
Hotspots view
Invocation Tree view
Steps in a typical debugging process
Supported processors
Toolbar
XSLT preferences
XSLT Profiler preferences
XSLT transformation scenario
FO Processor tab  
1473, 1519  
Output tab  
1474, 1520  
XSLT tab  
1464, 1511  
XSLT Transformer plugin extension  
2433  
XSLTProc preferences  
250  
XSpec Helper View add-on  
2562  
XSpec Test Results view  
2562  

YAML documents  
1177  
  Associate schema directly in YAML documents  
1182  
  Syntax highlighting  
1183  
  Validation  
1178  
  Validation scenarios  
1179  
YAML to JSON tool  
1137, 2632  
YAML validation scenarios  
1179  
Your first DITA document  
39, 2887
Copyright

Oxygen XML Editor User Manual

Syncro Soft SRL.

Copyright © 2002-2020 Syncro Soft SRL. All Rights Reserved.

All rights reserved: No parts of this work may be reproduced in any form or by any means - graphic, electronic, or mechanical, including photocopying, recording, taping, or information storage and retrieval systems - without the written permission of the publisher. Products that are referred to in this document may be either trademarks and/or registered trademarks of the respective owners. The publisher and the author make no claim to these trademarks.

Trademarks: Many of the designations used by manufacturers and sellers to distinguish their products are claimed as trademarks. Where those designations appear in this document, and Syncro Soft SRL was aware of a trademark claim, the designations have been rendered in caps or initial caps.

Notice: While every precaution has been taken in the preparation of this document, the publisher and the author assume no responsibility for errors or omissions, or for damages resulting from the use of information contained in this document or from the use of programs and source code that may accompany it. In no event shall the publisher and the author be liable for any loss of profit or any other commercial damage caused or alleged to have been caused directly or indirectly by this document.

Link disclaimer: Syncro Soft SRL is not responsible for the contents or reliability of any linked Websites referenced elsewhere within this documentation, and Syncro Soft SRL does not necessarily endorse the products, services, or information described or offered within them. Syncro Soft cannot guarantee that these links will work all the time and has no control over the availability of the linked pages.

Warranty: Syncro Soft SRL provides a limited warranty on this product. Refer to your sales agreement to establish the terms of the limited warranty. In addition, Oxygen XML Editor End-User License Agreement, as well as information regarding support for this product, while under warranty, is available through the Oxygen XML Editor End-User License Agreement.

Third-party components: Certain software programs or portions thereof included in the Product may contain software distributed under third-party agreements (“Third-Party Components”), which may contain terms that expand or limit rights to use certain portions of the Product (“Third-Party Terms”). Information identifying Third-Party Components and the Third-Party Terms that apply to them is available on the Third-party License Agreements webpage.

Terms and conditions: For the terms and conditions for using Oxygen XML Editor, see Oxygen XML Editor End-User License Agreement.

Documentation: For the most current versions of documentation, see the Oxygen XML Editor User Manual.

Contact Syncro Soft SRL: Syncro Soft SRL provides telephone numbers and e-mail addresses for you to report problems or to ask questions about your product, see the Oxygen support webpage.