

Oxygen XML Developer 28.1

User Guide

Contents

Chapter 1. Introduction **22**

Chapter 2. Getting Started **23**

What is Oxygen XML Developer 23

Getting Familiar with the Interface 24

Resources to Help You Get Started Using Oxygen XML Developer 26

Your First Document or Project 28

Creating a New Project 28

Chapter 3. Installation **33**

Installing Oxygen XML Developer on Windows 34

Installing Oxygen XML Developer on macOS 38

Installing Oxygen XML Developer on Linux 40

Installing Oxygen XML Developer on Windows Server 45

Installing Oxygen XML Developer on a Linux / UNIX Server 47

Site-Wide Deployment 49

Licensing 50

License Types 51

Installing License Servers 58

Managing License Servers 65

Common Problems: License Server Errors 70

Upgrading 71

Upgrading Oxygen XML Developer on Windows/Linux 72

Upgrading Oxygen XML Developer on macOS 73

Installing and Updating Add-ons 73

Privacy Options 75

Uninstalling 78

Chapter 4. Configuration **79**

Preferences 79

Global Preferences 81

Appearance Preferences 84

Application Layout Preferences 90

Add-ons Preferences 92

Project Level Settings Preferences....92
Document Type Association Preferences....93
Document Templates Preferences....122
Encoding Preferences....123
Editor Preferences....124
CSS Validator Preferences....168
XML Preferences....168
DITA Preferences....200
Markdown Preferences....204
Data Sources Preferences....205
SVN Preferences....210
Diff Preferences....215
Archive Preferences....219
Plugins Preferences....221
External Tools Preferences....221
Menu Shortcut Keys Preferences....223
File Types Preferences....226
Open/Find Resource Preferences Page....227
Custom Editor Variables Preferences....229
Network Connection Settings Preferences....230
XML Structure Outline Preferences....235
Views Preferences....235
Messages Preferences....235
Configuring Options....237
Customizing Default Options....238
Storing Global and Project Level Options....240
Sharing Application Settings....241
Importing/Exporting/Resetting Global Options....242
Configuring the Layout of the Views and Editors....243
Configuring Toolbars....249
Import/Export Transformation or Validation Scenarios....251
Editor Variables....252
Custom Editor Variables....261

Custom System Properties....261
Localizing the User Interface....266
Setting a Java Virtual Machine Parameter when Launching Oxygen XML Developer....267
Setting Parameters for the Application Launchers....267
Setting Parameters in the Command-Line Scripts....270
Creating Custom Startup Parameters File....270
How to Increase the Amount of Available Memory....271
Chapter 5. Perspectives....272
Editor Perspective....272
XSLT Debugger Perspective....274
XQuery Debugger Perspective....275
Database Perspective....276
Chapter 6. Editing Modes....279
Text Editing Mode....279
Grid Editing Mode....279
Design Editing Mode (Schema Diagram Editor)....280
Chapter 7. Working With Documents....283
Getting Familiar with the Interface....283
Configuring the Layout of the Views and Editors....285
Configuring Toolbars....291
Frequently Used Shortcut Keys....293
Creating, Opening, Saving, and Closing Documents....299
Creating New Documents and Templates....299
Opening Documents....313
Saving Documents....316
Auto Recover Documents....316
Closing Documents....317
Working with Remote Documents....318
Open URL....318
WebDAV over HTTPS....322
HTTP Authentication Schemes....324
Switching, Moving, or Hiding Editor Tabs....325
Contextual Menu of the Current Editor Tab....327

Viewing File Properties..... 328
Simple Text Editor.....329
Using Projects to Group Documents..... 330
Creating a New Project.....330
Project View.....334
Batch Validation and Transformation.....346
Sharing a Project - Team Collaboration..... 347
Contextual Project Operations Using 'Main Files' Support..... 350
Search and Find/Replace Features..... 354
Open/Find Resource View..... 354
Open/Find Resource Dialog Box..... 358
Searching in Content.....360
Searching in File Paths.....363
Searching in Reviews.....363
Find/Replace Dialog Box..... 364
Find/Replace in Multiple Files.....368
Find All Elements Dialog Box.....374
Find and Invoke Actions.....376
Quick Find Toolbar..... 378
Keyboard Shortcuts for Finding the Next and Previous Match..... 378
Regular Expressions Syntax..... 379
Spell Checking.....380
Spell Check Dictionaries and Term Lists..... 381
Learned Words..... 387
Ignored Words (Elements).....388
Automatic Spell Check..... 388
Spell Check Multiple Files..... 389
Working with Special Characters and Encoding..... 390
Unicode Support.....391
Opening and Saving Documents with Unsupported Characters..... 392
Unicode Fallback Font Support.....393
Inserting Special Characters with the Character Map..... 394
Image Preview.....397

Loading Large Documents....397
Optimize Loading for Large Files....398
Optimize Loading for Huge Files....399
Documents with Long Lines....399
Handling Read-Only Files....400
Scratch Buffer....400
Compare Files or Directories....401
Compare Files Tool....402
Compare Directories Tool....421
Compare Directories Against a Base (3-Way) Tool....427
Generate HTML Report for Directory Comparison....434
Viewing Status Information....437
Editor Highlights....437
Printing a Document....438
Chapter 8. Editing Supported Document Types....441
Editing XML Documents....442
Editing XML Documents in Text Mode....442
Editing XML Documents in Grid Mode....505
Validating XML Documents....514
XML Quick Fixes....551
Associating a Schema to XML Documents....554
Working with XML Catalogs....564
Modular Contextual XML Editing Using 'Main Files' Support....567
Search and Refactoring Actions for IDs and IDREFS....567
XML Referenced/Dependent Resources View....570
Combining XML Content Using DTD Entities and XInclude....573
Refactoring XML Documents....578
Applying All Default Quick Fix Proposals....615
XML Digital Signatures....617
Editing XSLT Stylesheets....626
Modular Contextual XSLT Editing Using 'Main Files' Support....627
Validating XSLT Stylesheets....627
XSLT Quick Fix Support....630

Content Completion in XSLT Stylesheets....632
Syntax Highlighting in XSLT.... 639
XSLT Outline View.... 639
XSLT Input View.... 644
XSLT Referenced/Dependent Resources View.... 646
XSLT Component Dependencies View.... 649
Highlight Component Occurrences....651
Finding XSLT References and Declarations.... 651
XSLT Stylesheet Component Documentation Support....652
XSLT 3.0 Text Value Templates.... 654
XSLT 3.0 Packages (xsl:package Element)....654
XSLT Refactoring Actions....655
XSLT Quick Assist Support....660
XSLT Unit Test (XSpec)....662
Generating Documentation for an XSLT Stylesheet.... 665
Compiling an XSL Stylesheet for Saxon....672
Convert Between JSON and XML from XSLT.... 674
Editing Ant Build Files.... 675
Modular Contextual Ant Build File Editing Using 'Main Files' Support.... 676
Validating Ant Build Files.... 676
Transforming Ant Build Files....677
Ant Quick Fix Support....678
Content Completion in Ant Build Files.... 678
Syntax Highlighting in Ant Files....679
Ant Outline View....680
Ant Referenced/Dependent Resources View....683
Ant Component Dependencies View....685
Highlight Component Occurrences....686
Find References and Declarations of Ant Components....686
Ant Refactoring Actions.... 687
Ant Quick Assist Support.... 688
Editing XML Schemas (XSD).... 689
XML Schema Design Mode (XML Schema Diagram Editor)....689

Editing XML Schema in Text Editing Mode....731
Modular Contextual XML Schema Editing Using 'Main Files' Support....731
Validating XML Schema Documents....732
Quick Fixes for DTD, XSD, and Relax NG Errors....733
Content Completion in XML Schema....733
Syntax Highlighting in XML Schema....734
XML Schema Outline View....735
XML Schema Attributes View....737
XML Schema Referenced/Dependent Resources View....739
XML Schema Component Dependencies View....742
Highlight Component Occurrences....744
Searching and Refactoring Actions in XML Schemas....744
XML Schema Quick Assist Support....746
Generating Sample XML Files....748
Generating Documentation for an XML Schema....753
Converting Schema to Another Schema Language....763
Converting Database to XML Schema....766
Flatten an XML Schema....767
Generating Java Classes from XML Schema....769
XML Schema Regular Expressions Builder Tool....769
XML Schema 1.1....771
Setting the XML Schema Version....772
Editing XQuery Documents....774
XQuery Validation....774
Content Completion in XQuery....775
Syntax Highlighting in XQuery....776
Formatting and Indenting XQuery Documents....777
Folding in XQuery Documents....777
XQuery Outline View....778
XQuery Builder View....779
XQuery Input View....783
Generating HTML Documentation for an XQuery Document....785
Transforming XML Documents Using XQuery....786

XQuery Unit Test (XSpec)....790
Editing WSDL Documents (Deprecated)....791
Modular Contextual WSDL Editing Using 'Main Files' Support....792
Validating WSDL Documents....793
Content Completion Assistance in WSDL Documents....793
WSDL Syntax Highlighting....794
WSDL Outline View....795
WSDL Referenced/Dependent Resources View in WSDL Documents....800
WSDL Component Dependencies View....802
Highlight Component Occurrences in WSDL Documents....804
Searching and Refactoring Operations in WSDL Documents....804
Quick Assist Support in WSDL Documents....807
Generating Documentation for WSDL Documents (Deprecated)....808
WSDL SOAP Analyzer Tool (Deprecated)....813
Editing CSS Stylesheets....817
Validating CSS Stylesheets....817
Content Completion in CSS Stylesheets....818
Syntax Highlighting in CSS Files....819
CSS Outline View....819
Folding in CSS Stylesheets....820
Formatting and Indenting CSS Stylesheets (Pretty Print)....820
Minifying CSS Stylesheets....820
Editing LESS Stylesheets....821
Validating LESS Stylesheets....822
Content Completion in LESS Stylesheets....822
Syntax Highlighting in LESS Files....822
Compiling LESS Stylesheets to CSS....823
Editing Relax NG Schemas....823
Modular Contextual Relax NG Schema Editing Using 'Main Files' Support....824
Relax NG Schema Diagram Editor....824
Validating Relax NG Schema Documents....828
Content Completion in Relax NG Schemas....829
Syntax Highlighting in Relax NG Schemas....830

Quick Fixes for DTD, XSD, and Relax NG Errors ... 830
Relax NG Outline View ... 831
RNG Referenced/Dependent Resources View ... 834
Relax NG Schema Component Dependencies View ... 837
Searching and Refactoring Actions in RNG Schemas ... 838
RNG Quick Assist Support ... 840
Configuring a Custom Datatype Library for a RELAX NG Schema ... 842
Editing NVDL Schemas ... 842
NVDL Schema Diagram ... 842
Validating NVDL Schema Documents ... 845
Content Completion in NVDL Schemas ... 845
Syntax Highlighting in NVDL Schemas ... 846
NVDL Outline View ... 847
NVDL Schema Component Dependencies View ... 847
Searching and Refactoring Actions in NVDL Schemas ... 848
Editing JSON Documents ... 850
JSON Editor ... 850
Navigating References in JSON Documents ... 852
Validating JSON Documents ... 855
Content Completion Assistant in JSON ... 863
Associating a Schema to JSON Documents ... 865
Syntax Highlighting in JSON Documents ... 870
Folding in JSON ... 870
JSON Outline View ... 870
JSON to XML Converter ... 873
XML to JSON Converter ... 876
JSON to YAML Converter ... 879
YAML to JSON Converter ... 879
Contextual Menu Actions in JSON Documents ... 880
Transforming and Querying JSON Documents ... 885
Editing JSON Schema Documents ... 893
JSON Schema Editor ... 893
JSON Schema Design Mode (JSON Schema Diagram Editor) ... 894

Generating JSON Schema from a JSON File....915
Generating JSON Schema Documentation....917
Generating Sample JSON Files from a JSON Schema....917
XSD to JSON Schema Converter....919
JSON Schema Converter....919
Validating JSON Schema Documents....920
Syntax Highlighting in JSON Schema Documents....921
Flatten JSON Schema....922
Editing JSON Lines Documents....922
Editing JSON5 Documents....922
Editing YAML Documents....923
YAML Editor....923
Validating YAML Documents....924
Content Completion Assistant in YAML....929
Syntax Highlighting in YAML Documents....930
Folding in YAML Documents....930
Formatting/Indenting YAML Documents....930
YAML Outline View....930
YAML to JSON Converter....932
JSON to YAML Converter....933
Contextual Menu Actions in YAML Documents....934
Editing XLIFF Documents....939
Editing JavaScript Documents....940
JavaScript Editing Actions....940
Validating JavaScript Files....943
Content Completion in JavaScript Documents....943
Syntax Highlighting in JavaScript Documents....944
JavaScript Outline View....945
Editing XProc Scripts....946
Editing Schematron Schemas....948
Examples of Schematron Rules and Quick Fixes....950
Modular Contextual Schematron Editing Using 'Main Files' Support....962
Presenting Schematron Validation Issues....962

Integrating Schematron Rules in a Framework and Sharing Them....963
Validating Schematron Documents....964
Content Completion in Schematron Documents....965
Syntax Highlighting in Schematron....966
Embedding Schematron Rules in XML Schema or RELAX NG....967
Schematron Outline View....968
Schematron Referenced/Dependent Resources View....970
Highlight Component Occurrences in Schematron Documents....972
Searching and Refactoring Operations in Schematron Documents....973
Quick Assist Support in Schematron Documents....975
Schematron Unit Test (XSpec)....976
Editing Schematron Quick Fixes....978
Examples of Schematron Rules and Quick Fixes....978
Defining Schematron Quick Fixes....991
Integrating SQF in a Framework and Sharing Them....1000
Validating Schematron Quick Fixes....1002
Content Completion in SQF....1003
Highlight Quick Fix Occurrences in SQF....1003
Searching and Refactoring Operations in SQF....1003
Embedding Schematron Quick Fixes in Relax NG or XML Schema....1005
Editing SVG Files....1006
Standalone SVG Viewer....1007
Integrated SVG Viewer in the Results Panel....1009
Editing HTML Documents....1010
HTML Editor....1010
HTML Validation....1011
HTML Content Completion Assistant....1012
Syntax Highlighting in HTML Documents....1013
Folding in HTML....1014
Minifying HTML Documents....1014
HTML Outline View....1014
Querying HTML Documents with XPath....1015
Editing Markdown Documents....1016

Markdown Editor....1017
Creating New Markdown Documents....1018
Actions Available in the Markdown Editor....1019
Syntax Highlighting in the Markdown Editor....1026
Automatic Validation in Markdown Documents....1026
Markdown Editor Syntax Rules and Specifications....1027
Other Supported Document Types....1040
Chapter 9. Built-in Frameworks (Document Types)....1042
DocBook 4 Document Type (Framework)....1042
Inserting an Olink in DocBook Documents....1043
DocBook 5 Document Type (Framework)....1047
Inserting an Olink in DocBook Documents....1048
DocBook Assembly (5.1 and Later)....1051
DocBook Topic (5.1 and Later)....1052
DocBook Targetset Document Type (Framework)....1053
DITA Topics Document Type (Framework)....1054
DITA Map Document Type (Framework)....1055
S1000D® Document Type (Framework)....1056
XHTML Document Type (Framework)....1058
XHTML Validation....1059
TEI P5 Document Type (Framework)....1059
How to Install a TEI Framework with the Latest Schema and Stylesheets....1060
TEI ODD Document Type (Framework)....1061
jTEI Document Type (Framework)....1062
JATS Document Type (Framework)....1063
EPUB Document Type (Framework)....1064
OpenAPI (Swagger) Document Type (Framework)....1065
OpenAPI Test Scenario Document Type (Framework)....1066
AsyncAPI Document Type (Framework)....1067
JSON-LD Document Type (Framework)....1068
Chapter 10. Additional XML Editing Frameworks (Document Types)....1069
Chapter 11. Publishing....1070
Transformation Scenarios....1070

Built-in Transformation Scenarios.... 1071
Creating New Transformation Scenarios.... 1101
Editing a Transformation Scenario.... 1207
Duplicating a Transformation Scenario.... 1209
Applying Associated Transformation Scenarios.... 1210
Configure Transformation Scenario(s) Dialog Box.... 1210
Batch Transformations.... 1215
Sharing Transformation Scenarios.... 1216
Transformation Scenarios View.... 1217
WebHelp Output Customization.... 1221
WebHelp Responsive Output for DITA.... 1222
DITA to PDF Output Customization.... 1438
CSS-based PDF Customization.... 1438
XSL-FO to PDF Customization.... 1713
DocBook to PDF Output Customization.... 1724
Chapter 12. Working with XPath Expressions.... 1726
XPath Toolbar.... 1727
XPath Builder View.... 1729
XPath Expression Results View.... 1732
XPath and XML Catalogs.... 1733
XPath Prefix Mapping.... 1734
Chapter 13. Working with Archives.... 1735
Browsing Archives.... 1735
Working with Archive Files.... 1738
Creating an Archive.... 1740
Editing and Saving Files Inside an Archive.... 1740
Migrating Archives to DITA or TEI.... 1741
Chapter 14. Databases and SharePoint.... 1742
Working with Databases.... 1742
Data Source Explorer View.... 1742
Table Explorer View.... 1744
Database Connection Support.... 1747
WebDAV Connections.... 1789

SQL Execution Support....1792
XQuery and Databases....1795
Integration with Microsoft SharePoint....1802
How to Configure a SharePoint Connection....1803
SharePoint Browser View....1807
SharePoint Contextual Menu Actions....1809
Browsing for Remote Files with SharePoint Online....1812
MS Azure Active Directory Authentication....1813
Chapter 15. Importing Data....1815
Import from Text Files....1815
Import from MS Excel Files....1817
Import Database Data as an XML Document....1820
Import from HTML Files....1822
Import Content Dynamically....1822
Chapter 16. XSLT/XQuery Debugging....1826
Debugger Layout....1827
Control Toolbar....1828
Debugging Information Views....1832
Multiple Output Documents in XSLT 2.0 and XSLT 3.0....1845
Steps in a Typical Debugging Process....1845
Identify the XSLT / XQuery Expression that Generated Particular Output....1846
Using Breakpoints....1849
Performance Profiling of XSLT Stylesheets and XQuery Documents....1850
Invocation Tree View....1852
Hotspots View....1853
Debugging XSLT that Call Java Extensions....1855
Debugging Java Extensions....1855
Supported Processors for XSLT / XQuery Debugging....1856
Chapter 17. Extending Oxygen With the SDK....1857
Extending Oxygen XML Developer with Plugins....1857
General Configuration of an Oxygen XML Developer Plugin....1857
Installing an Oxygen XML Developer Plugin....1860
Types of Plugin Extensions Available with the SDK....1861

How to Write a CMS Integration Plugin.... 1885
How to Write A Custom Protocol Plugin.... 1890
How to Share a Class Loader Between a Framework and Plugin.... 1890
Packing and Deploying Plugins as Add-ons.... 1891
Testing Plugins and Java Extensions.... 1892
Disabling a Plugin.... 1897
Developer Quick Start Guide.... 1897
Plugins.... 1898
Frameworks.... 1904
Difference Between a Framework (Document Type) and a Plugin Extension.... 1909
SDK Common Use Cases.... 1910
Add Custom Actions to the Contextual Menu.... 1910
Auto-Generate an ID When a Document is Opened or Created.... 1910
Customize the Outline View in Text Mode.... 1911
Open a Document from Another Application.... 1917
Run XSLT or XQuery Transformations.... 1918
Save a New Document with a Predefined File Name Pattern.... 1918
Chapter 18. Add-ons.... 1921
Chapter 19. Tools.... 1922
XML Refactoring.... 1922
Built-in Refactoring Operations.... 1925
Custom Refactoring Operations.... 1938
Storing and Sharing Refactoring Operations.... 1952
Localizing XML Refactoring Operations.... 1953
Generate Sample XML Files.... 1954
Applying All Default Quick Fix Proposals.... 1960
Generate/Convert Schema.... 1962
Convert DB Structure to XML Schema.... 1965
Flatten Schema.... 1966
Generate Java Classes from XSD.... 1967
Compile XSL Stylesheet for Saxon.... 1968
JSON Tools.... 1970
Generate Sample JSON Files.... 1970

Generate JSON Schema....1972
JSON to YAML....1974
YAML to JSON....1974
JSON to XML....1975
XML to JSON....1978
XSD to JSON Schema Converter....1982
JSON Schema Converter....1982
OpenAPI Tester....1983
Run OpenAPI Test Scenario....1983
Format and Indent Files....1984
Generate Documentation....1985
XML Schema Documentation Generator....1986
XSLT Stylesheet Documentation Generator....1989
XQuery Documentation Generator....1993
WSDL Documentation Generator (Deprecated)....1995
JSON Schema Documentation Generator....1998
OpenAPI Documentation Generator....1998
Canonicalize....1999
Sign....2000
Verify Signature....2003
WSDL SOAP Analyzer (Deprecated)....2003
Testing Remote WSDL Files....2005
XML Schema Regular Expressions Builder....2006
Large File Viewer....2007
Hex Viewer....2009
SVG Viewer....2010
Tree Editor (Deprecated)....2012
Comparison Tools....2012
Compare Files....2012
Merge Documents with Change Tracking Highlights....2039
Compare Directories....2040
Compare Directories Against a Base (3-Way)....2046
Generate HTML Report for Directory Comparison....2053

Merge Directories with Change Tracking Highlights....2056
Syncro SVN Client (Deprecated)....2060
Main Window....2061
Getting Started....2076
Syncro SVN Client Views....2152
Revision Graph of an SVN Resource....2188
Oxygen XML Developer SVN Preferences....2192
Entering Local Paths and URLs....2192
Technical Issues....2194
External Tools....2197
AI Positron Overview....2198
AI Translator Overview....2199
Chapter 20. Troubleshooting....2200
Getting Help....2200
Help Menu....2200
Performance Problems and Solutions....2204
Display Problems on Linux or Solaris....2205
Out of Memory on External Processes....2205
Too many nested apply-templates calls Error When Running a Transformation....2206
Performance Issues with Large Documents....2206
Performance Issues when Using Oxygen XML Developer with Remote Desktop....2207
Misc Problems and Solutions....2207
Address Family Not Supported by Protocol Family....2207
Application Reports Errors During Startup After Installing a New Version....2208
Application Takes Several Minutes to Start....2209
Archive Distribution Fails to Run on macOS 10.12 (Sierra)....2210
Blank Window is Shown When Starting the App Over an RDP Connection on Linux....2210
Cannot Connect to SVN Repository from Repositories View....2211
Cannot Open Files from Desktop/Downloads/OneDrive on macOS....2212
Cannot Uninstall Oxygen XML Developer in Windows....2213
Compatibility Issue Between Java and Certain Graphics Card Drivers....2213
Crash at Startup on Windows with an Error About the nvoglv32.dll File....2213
Damaged File Associations on macOS....2214

Details to Submit in a Request for Technical Support Using the Online Form..... 2215
Dialog Boxes Cannot Be Resized on Mac.....2216
DITA Map Transformation Fails (Cannot Connect to External Location).....2216
DITA Map WebHelp Transformation Fails (Duplicate Topic References Found).....2216
DITA-OT Transformation Takes a Long Time to Process..... 2217
DITA PDF Transformation Fails..... 2218
DITA PDF Processing Common Errors.....2218
DITA PDF CSS-based Processing Common Errors..... 2220
DITA to CHM Transformation Fails - Cannot Open File..... 2220
DITA to CHM Transformation Fails - Compilation Failed.....2220
Fonts Installed in Windows Do Not Appear in Fonts Preferences Page.....2221
Format and Indent Fails..... 2221
Handshake Failure Error When Accessing an HTTPS (SSL) Resource.....2221
Hunspell Spell Checker is Unusable on Your Platform Error.....2222
High Resolution Scaling Issues..... 2223
High Resolution Scaling Issues on Linux.....2223
Images Appear Stretched Out in the PDF Output..... 2223
Increasing the Memory for the Ant Process.....2224
Java Virtual Machine (JVM) Crash on macOS.....2225
Keyboard Language Resets to Default on Windows.....2225
Keyboard Shortcuts Do Not Work At All..... 2225
Keyboard Shortcuts Result in Unexpected Behavior.....2226
Mac Touch Bar Function Keys Do Not Work..... 2226
Server Signature Mismatch Error.....2227
Navigation to a Web Page is Canceled when Viewing CHM on a Network Drive.....2228
Out Of Memory Error When Opening Large Documents.....2228
References Outside the Main DITA Map Folder..... 2229
Saxon 9.7 Transformer Issues.....2229
Scroll Function of my Notebook Trackpad is Not Working..... 2230
Special Characters are Replaced with a Square.....2230
TocJS Transformation Does not Generate All Files for a Tree-Like TOC.....2231
Text on Buttons and Labels is Invisible for Linux Installer.....2231
XML Document Takes a Long Time to Open..... 2232

XSLT Debugger Is Very Slow....2232
Chapter 21. Accessibility Support in Oxygen.... 2233
Oxygen XML Developer VPAT Accessibility Conformance Report....2236
Chapter 22. Scripting Oxygen.... 2263
DITA Validate and Check For Completeness.... 2263
Transform.... 2264
Validate.... 2265
XML Refactoring.... 2269
DITA Translation Package Builder....2270
Batch Converter.... 2272
Compile Framework Script.... 2273
XSLT Stylesheets Documentation.... 2274
XML Schema Documentation....2275
JSON Schema Documentation....2275
OpenAPI Documentation....2277
WSDL Documentation (Deprecated).... 2279
XML Instance Generator.... 2280
Flatten XML Schema....2281
Compare Directories....2281
Compare Files.... 2288
Providing Arguments to Comparison Scripts via a Configuration File.... 2293
Merge Files with Change Tracking Highlights....2296
Merge Directories with Change Tracking Highlights.... 2298
Format and Indent Files....2301
Chapter 23. Glossary....2303
Active Cell....2303
Anchor....2303
Apache Ant....2303
Block Element.... 2303
Bookmap....2303
Canonicalize.... 2303
Content Completion Assistant....2303
Dockable.... 2304

Document Type Association....2304
DITA Map....2304
DITA Open Toolkit....2305
DITA-OT-DIR....2305
Foldable Element....2305
Framework....2305
Global Options....2305
IDML....2306
Inline Element....2306
Java Archive....2306
Key Space....2306
Keystore....2306
Main File....2306
Perspective....2307
Plugin....2307
Pretty-Print....2307
Project Options....2308
QName....2308
Quick Fix/Assist....2308
Quick Fix....2308
Root Map....2309
WebHelp Output Directory....2309
Working Set....2309
XML Catalog....2309
Index....a
Copyright....ba

1.
Introduction

Welcome to the User Manual of Oxygen XML Developer 28.1.

Oxygen XML Developer 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 offers developers a powerful **Integrated Development Environment** and the intuitive **Graphical User Interface** of Oxygen XML Developer 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 Developer offers support for generating output to multiple target formats, including: *PDF*, *PS*, *TXT*, *HTML*, *JavaHelp*, *WebHelp*, and *XML*.

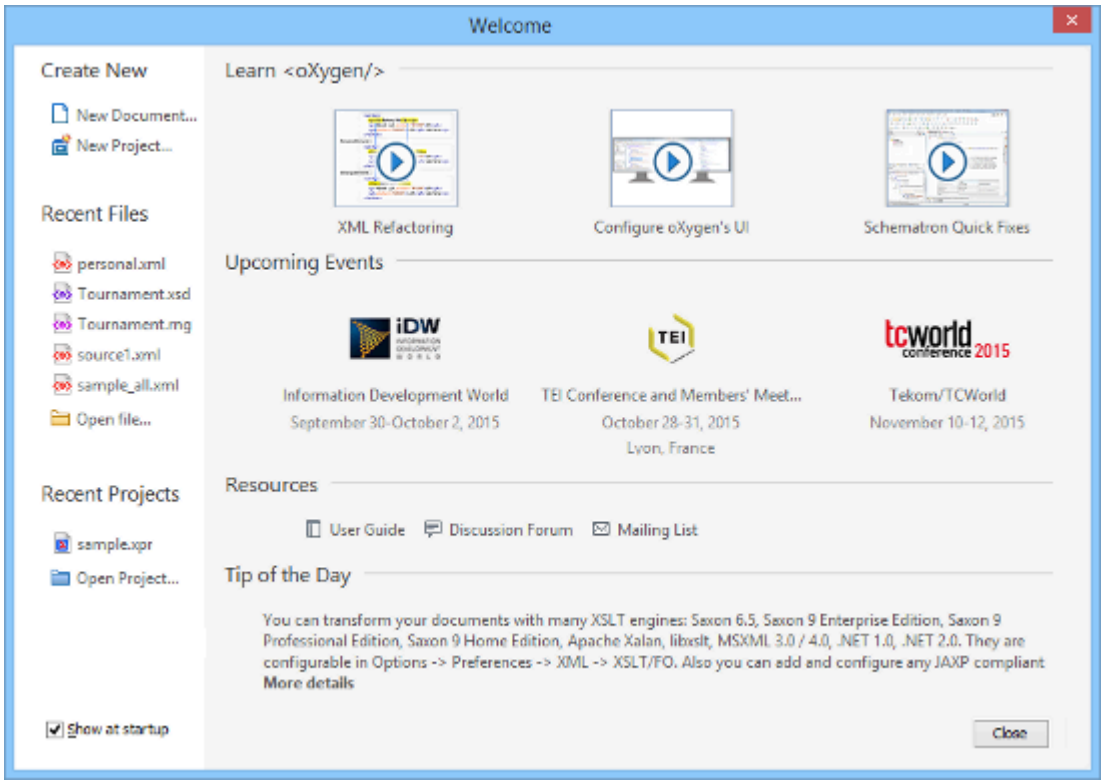
This user guide is focused on describing features, functionality, the application interface, and to help you quickly get started. It also includes a vast amount of advanced technical information and instructional topics that are designed to teach you how to use Oxygen XML Developer to accomplish your 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 Developer would be to install the software. For detailed information about that process, see the Installation chapter *(on page 33)*.

After installation, when you launch Oxygen XML Developer 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



If you do not want it to be displayed every time you launch Oxygen XML Developer, 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 Developer

Oxygen XML Developer 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 Developer 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.

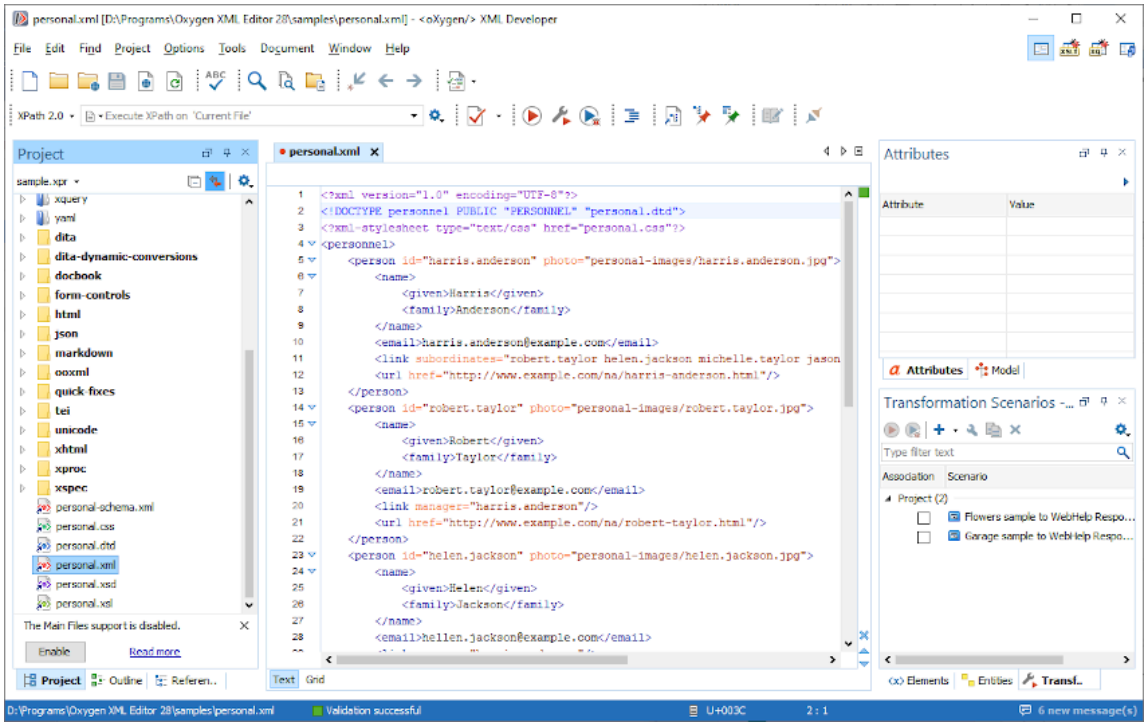
For a list of many of the features and technologies that are included in Oxygen XML Developer, see the Oxygen Website.

Getting Familiar with the Interface

Oxygen XML Developer includes several *perspectives (on page 2307)* and editing modes *(on page 279)* 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 285)*, and you can customize the toolbars *(on page 291)*.

Figure 2. Clickable Image Map of the Oxygen XML Developer Interface



1. Menus *(on page 25)*
2. Toolbars *(on page 25)*
3. Left-side Helper Views *(on page 25)*
4. Right-side Helper Views *(on page 25)*
5. Editor Pane *(on page 25)*
6. Perspectives *(on page 25)*
7. Status Bar *(on page 25)*
8. Editing Modes *(on page 279)*

Regardless of the *perspective (on page 2307)* or editing mode 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 Developer. Most of the menus are common for all types of documents, but Oxygen XML Developer 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 291)* to suit your specific needs.

Helper Views

Oxygen XML Developer includes a large variety of *dockable (on page 2304)* 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 285)* according to your preferences.

Editor Pane

The main editing area in the center of the application. Each editing mode 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 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 325)*.

Perspectives

Oxygen XML Developer includes several different *perspectives (on page 272)* 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 Developer for the first time. Oxygen XML Developer 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 391)* 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, the current line and character position is displayed.
- If the **Check for notifications** option *(on page 81)* 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 Developer website, the announcement of upcoming webinars and conferences where the Oxygen XML Developer 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 83)* 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.

Resources to Help You Get Started Using Oxygen XML Developer

Configuring Oxygen XML Developer

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

See the Configuring Oxygen section *(on page 79)* for details on the various ways that you can configure the application and its features.

Video Tutorials and Webinars

The Oxygen XML Developer website includes numerous video demonstrations and webinars that present many of the features that are available in Oxygen XML Developer 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 Developer Documentation

The Oxygen XML Developer 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 279)*** - Provides information about the **Text** editor.
- **XML Schema Diagram Editor** *(on page 280)* - Provides information about the schema design mode.
- **Editing Specific Document Types Chapter** ***(on page 441)*** - Includes information about editing numerous different types of documents.
- **Publishing Chapter** *(on page 1070)* - Provides information about the various ways that you can publish content.
- **Importing Data Chapter** ***(on page 1815)*** - Provides information about importing data from text files, MS Excel files, database data, and HTML files.
- **Tools Chapter** *(on page 1922)* - Details about the various built-in tools that are available in Oxygen XML Developer.
- **Add-ons Chapter** *(on page 1921)* - Information about how to extend the functionality of Oxygen XML Developer through add-ons.

Sample Documents

Your installation of Oxygen XML Developer 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 Developer. 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 334)*.
- **Sample files *(personal.xml, etc.)*** - A collection of interrelated sample files that will allow you to experiment with the structure and relationship between 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 Developer:

- See the Oxygen XML Developer Blog Site for a large variety of current and archived blogs regarding numerous features, requests, and instructional topics.
- Take advantage of the Oxygen XML Developer 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 Developer, see the Publishing DITA to WebHelp section of the website.

- For more information about various additional tools that are integrated into Oxygen XML Developer, see the Tools section *(on page 1922)*.
- 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 Developer *frameworks (on page 2305)* and *plugins (on page 2307)*, and to integrate Eclipse plugins.
- For a list of new features that were implemented in the latest version of Oxygen XML Developer, see the What's New Section on the website.
- You can select the **Tip of the Day** *(on page 2202)* action in the **Help** menu *(on page 2200)* to display a dialog box that includes a variety of tips for using Oxygen XML Developer.
- You can select **Show Dynamic Help view** *(on page 2200)* from the **Help** menu *(on page 2200)* 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.

Creating a New Project

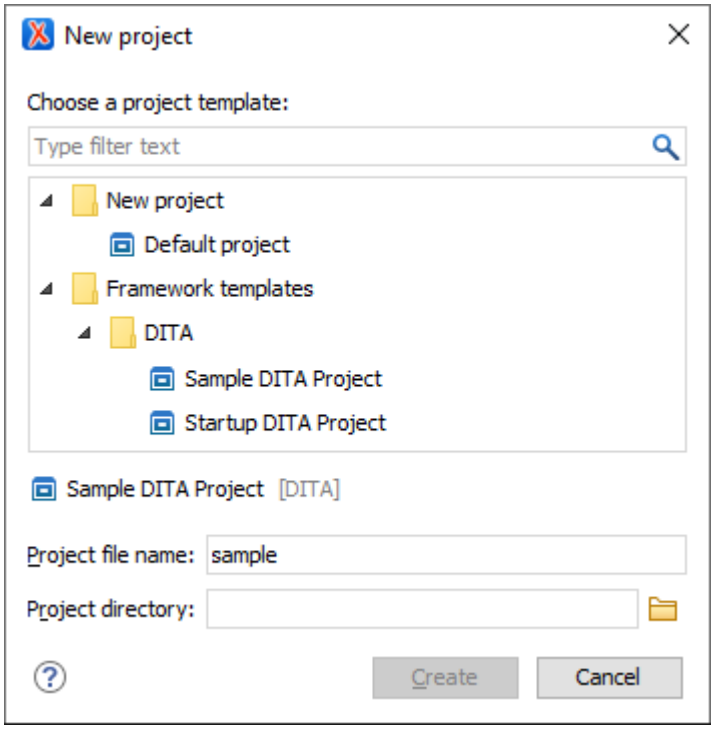
Oxygen XML Developer allows you to organize your files into projects. Projects help you manage and organize your files and allow you to perform batch operations (such as validations and transformations) over multiple files. You can also share your project settings and transformation/validation scenarios *(on page 347)* with other users. Use the **Project** view *(on page 334)* 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 3. 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 331)*

- **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 1735)*.

After selecting a project template, you can specify the following:

Project file name

Specifies the name of the new project file. Oxygen XML Developer 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 Developer 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 309)* 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 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 334)* 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 334)* 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 334)* 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 334)*.

For more details about how you can share projects with other users, see Sharing a Project - Team Collaboration *(on page 347)*.

Related information
Using Projects to Group Documents *(on page 330)*

3.
Installation

Oxygen XML Developer is available on Windows, Linux, and macOS and there are a variety of methods and options for installing and running Oxygen XML Developer 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 Developer Runs

You can install Oxygen XML Developer 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 *(on page 51)* to run Oxygen XML Developer.

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 71)* Oxygen XML Developer, transfer a license *(on page 53)*, or uninstall *(on page 78)* Oxygen XML Developer.

Getting Help With Installation

If you need help, email support at: support@oxygenxml.com.

Installing Oxygen XML Developer on Windows

System Requirements

Operating Systems

The product has been fully tested on Windows versions 10 and 11. The latest version of Oxygen XML Developer 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 Developer only officially supports Java Virtual Machines with version 21 from Oracle or Eclipse Adoptium. If you use the native Windows installer, Oxygen XML Developer 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 21 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 Developer may work with other versions of Java, but since Oxygen XML Developer has only been thoroughly tested with specific versions, there is no guarantee that it will be stable with any other Java version.

Oxygen XML Developer 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 Developer 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 Developer 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 Developer using the batch file, `oxygenDeveloper.bat`, you can edit the batch file to specify a particular version to use.

Windows Installer

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

1. Make sure that your system meets the system requirements *(on page 34)*.
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 Developer using one of the following methods:
   - Use one of the shortcuts created by the installer.
   - Run `oxygenDeveloper.bat`, which is located in the installation directory.
6. To license your copy of Oxygen XML Developer, go to **Help > Register** and enter your license information *(on page 51)*.

Windows 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 Developer 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 Developer installer for Windows supports a variety of 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 38)* 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 Developer 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-64bit-openjdk.exe -q -console`). Otherwise, it will run in the background.

-varfile

Specifies the location of a `response.varfile` *(on page 38)*, 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-64bit-openjdk.exe -q -overwrite -console -VautoVersionChecking=false
```

Command-Line Variables for Preconfiguring License Server Details

For organizations that use a license server to manage user licenses, the Oxygen XML Developer 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.developer`).

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.developer`).

reportProblem

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

EXAMPLE:

```
oxygen-64bit-openjdk.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"
```

Note:
For information about deploying and registering floating licenses for multiple users, see Registering Floating Licenses for Multiple Users *(on page 57)*.

Windows Installer *response.varfile*

The Oxygen XML Developer 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 35)*. 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**.

checkNotifications

Used to control checking for application-related news and events. 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 to populate the application welcome screen. Possible values are **true** (default) or **false**.

reportProblem

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

installAddons

Set to **false** to prohibit the application from allowing the end user to install add-ons.

installAIPositronAddons

Set to **false** to prohibit the application from allowing the end user to install any of the **AI Positron** add-ons.

allowAIPositronService

Set to **false** to prohibit the **AI Positron** add-on from connecting and using models from the **AI Positron Service** platform.

Installing Oxygen XML Developer on macOS

System Requirements

Operating system

The product has been fully tested on macOS 14 (Sonoma), 15 (Sequoia), and 26 (Tahoe). The latest version of Oxygen XML Developer might work on older versions of macOS, but they have not been officially tested.

CPU

- Minimum - Intel-based Dual-core Mac, 2 GHz
- Recommended - Apple M1 or newer

Memory

- Minimum - 4 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 Developer on macOS, follow these steps:

1. Download the macOS installation package (oxygenDeveloper-openjdk.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 oxygenDeveloper-openjdk.dmg disk image file to mount it.
4. Drag/Copy the *Oxygen XML Developer* folder to your `/Applications` folder (or another location if you wish).

   Warning:
   If you receive a warning that an *Oxygen XML Developer* 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 Developer* folder (always copy the folder itself), otherwise you will omit invisible files/folders and the application may no longer start.

5. Start Oxygen XML Developer, using one of the following methods:

- Double-click `Oxygen XML Developer.app`.
- Run `sh oxygenDeveloper.shsh oxygenJSONEditor.sh` in the command-line interface.

6. To license your copy of Oxygen XML Developer, go to **Help > Register** to enter your license key *(on page 51)*.

macOS Unattended Installation

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

1. Download the macOS installation package (oxygenDeveloper-openjdk.dmg).
2. Mount the oxygenDeveloper-openjdk.dmg file in the command line.

```
hdiutil attach oxygenDeveloper-openjdk.dmg
```

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:

```
cp -aR "/Volumes/Oxygen XML Developer 28.1/Oxygen XML Developer" /Applications/|
```

4. Eject the mounted disc image:

```
hdiutil detach "/Volumes/Oxygen XML Developer 28.1"|
```



Note:
For information about deploying and registering floating licenses for multiple users, see Registering Floating Licenses for Multiple Users *(on page 57)*.

Installing Oxygen XML Developer on Linux

System Requirements

Operating System

The product has been fully tested on Ubuntu 22.04. The latest version of Oxygen XML Developer 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 Developer only officially supports Java Virtual Machines with version 21 from Oracle or Eclipse Adoptium. If you use the Linux installer, Oxygen XML Developer 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 21 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 Developer may work with other versions of Java, but since Oxygen XML Developer 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 Developer does not work with the **GNU libgcj** Java Virtual Machine.

Oxygen XML Developer 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 Developer 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 Developer 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 Developer by editing the script file, `oxygenDeveloper.sh`.

X.org

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

Linux Installer

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

1. Make sure that your system meets the system requirements *(on page 40)*.
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 ./oxygenDeveloper-64bit-openjdk.sh`.

   Warning:
   If you are running the installer as root and your Linux distribution uses Wayland (such as **Ubuntu 17.10** or **Fedora 25**), before running the installer, the local user must first allow the root user to access the X server by running the following command (as the local user):

   ```
   xhost +SI:localuser:root
   ```

5. Start Oxygen XML Developer using one of the following methods:
   - Use the `developer` shortcut created by the installer.

     Note:
     For **Ubuntu 17.10** (or later), a security dialog box will appear the first time you start the application where you need to select **Trust and Launch** to continue. This dialog box will not appear on subsequent launches.

   - From a command line, type `sh oxygenDeveloper.sh`. This file is located in the installation folder.
6. To license your copy of Oxygen XML Developer go to **Help > Register** and enter your license information *(on page 51)*.

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 Developer 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 Developer installer for Linux supports a variety of 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 44)* 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 Developer 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 44)*, normally to be used during an unattended installation.

-V

Used to define a variable parameter *(on page 44)* to be used by an installation.

EXAMPLE:

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

Command-Line Variables for Preconfiguring License Server Details

For organizations that use a license server to manage user licenses, the Oxygen XML Developer 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.developer`).

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.developer`).

reportProblem

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

EXAMPLE:

```
oxygen.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"
```

Note:
For information about deploying and registering floating licenses for multiple users, see Registering Floating Licenses for Multiple Users *(on page 57)*.

Linux Installer *response.varfile*

The Oxygen XML Developer 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 42)*. For more information about the `response.varfile` format, see 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**.

checkNotifications

Used to control checking for application-related news and events. Possible values are **true** (default) or **false**.

reportProblem

Used to report a problem encountered while using Oxygen XML Developer. 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 to populate the application welcome screen. Possible values are **true** (default) or **false**.

installAddons

Set to **false** to prohibit the application from allowing the end user to install add-ons.

installAIPositronAddons

Set to **false** to prohibit the application from allowing the end user to install any of the **AI Positron** add-ons.

allowAIPositronService

Set to **false** to prohibit the **AI Positron** add-on from connecting and using models from the **AI Positron Service** platform.

Installing Oxygen XML Developer 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 Developer only officially supports Java Virtual Machines with version 21 from Oracle or Eclipse Adoptium. If you use the native Windows installer, Oxygen XML Developer 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 21 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 Developer may work with other versions of Java, but since Oxygen XML Developer has only been thoroughly tested with specific versions, there is no guarantee that it will be stable with any other Java version.

Oxygen XML Developer 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 Developer 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 Developer 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 Developer using the batch file, `oxygenDeveloper.bat`, you can edit the batch file to specify a particular version to use.

Windows Installer

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

1. Make sure that your system meets the system requirements *(on page 45)*.
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 Developer using one of the following methods:
    - Use one of the shortcuts created by the installer.
    - Run `oxygenDeveloper.bat`, which is located in the installation directory.
6. To license your copy of Oxygen XML Developer go to **Help > Register** and enter your license information *(on page 51)*.

Configuring Windows Terminal Server

1. Install Oxygen XML Developer on the server and make its shortcuts available to all users.
2. Make sure you allocate sufficient memory to Oxygen XML Developer by adding the `-Xmx` parameter either in the *.bat* startup script *(on page 267)*, or in the *.vmoptions* configuration file *(on page 270)* (if you start it from an executable launcher).

Installing Oxygen XML Developer on a Linux / UNIX Server

System Requirements

Operating system

The product has been fully tested on Ubuntu 22.04. The latest version of Oxygen XML Developer 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 Developer only officially supports Java Virtual Machines with version 21 from Oracle or Eclipse Adoptium. If you use the Linux installer, Oxygen XML Developer 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 21 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 Developer may work with other versions of Java, but since Oxygen XML Developer 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 Developer does not work with the **GNU libgcj** Java Virtual Machine.

Oxygen XML Developer 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 Developer 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 Developer 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 Developer by editing the script file, `oxygenDeveloper.sh`.

Linux Installer

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

1. Make sure that your system meets the system requirements *(on page 47)*.
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 Developer using one of the following methods:
   - Use the `developer` shortcut created by the installer.
   - From a command line, type sh oxygenDeveloper.sh. This file is located in the installation folder.
6. To license your copy of Oxygen XML Developer go to **Help > Register** and enter your license information *(on page 51)*.

Unix/Linux Server Configuration

1. Install Oxygen XML Developer on the server and make sure the oxygenDeveloper.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 Developer by setting an appropriate value for the `-Xmx` parameter in the `.sh` startup script.



Note:

The default value of the `-Xmx` parameter is about a quarter of the maximum internal memory available on the machine. To avoid performance issues with large documents *(on page 2206)*, 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 Developer by typing `sh oxygenDeveloper.sh` from the command line. This file is located in the installation folder.

Site-Wide Deployment

If you are deploying Oxygen XML Developer for a group, there are various things you can do to customize Oxygen XML Developer 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 238)* for Oxygen XML Developer. 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 Developer, 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 Developer project files *(on page 330)* are used to configure a project. You can create and deploy default project files *(on page 330)* 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 347)* 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 35)* or Linux *(on page 42)* installs.

Using floating licenses

If you have a number of people using Oxygen XML Developer on a part-time basis or in different time zones, you can use a floating license *(on page 55)* so that multiple people can share a license.

Licensing

This section contains information about licensing Oxygen XML Developer, including details about the types of licenses that are available, managing licenses, using a license server to manage licences for an organization, and information about managing 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/Author/Developer**, you must set up an Oxygen License Server *(on page 58)*.

Registering License Keys

To help you comply with the **Oxygen** EULA (terms of licensing), all floating or named-user licenses must be registered. 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 deployment of the license server is sent to the **Oxygen** servers. The servers will then 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 version of **Oxygen XML Editor/Author/Developer** 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.

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

- You can purchase one or more licenses from the Oxygen XML Developer 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 Developer website at https://www.oxygenxml.com/register.html.

License Types

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

The following license types are available:

- A **Named-User License** *(on page 52)* may be used by a single *Named User* 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.
- A **Floating License** *(on page 55)* may be used by any user on any machine. However, the total number of copies of Oxygen XML Developer 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 Developer (for example, Standalone and Eclipse Plugin) at the same time on the same computer, consumes a single floating license.
- A **Subscription** license 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.

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.

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

Named-User Licenses

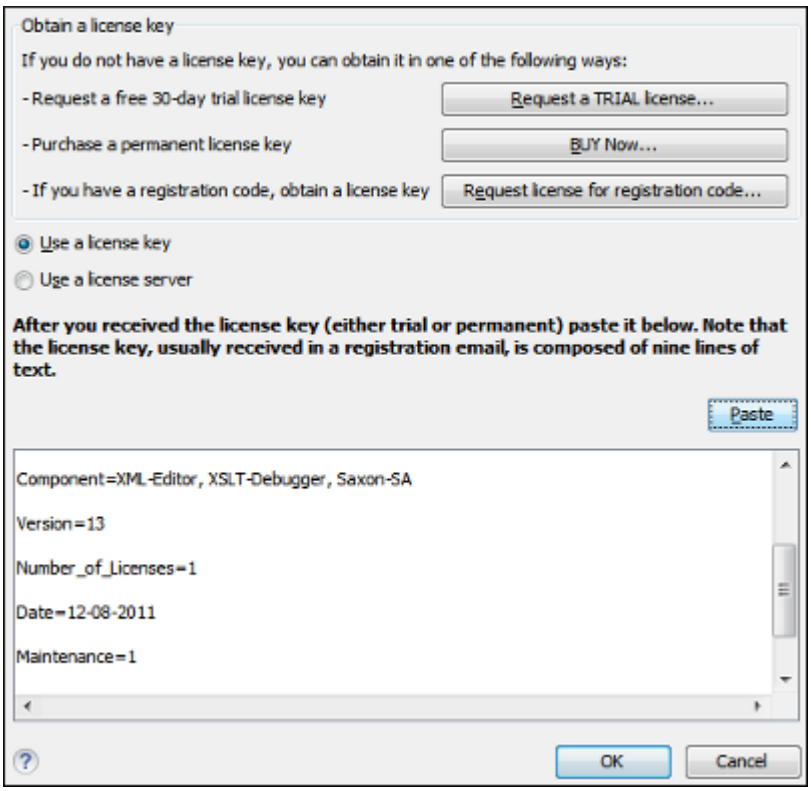
A **Named-User License** can be used by a single *Named User* on one or more computers. Named-user licenses cannot be transferred to another *named user*. If you purchase multiple named-user licenses, you will receive a single license key that is valid for a specified number of *named users*. It is your responsibility to keep track of which *named users* are assigned to each license.

Registering a Named-User License

To register a *Named-User License* on a machine owned by the *Named User*, follow these steps:

1. Purchase a license from the Oxygen XML Developer 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 Developer.
   If this is a new installation of Oxygen XML Developer, the registration dialog box is displayed. If the registration dialog box is not displayed, go to **Help > Register**.

   Figure 4. 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 50)* 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 Developer End-User License Agreement

Transferring a License Key

If you want to transfer your Oxygen XML Developer 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 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.

Subscription Licenses

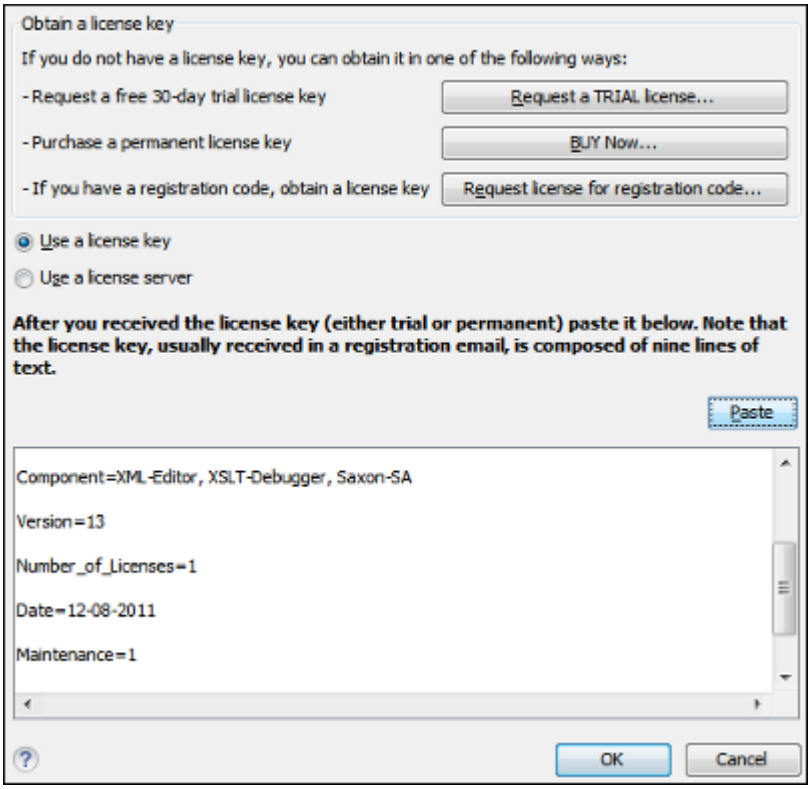
A **Subscription** license that allows you to use the application for a specific period of time (either 6 months or 1 year). This type of license is user-based and is covered by a Support and Maintenance Pack, which means that during the subscription period you will get free upgrades to all major and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

Registering a Subscription License

To register a *Subscription License*, follow these steps:

1. Purchase a license from the Oxygen XML Developer 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 Developer.
   If this is a new installation of Oxygen XML Developer, the registration dialog box is displayed. If the registration dialog box is not displayed, go to **Help > Register**.

Figure 5. 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 50)* 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 Developer End-User License Agreement

Automatic Check for Subscription Renewal

The **Oxygen License Server** has a mechanism that tries to detect when you purchase a renewal of your current subscription and automatically updates the license key.

To determine that a license key you purchased is a renewal of the license key you have currently installed in the License Server, it uses the **Previous order reference number** if you inserted it in the checkout process.

This automatic renewal mechanism makes an HTTP request to *https://oxygenxml.com/subscription_management/check_renewal.php* and passes the following as parameters:

- The SGN field of the existing license key.
- The server signature that uniquely identifies the License Server installation.

This request is made by the License Server automatically (or manually by pressing the **Check now** link).

This mechanism can be disabled by deselecting the **Automatically check for subscription renewal** checkbox.

Floating Licenses

The floating license type is commonly used by organizations that have a large number of infrequent users who do not require simultaneous access to the application. Instead of each user having their own individual license key, a pool of licenses is available for use on demand, one at a time.

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 58)*. Then you will need to request a floating license from the server *(on page 55)*. This process is designed to ensure compliance 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 56)*.

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 57)*.

Requesting a Floating License from a 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 Developer.
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 Developer. 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 Multiple Users

If you are an administrator and you want to register floating licenses for multiple users without having to open Oxygen XML Developer 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 Developer:
    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 55)*.

    Step Result: A `license.xml` file is created.
3. Copy the `license.xml` file from the `preferences` directory *(on page 80)* and place it in the installation folder on each machine to be registered.

Related information
Installing License Servers *(on page 58)*

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 Developer running on your machine are using your floating license.
- You register a *Named User* license with your copy of Oxygen XML Developer, and no other copies of Oxygen XML Developer 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 65)*.

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 66)*.

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* 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 Developer.

Reserving a Floating License

There may be times when you need 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. **Important:** The license server must be configured by the server administrator to allow license locking. See License Server Management and Statistics - Configuration *(on page 65)*.
2. Select **Lock floating license** from the **Help** menu.
3. 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.

   Note:

   A locked floating license uses one license from the license pool for the entire period it remains locked.

Registering Floating Licenses for Multiple Users

If you are an administrator and you want to register floating licenses for multiple users without having to open Oxygen XML Developer 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 Developer:
    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 55)*.

    Step Result: A `license.xml` file is created.
3. Copy the `license.xml` file from the `preferences` directory *(on page 80)* and place it in the installation folder on each machine to be registered.

Related information
Requesting a Floating License from a License Server *(on page 55)*
Installing License Servers *(on page 58)*

Installing License Servers

If you are using floating licenses or a large number of user-based licenses (20 or more) for **Oxygen XML Editor/Author/Developer**, you must set up an **Oxygen License Server**.

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

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

HTTP License Server System Requirements

Table 1. Minimum Requirements

Hardware	Specification
CPU	1 core
RAM	512 MB/Linux OS, 1 GB/Windows OS (256 MB available memory)
Hard Disk Space	500 MB
Network Requirements	**Network interfaces stay unchanged (static MAC addresses) after activation**

Table 1. Minimum Requirements (continued)

Hardware	Specification
Server OS Requirements	• Linux • Windows (Server 2022 is supported)
Antivirus and Firewall Requirements	Allow access to the configured TCP port (default 8080)

Note:
Oxygen XML Editor/Author/Developer 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.

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

License Activation

The activation process involves binding your license key to a unique license server deployment. Once the process is complete, you cannot activate the license key with another license server deployment.

To activate the license key directly from License Server's UI, you need to access it from a browser with internet access.

Note:
During this process, the activation signature, provided license key, and any configured public URLs are sent to `oxygenxml.com`.

Activation Signature

The default signature calculation is *"Use machine signature (MAC, etc.)"* where the *activation signature* of the License Server instance is computed from various machine IDs.

The **Use the public URL of the License Server** option offers a more flexible activation signature, as you can choose where you provide the exact URL that *Oxygen* applications will use to request a license (the URL should be in the format of: `http(s)://my-domain:port/context-name/license-servlet`). This reduces number of machine IDs used to compute the *activation signature*.

Note:
The public server URL does not have to be accessible from outside your network. It should not be a local URL (e.g. `localhost` and `127.0.0.1`) as they are not supported.

Manual License Activation Procedure

If you cannot access the License Server administration page from a browser that has internet access (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 has not yet been 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 selecting **Replace** on the next page.

3. Go to the activation page at: http://www.oxygenxml.com/activation/.
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 Developer

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

- **Windows:** Windows Installation: Command-Line Parameters for Preconfiguring License Server Details *(on page 36)*.
- **Linux:** Linux Installation: Command-Line Parameters for Preconfiguring License Server Details *(on page 43)*.

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 70)*

Installing the License Server Distribution for Windows

1. Download the HTTP license server installer from the HTTP License Server website.
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** license server administrative interface.
    b. **Standard user credentials** - Used by an **Oxygen** 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 edit the following *vmoptions* file: `oXygen HTTP License Server\Windows Service\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`.

HTTPS Support

The All-Platform Distribution of the HTTP License Server do not provide the means to use HTTPS (TLS). If you need HTTPS, you should install the HTTP License Server WAR Distribution *(on page 62)* on Tomcat (preferably 9.x). See the WAR Distribution HTTPS Support *(on page 64)* section for more details.

Installing the License Server All-Platform Distribution

1. **[Prerequisite]** Java 11 or later must be installed.
2. Download the HTTP license server all-platform archive from the HTTP License Server website.
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 port other than the default 8080, you can pass a new port number as an argument to the scripts (for example, `licenseServer.bat 8082`). You can also change the port by editing the *vmoptions* file located at `oXygen HTTP License Server\Windows Service\oXygenHTTPLicenseServer.vmoptions`.

5. On the first run, you are prompted to set two sets of credentials:
    a. **Administrator credentials** - Used for accessing the **Oxygen** license server administrative interface.
    b. **Standard user credentials** - Used by an **Oxygen** 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 error messages are redirected to the `errLicenseServer.log` file in `oXygen HTTP License Server\work\Windows Service` folder.

HTTPS Support

The All-Platform Distribution of the HTTP License Server do not provide the means to use HTTPS (TLS). If you need HTTPS, you should install the HTTP License Server WAR Distribution *(on page 62)* on Tomcat (preferably 9.x). See the WAR Distribution HTTPS Support *(on page 64)* section for more details.

Installing the 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 through 9.x is recommended (available at http://tomcat.apache.org). Tomcat 9.x is officially supported.

   Note:
   Tomcat 10.x and later will not work with the license server.

   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.

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** 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**.
    c. Another user with the role *manager-gui*, used for accessing the Apache Tomcat Web Application Manager. In the subsequent example, this user name is **tomcat**.

    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:

```
<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"/>
  <role rolename="manager-gui"/>
  <user username="John" password="user_pass" roles="user"/>
  <user username="Mary" password="admin_pass" roles="admin"/>
  <user username="tomcat" password="s3cret" roles="manager-gui"/>
</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 *manager-gui* role (*tomcat* 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 `manager-gui` role (*tomcat* 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.

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** 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 Developer 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`.

HTTPS Support

If you need HTTPS, you should install Tomcat (preferably 9.x) and follow its respective documentation regarding enabling and configuring TLS support.

> **Important:**
> **If you use HTTPS, you will need an SSL certificate that is issued by a trusted authority**. Self-signed certificates (or certificates issues by a local authority) are not accepted by the Oxygen XML Developer application (client side).

Installing Multiple Instances of WAR Distribution on a Tomcat Web Server

For organizations that have multiple sets of licenses (for example, an integrator with multiple clients might host a different license server for each client), use 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/`) 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).

Managing License Servers

This section includes information about managing the your license server.

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 displays the current status of the server and provides additional instructions for using the license server with **Oxygen XML Editor/Author/Developer**.

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

- **Current Allocated Licenses** - Opens the **Allocated License Report** page *(on page 65)*.
- **Usage Statistics** (Available only for floating licenses) - Opens the **License Usage Statistics** page *(on page 66)*.
- **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 68)*.
- **Configuration** - Opens a page where you can configure notification settings and set up the mail server used for sending emails whenever license requests from users are rejected. This page also contains a **Allow users to lock licenses** option. Enabling this option allows the users to lock/reserve a floating license *(on page 57)*.
- **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** 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`.

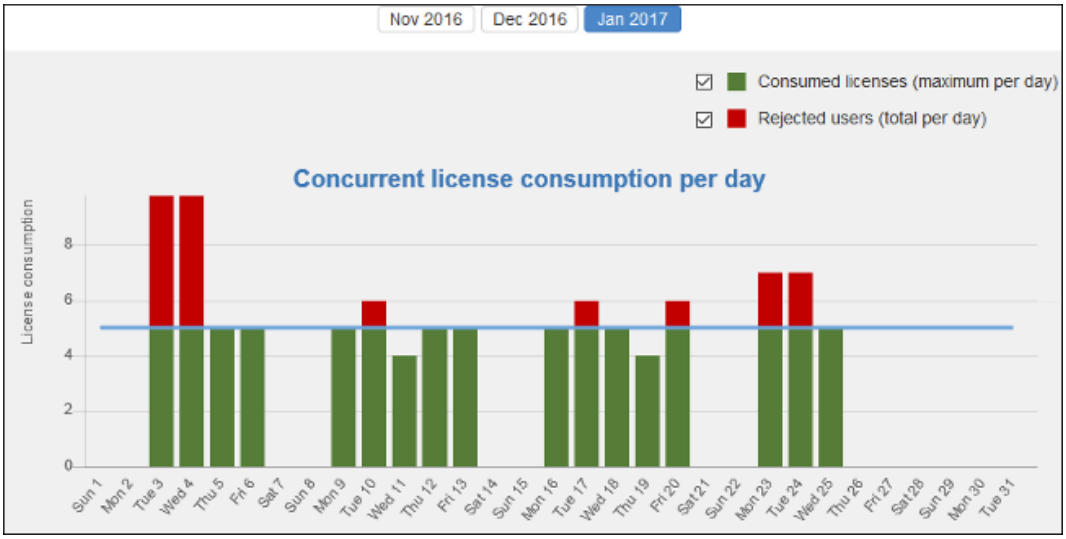
License Usage Statistics Page (Floating License Only)

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.

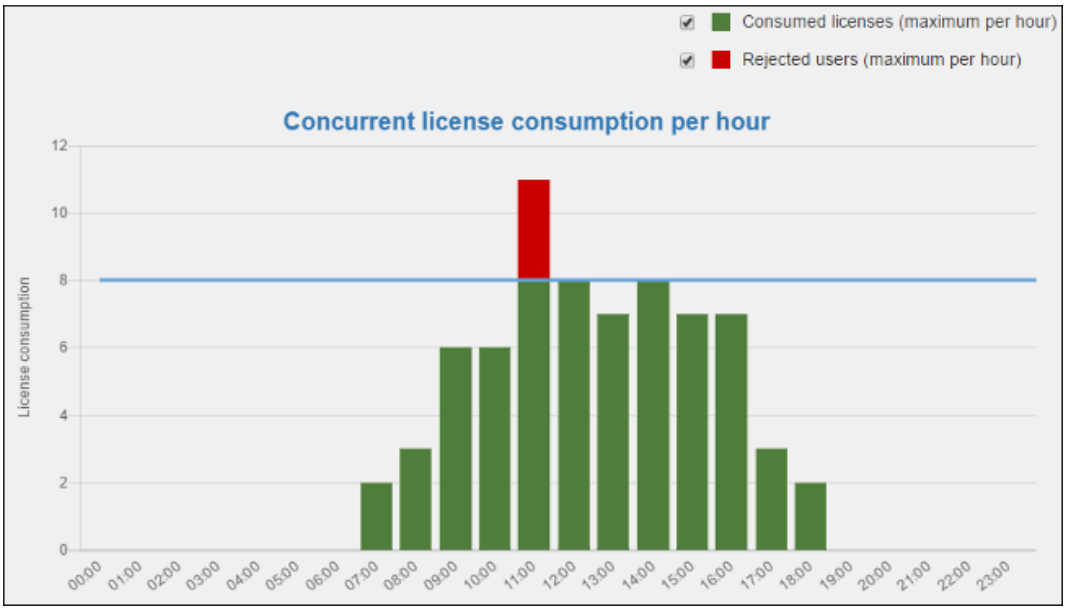
Tip:
You can click on any bar to see the license consumption per hour for that particular day.

Figure 6. Concurrent License Consumption per Day Chart



- **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.

Figure 7. Concurrent License Consumption per Hour Chart



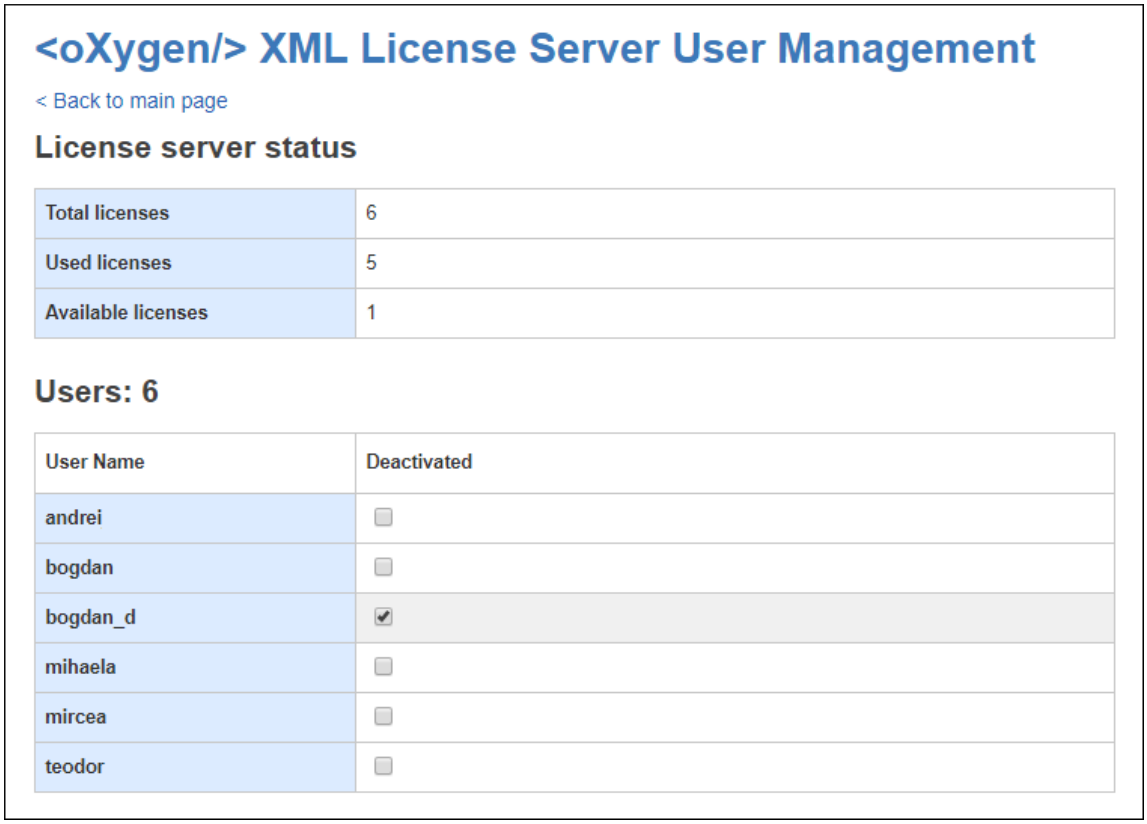
Users Management Page (Named-User License Only)

When a named-user license key is used, the license server allocates available licenses in the order they are requested until the maximum number is reached. Any additional users attempting to obtain 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.

Figure 8. Users List Management Page



Replacing or Removing a License Key in an HTTP License Server

The following procedure assumes that your HTTP license server contains a previously activated license key 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 50)*.

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** 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 60)*.

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 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** 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 Developer are released to provide enhancements that fix problems and add new features.

By default, Oxygen XML Developer automatically checks for new versions at startup (or every 24 hours from 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 a new patch of a new version is detected, the dialog box will be displayed only when the new patch version includes a critical bug fix or when the license allows upgrading to the new version.

To disable this check, open the **Preferences** dialog box **(Options > Preferences)** *(on page 79)*, 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 Developer on Windows/Linux

What is Preserved During an Upgrade?

When you install a new version of Oxygen XML Developer, some data is preserved and some is overwritten. If there is a previous version of Oxygen XML Developer 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 2305)* files, XSLT stylesheets, *XML Catalogs (on page 2309)*, 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 2305)* that were stored outside the installation directory (as configured in **Document type associations > Locations** *(on page 94)*) 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 2305)* files, XSLT stylesheets, *XML Catalogs (on page 2309)*, 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 2305)* that were stored outside the installation directory (as configured in **Document type associations > Locations** *(on page 94)*) will be preserved and will be found by the new installation.

How to Upgrade Oxygen XML Developer 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 Developer.
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 Developer on macOS

What is Preserved During an Upgrade?

When you install a new version of Oxygen XML Developer, 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 2305)* files, XSLT stylesheets, *XML Catalogs (on page 2309)*, 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 2305)* that were stored outside the installation directory (as configured in **Document type associations > Locations** *(on page 94)*) will be preserved and will be found by the new installation.

How to Upgrade Oxygen XML Developer on macOS

1. Uninstall the current version of Oxygen XML Developer *(on page 78)* or rename the installation directory.
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 Developer.
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 Developer provides an *add-on (on page 2307)* mechanism that can automatically discover and install *plugins (on page 1891)* 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 Developer 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.

   DANGER:
   Installing from update sites other than the official **Oxygen** sites may cause security risks. Please verify that all your installed third-party add-ons come from companies or individuals that you have verified and that you implicitly trust. As a way to help with identifying possible add-on security problems, Oxygen XML Developer issues warnings for unsigned plugins.

   Note:
   All add-ons are installed in the `extensions` directory inside the Oxygen XML Developer preferences directory *(on page 80)*.

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 Developer 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 Developer version.

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

After installing or upgrading to a new version of Oxygen XML Developer, 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 1891)*

Privacy Options

As an on-premise desktop application, Oxygen XML Developer does not store sensitive user data or user-specific files on remote servers. The full Privacy Policy is available on the official Oxygen XML Developer web site.

Whenever it is started, the application may connect to its official web site (**https://www.oxygenxml.com**) to provide notifications about new releases or new events. In the **Global** preferences page, there are some checkboxes that can be disabled if you do not want this type of information retrieved:

Automatic Version Checking

If this option is selected, the application obtains information about new available versions from the official Oxygen XML Developer web site. No specific information about the current installation is passed to the Oxygen XML Developer web site during this process.

Check for Oxygen-related events at startup

If this option is selected, the application obtains information about events that get displayed in the **Welcome** screen. The types of downloaded events include information about the addition of new videos on the website, announcements of upcoming webinars and conferences where the Oxygen XML Developer team will participate, and more. No specific information about the current installation is passed to the Oxygen XML Developer web site during this process.

Check for Oxygen-related notifications

If this option is selected, the application obtains information about other various notifications that get displayed on the right side of the status bar *(on page 285)*. No specific information about the current installation is passed to the Oxygen XML Developer web site during this process.

Providing installation-specific details to the Oxygen XML Developer technical support team is also possible by using the **Help > Report problem** action from the main menu. Also, if an *unhandled/fatal* error occurs in the application, the user is presented with a **Report problem** dialog box and can choose to submit the error and its context for analysis. The **Report problem** dialog box shows the exact specific details that are sent to the Oxygen XML Developer technical support team and can be examined by the end user before being sent. No files or confidential information are sent.

When the application is installed on Windows or Linux using the provided installation kit, the last installation step allows you to click a **Privacy Options** link to choose which privacy-related settings should be disabled before the application starts for the first time. The **Privacy Options** can also be disabled when performing an unattended install. See instructions here: Windows Unattended Installation *(on page 35)*.

Besides the settings already present in the **Global** preferences page, the **Privacy Options** dialog box contains two extra settings:

Allow installing add-ons

If this option is not selected, the end user is not allowed to install add-ons in the application.

Allow installing "AI Positron" add-on

If this option is not selected, the end user is not allowed to install the **AI Positron** add-on in the application.

Allow connections to the AI Positron service

If this option is not selected, the end user is not allowed to use the AI Positron service in the application.

Figure 9. Installation - Privacy Policy Link

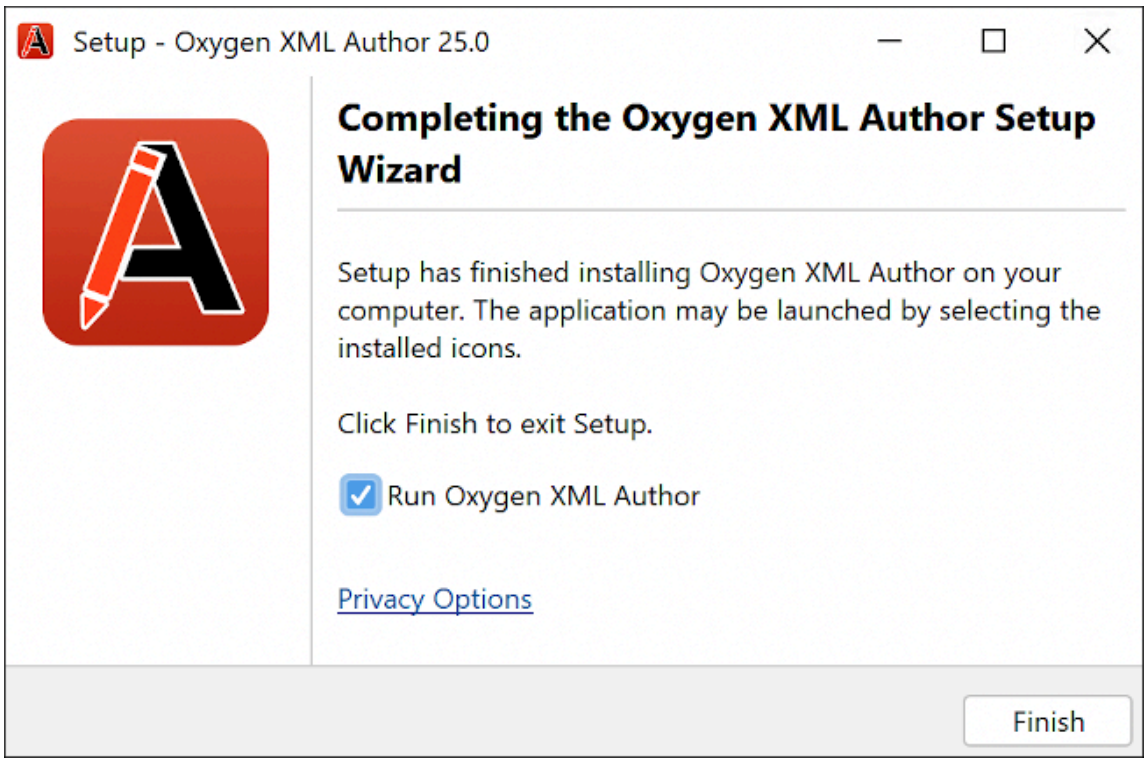
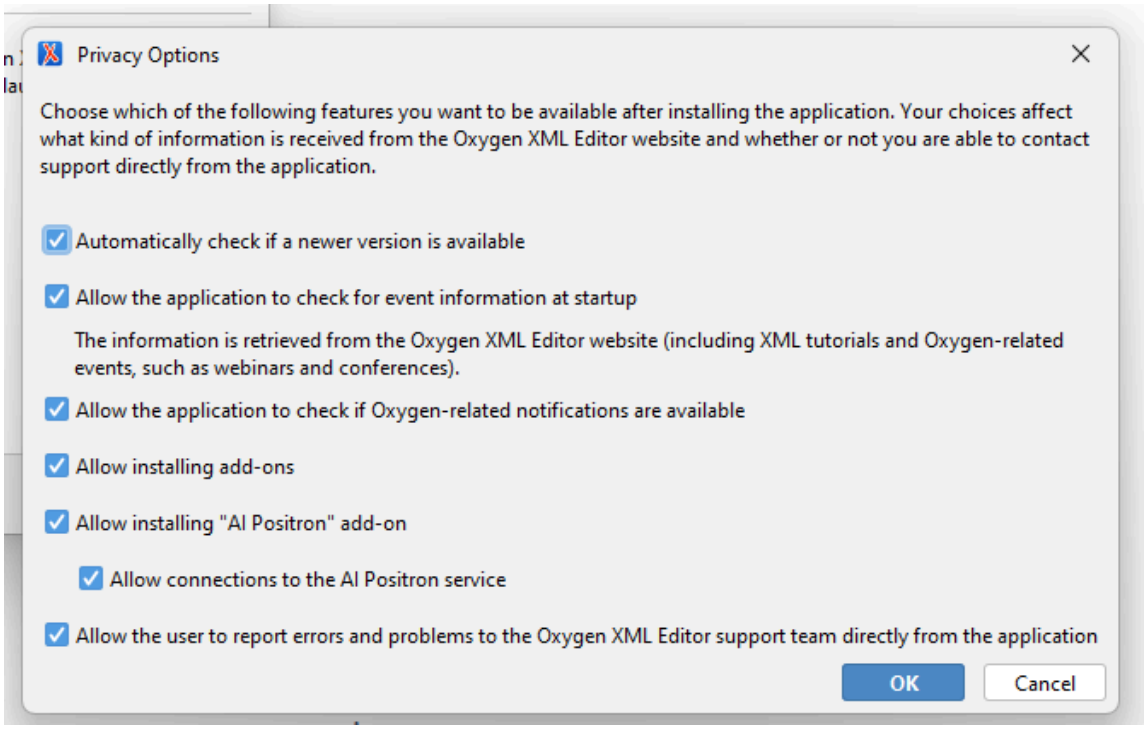


Figure 10. Installation - Privacy Options



Uninstalling

How to Uninstall Oxygen XML Developer



CAUTION:
The following procedure will remove Oxygen XML Developer 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 Developer 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%\com.oxygenxml.developer%APPDATA%\com.oxygenxml.jsoneditor`. Note that the `AppData` directory is hidden. If you cannot locate it, type `%APPDATA%` and press `ENTER` in the File Explorer address bar. (`%APPDATA%` expands to `[user-home-dir]\AppData\Roaming`).
   - **macOS** - Remove the directory: `Library/Preferences/com.oxygenxml.developerLibrary/Preferences/com.oxygenxml.jsoneditor` of the user home folder.
   - **On Linux**, remove the directory: `.com.oxygenxml.developer.com.oxygenxml.jsoneditor` 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 Developer

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 252)* that are available for customizing user-defined commands.

Preferences

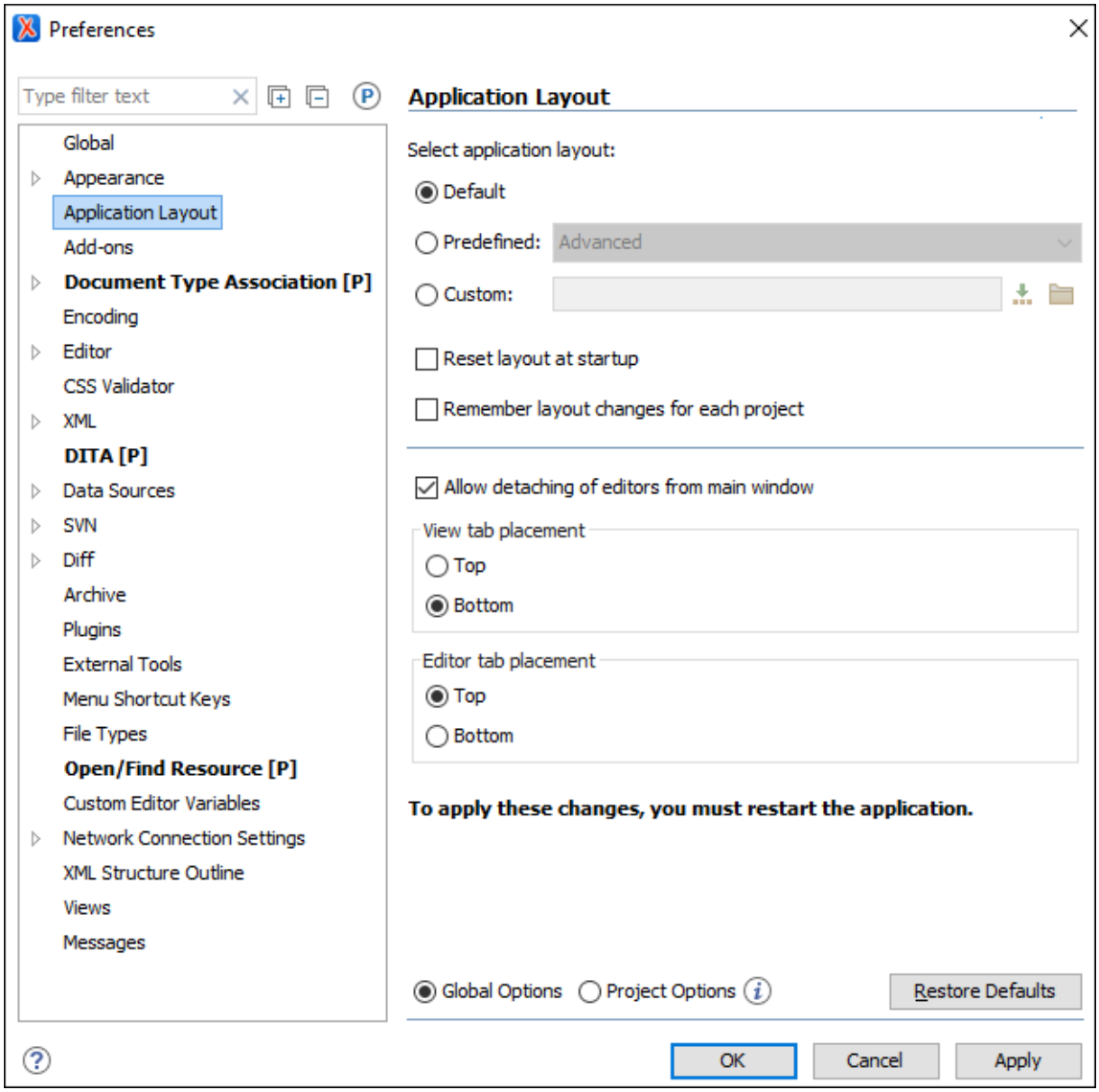
You can configure Oxygen XML Developer options using the **Preferences** dialog box.

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 241)*.

Figure 11. Preferences Dialog Box



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 241)* or **Project Options** *(on page 241)*.

Figure 12. Controlling the Storage of the Preferences



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:

- **Windows (7, 8, 10)** - `[user_home_directory]\AppData\Roaming\com.oxygenxml.developer`
- **macOS** - `[user_home_directory]/Library/Preferences/com.oxygenxml.developer`
- **Linux/Unix** - `[user_home_directory]/.com.oxygenxml.developer`

Global Preferences

The global options cover various aspects of the overall operation of Oxygen XML Developer. To configure the **Global** options, open the **Preferences** dialog box **(Options > Preferences)** *(on page 79)* and go to **Global**.

The following options are available in the **Global** preferences page:

Automatic Version Checking

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

Check for Oxygen-related events at startup

If this option is selected, Oxygen XML Developer will check for various new event updates on the Oxygen XML Developer 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 Developer 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 Developer 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 Developer 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 266)*. You can use this option to set the language of the user interface to a language that is not shipped with Oxygen XML Developer.



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 82)* 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 81)* 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 Developer 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 Developer 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 Developer 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 Developer 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 Developer automatically updates unmodified editors if the edited file changes externally.

Beep on operation finished

When this option is selected, Oxygen XML Developer 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 Developer 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 Developer uses too much memory. You can change the amount of memory available to Oxygen XML Developer by changing the parameters of the application launcher *(on page 267)*.

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 325)*. 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 Developer 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 (Windows and Linux only)

If this option is selected, Oxygen XML Developer 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 313)* 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, , main editing pane, or another location within the application).
- **Last visited directory** - The file browser opens the last visited folder.

Appearance Preferences

This preferences page contains various options that allow you to change the appearance of the user interface of Oxygen XML Developer. To configure the **Appearance** options, open the **Preferences** dialog box **(Options > Preferences)** *(on page 79)* 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 84)* is set to *Default* or *Windows*, Oxygen XML Developer 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 Developer. 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 85)* 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 84)* (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 85)*.

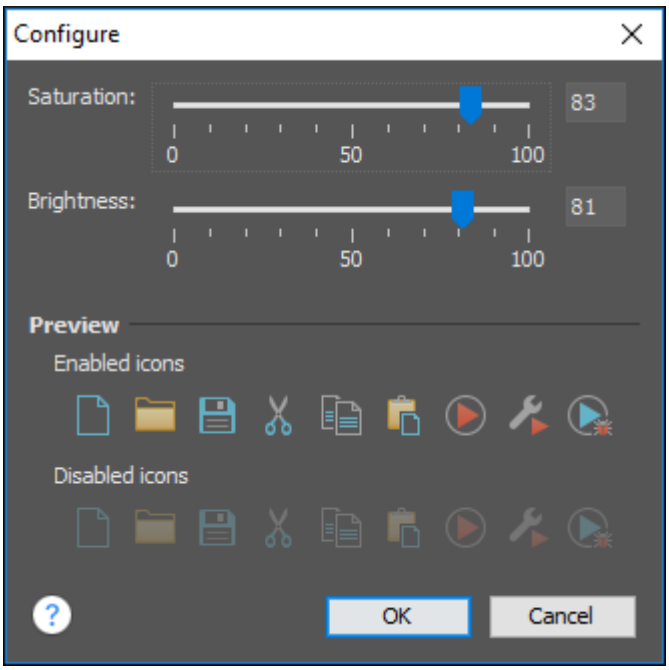
Export

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 Developer.

Configure icon saturation and brightness link

This link is available if you are using the **Graphite** theme *(on page 84)*. It opens a dialog box where you can configure the saturation and brightness for all the icons in Oxygen XML Developer.

Figure 13. Configure Icon Saturation and Brightness Dialog Box



Colors Preferences

Oxygen XML Developer 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 79)* 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 Developer allows you to choose the fonts to be used in the **Text**, **Design**, and **Grid** editor modes. To configure the font options, open the **Preferences** dialog box **(Options > Preferences)** *(on page 79)* 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.

Tip:
If you have recently installed fonts in Windows and they do not appear in the list of available fonts, you need to make sure you install the fonts via the **Install for all users** option instead of **Install for me**. For more details, see Fonts Installed in Windows Do Not Appear in Fonts Preferences Page *(on page 2221)*.

Schema default font

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

- The **Design** mode.
- 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. This same font influences the appearance of the content in the **Project** and **DITA Maps Manager** views. 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.

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

Related information
Changing the Font Size in the Editor *(on page 446)*

Application Layout Preferences

Oxygen XML Developer offers various *perspectives (on page 2307)* 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 79)* 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 2307)*. 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.
- **Basic** - Only the **Project** view *(on page 334)* and **Outline** view *(on page 465)* are visible. Recommended when you edit XML content and you need maximum screen space.
- **Schema development** - The **Project** *(on page 334)*, **Component Dependencies** *(on page 742)*, **Referenced/Dependent Resources** *(on page 739)*, **Outline** *(on page 735)*, **Palette** *(on page 691)*, and **Attributes** *(on page 737)* views are displayed.
- **XQuery development** - The **Project** *(on page 334)*, **Outline** *(on page 778)*, **XSLT/XQuery Input** *(on page 644)*, **XPath/XQuery Builder** *(on page 779)*, and **Transformation Scenarios** *(on page 1217)* views are displayed.
- **XSLT development** - The **Project** *(on page 334)*, **Component Dependencies** *(on page 649)*, **Referenced/Dependent Resources** *(on page*

646), **Outline** *(on page 639)*, **Attributes** *(on page 468)*, **Model** *(on page 470)*, **XSLT/XQuery Input** *(on page 644)*, **XPath/XQuery Builder** *(on page 779)*, and **Transformation Scenarios** *(on page 1217)* views are displayed.

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 Developer 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 Developer 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 Developer.

Resources

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

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

Add-ons Preferences

You can use *add-ons (on page 2307)* to enhance the functionality of Oxygen XML Developer. To configure the **Add-ons** options, open the **Preferences** dialog box **(Options > Preferences)** *(on page 79)* 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 Developer 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.

> **Attention:**
> To ensure safety, when using an add-on site, make sure it is reputable and trustworthy. Do not install add-ons that are unsigned or from sources that you do not fully trust.

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 79)* 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 526)* will be stored according to their storage location.

- If you associate a scenario that is stored at the project or framework 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 1101)* will be stored according to their storage location.

- If you associate a scenario that is stored at the project or framework 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).

Document Type Association Preferences

Oxygen XML Developer uses *document type associations (on page 2304)* to associate a document type *(on page 1042)* with a set of functionality provided by a *framework (on page 2305)*. To configure the **Document Type Association** options, open the **Preferences** dialog box **(Options > Preferences)** *(on page 79)* 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 2305)* (*document type associations (on page 2304)*), sorted by priority and alphabetically. Each edited document type has a set of association rules *(on page 97)* (used by the application to detect the proper document type association to use for an open XML document).

Disable all

Disables all document types listed in the table.

New

Opens a **Document type** configuration dialog box *(on page 95)* that allows you to add a new *framework*.

Edit

Opens a **Document type** configuration dialog box *(on page 95)* 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 95)* 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 95)* 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 Developer 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.

Locations Preferences

Oxygen XML Developer allows you to change the location where *frameworks (on page 2305)* (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 267)* or in the startup scripts *(on page 270)*.

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 *(on page 96)*.
- **Additional *framework* directories** - The document type configuration is loaded from one of the specified **Additional frameworks directories** list.
- **Add-ons** - An *add-on (on page 2307)* can contribute a *framework*. You can manage the add-ons locations in the Add-ons preferences page *(on page 92)*.
- **The `frameworks` folder** - The main folder containing *framework* configurations.

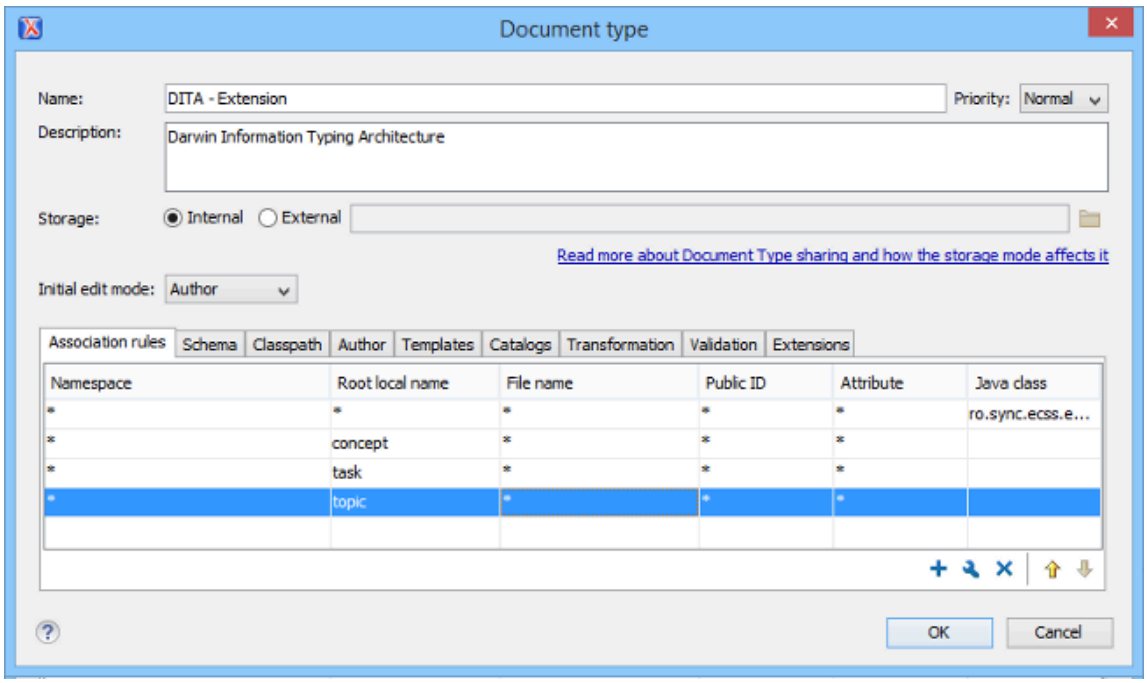
All loaded document type configurations are first sorted by priority, then by document type name and then by load location (in the exact order specified above). When an XML document is opened, the application chooses the first document type configuration from the sorted list that matches the specific 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 (on page 2305)* (document type). It is displayed when you use the **New**, **Edit**, **Duplicate**, or **Extend** buttons in the **Document Type Association** preferences page *(on page 93)* (open the **Preferences** dialog box **(Options > Preferences)** *(on page 79)* and go to **Document Type Association**).

Figure 14. 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 93)*.

Priority

Depending on the priority level, Oxygen XML Developer 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 93)*.

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 Developer internal options file.

Initial edit mode

Sets the default edit mode when you open a document for the first time: **Editor specific**, **Text**, **Grid** and **Design** (available 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 126)* 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 79)* 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 79)*, go to **Document Type Association**, use the **New**, **Edit**, **Duplicate**, or **Extend** button *(on page 93)*, 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.

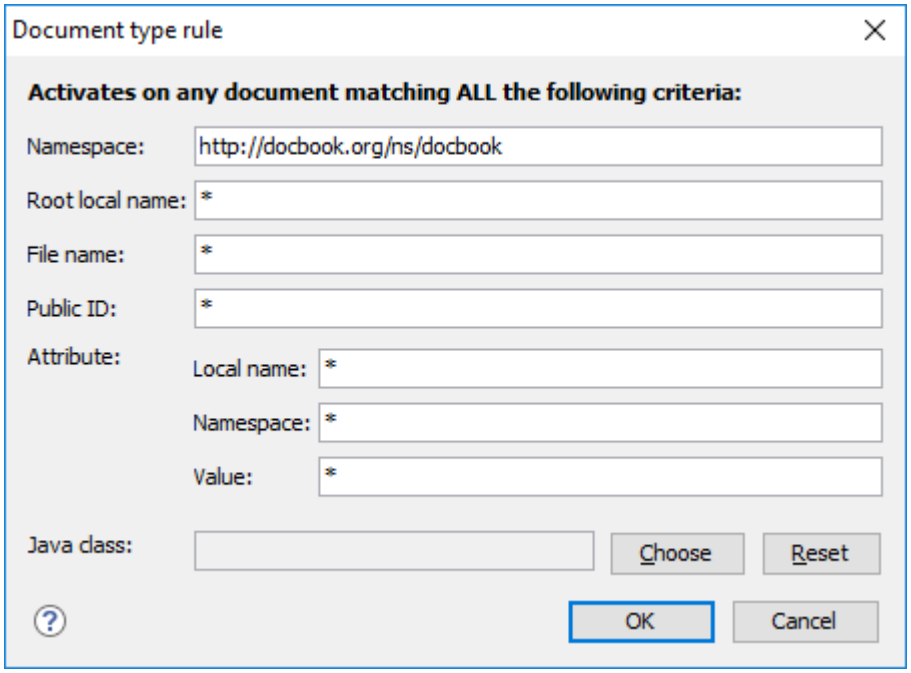
Move Down

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

By combining multiple association rules you can instruct Oxygen XML Developer to identify the type of a document. Oxygen XML Developer 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.

Figure 15. Document Type Rule Dialog Box



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 252)* 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 Developer to use if a 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 79)*, go to **Document Type Association**, use the **New**, **Edit**, **Duplicate**, or **Extend** button *(on page 93)*, 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 252)* 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 258)* 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 2306)* libraries that hold implementations for API extensions, implementations for custom **Author** mode operations, various resources (such as stylesheets), and *framework (on page 2305)* translation files. Oxygen XML Developer loads the resources looking in the folders in the order they appear in the list (from top to bottom).

To open the **Classpath** tab of the **Document type** configuration dialog box, open the **Preferences** dialog box **(Options > Preferences)** *(on page 79)*, go to **Document Type Association**, use the **New**, **Edit**, **Duplicate**, or **Extend** button *(on page 93)*, 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** *(on page 252)* 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** *(on page 252)* 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 1890)*

Use this option to specify the ID of a *plugin (on page 2307)*. The current *framework* has access to the classes loaded for the *plugin*.

Related information

Extensions Tab *(on page 122)*

Author Tab *(on page 100)*

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 2305)*-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 79)*, go to **Document Type Association**, use the **New**, **Edit**, **Duplicate**, or **Extend** button *(on page 93)*, 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 Developer 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 79)*, go to **Document Type Association**, use the **New**, **Edit**, **Duplicate**, or **Extend** button *(on page 93)*, 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 252)* button, or the browsing actions in the **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 2305)*. 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 79)*, 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 96)* 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 299)* to help you get started.

Note:
You can add or edit the action files outside of Oxygen XML Developer, but you will need to restart the application each time to reload the changes.

Open in editor ( )

For exported actions, there is a **Open in editor** action in the contextual menu that will open the 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 103)* 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 103)* where you can edit the selected action.

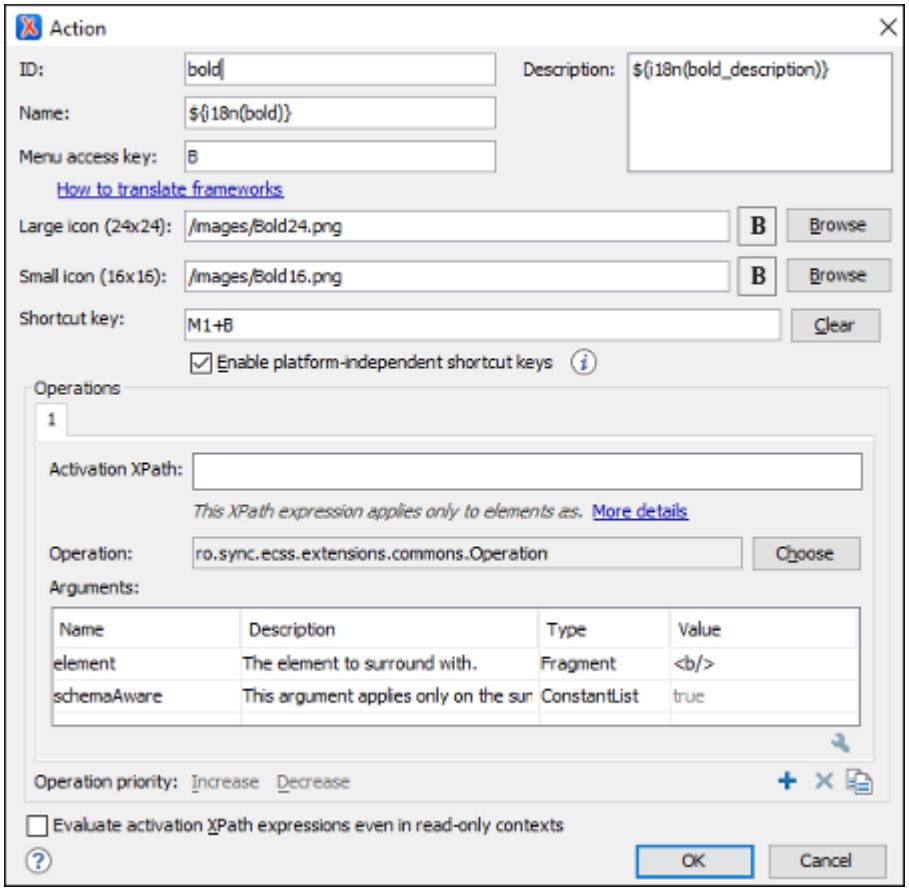
Delete an existing action ( )

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 79)*, go to **Document Type Association**, use the **New**, **Edit**, **Duplicate**, or **Extend** button *(on page 93)*, 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.

Figure 16. Action Dialog Box



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 **${i18n('key')}** editor variable *(on page 259)* 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 **${i18n('key')}** editor variable *(on page 259)* to allow for multiple translations of the description.

How to translate frameworks link

Use this link to see more information about localizing frameworks.

Large icon

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

> **Tip:**
> A good practice is to store the image files inside the *framework* directory and use the *${frameworks}* editor variable *(on page 259)* 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 Developer uses for the contextual menu action.

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 106)*.

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 106)*.

Operation

Specifies the invoked operation that can be a default operation or a custom operation.

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.

Controlling Which Author Operations Gets Executed Through XPath Expressions

An **Author** mode action can have multiple operation modes, each one invoking an Author operation 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 107)* - 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 109)* - Use this function to check whether or not an element is a valid global element for the current *framework (on page 2305)*, according to the associated schema.
- *oxy:current-selected-element() (on page 110)* - Use this function to get the currently selected element.
- *oxy:selected-elements() (on page 110)* - Use this function to get the selected elements.
- *oxy:is-required-element() (on page 111)* - 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 111)* - 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.

  ```
  oxy:allows-child-element("*:para")
  ```

  The above example verifies if the `<para>` element (of any namespace) is allowed in the current context.

- A prefix-qualified name of an element.

  ```
  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.

```
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.

```
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.

```
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.

```
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 2305)*, 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.

  ```
  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.

```
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.

  ```
  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.

  ```
  oxy:allows-global-element("*", "*:localname", " topic/topic ")
  ```
- A qualified name of an attribute.

  ```
  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 `<b>` elements that are children of the currently selected `<p>` 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 `<para>` element with the `@novice` 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 *xi:included* 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 Developer, 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 2305)*-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 79)*, go to **Document Type Association**, use the **New**, **Edit**, **Duplicate**, or **Extend** button *(on page 93)*, 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 Developer 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**

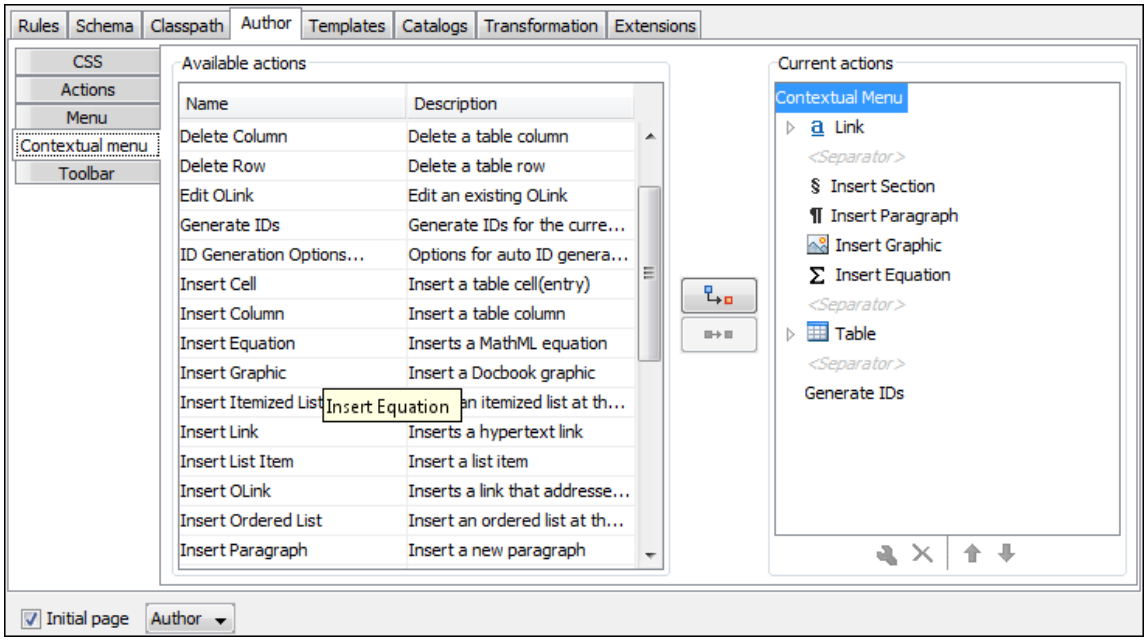
Moves an item down.

Contextual Menu Subtab

In the **Contextual menu** subtab you configure what *framework (on page 2305)*-specific action the *Content Completion Assistant (on page 2303)* 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 79)*, go to **Document Type Association**, use the **New**, **Edit**, **Duplicate**, or **Extend** button *(on page 93)*, click on the **Author** tab, and then the **Contextual Menu** subtab.

Figure 17. 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.

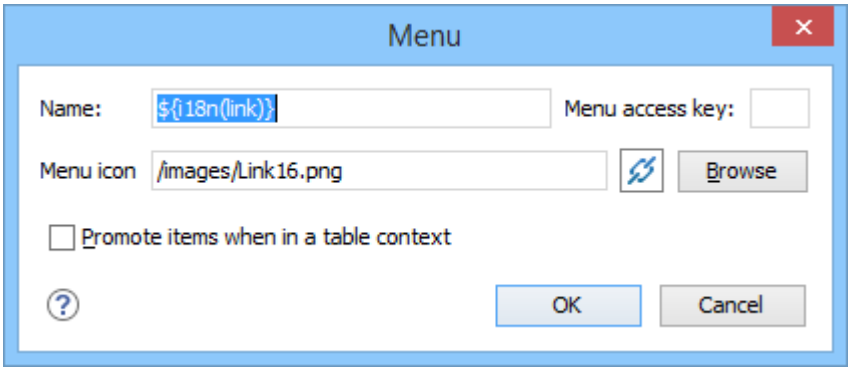
Add as child

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).

Figure 18. Menu Action Configuration Dialog Box



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 259)* 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 Developer 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 2305)*-specific action the Oxygen XML Developer 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 79)*, go to **Document Type Association**, use the **New**, **Edit**, **Duplicate**, or **Extend** button *(on page 93)*, 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 Developer 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 2305)*-specific the *Content Completion Assistant (on page 2303)* 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 79)*, go to **Document Type Association**, use the **New**, **Edit**, **Duplicate**, or **Extend** button *(on page 93)*, click on the **Author** tab, and then the **Content Completion** subtab.

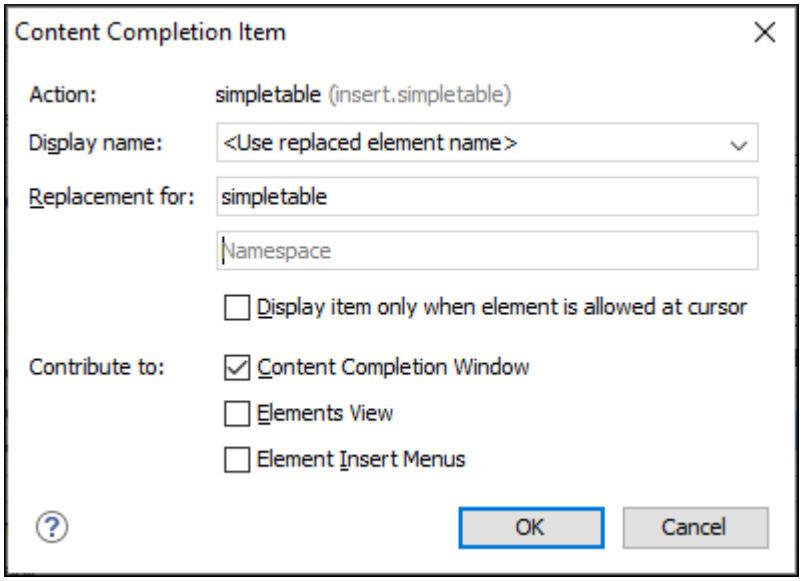
Available and Current Actions

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 19. 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 117)*.
- **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 103)* using the `oxy:allows-child-element()` XPath extension function. Activation XPaths for the action are still checked when the action is invoked.

Contribute to

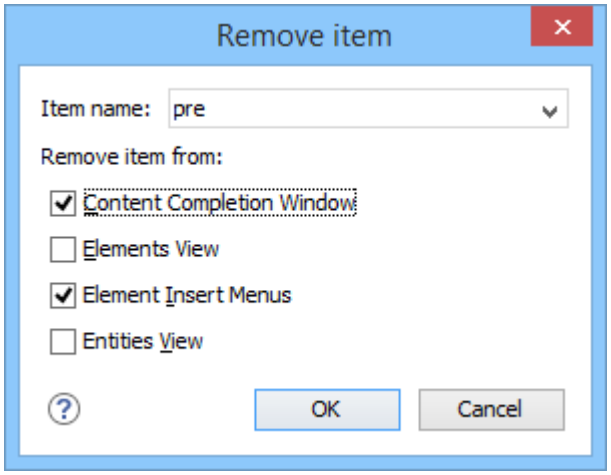
Use this section to specify where to display the configured item in the interface:

- **Content Completion Window** - The configured item will appear in the *Content Completion Assistant (on page 2303)*.
- **Elements View** - The configured item will appear in the **Elements** view.
- **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).

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 20. 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 2303)*.
- **Elements View** - The element will not appear in the **Elements** view.
- **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).
- **Entities View** - The element will not appear in the **Entities** view *(on page 472)*.

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 299)* 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 79)*, go to **Document Type Association**, use the **New**, **Edit**, **Duplicate**, or **Extend** button *(on page 93)*, 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 252)* 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.

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 252)* 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 307)*
Customizing Document Templates *(on page 309)*
Sharing Custom Document Templates *(on page 313)*

Catalogs Tab

The **Catalogs** tab specifies a list of *XML Catalogs*, specifically for the edited *framework (on page 2305)*, that are added to list of catalogs that Oxygen XML Developer 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 79)*, go to **Document Type Association**, use the **New**, **Edit**, **Duplicate**, or **Extend** button *(on page 93)*, 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 252)* 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 2305)* you are editing. These transformation scenarios are presented in the **Configure Transformation Scenarios** dialog box *(on page 1210)* 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 79)*, go to **Document Type Association**, use the **New**, **Edit**, **Duplicate**, or **Extend** button *(on page 93)*, 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 1210)*.

New

Opens the **New scenario** dialog box allowing you to create a new transformation scenario for the particular document type *(on page 1101)*.

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 1209)*.

Edit

Opens the **Edit scenario** dialog box allowing you to edit the properties of the currently selected transformation scenario *(on page 1207)*.

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 2305)* 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 350)* or Modular Contextual XML Editing Using 'Main Files' Support *(on page 567)*.

To open the **Validation** tab of the **Document type** configuration dialog box, open the **Preferences** dialog box **(Options > Preferences)** *(on page 79)*, go to **Document Type Association**, use the **New**, **Edit**, **Duplicate**, or **Extend** button *(on page 93)*, 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.

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.

Edit

Opens the **Edit scenario** dialog box allowing you to edit the properties of the currently selected validation scenario.

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 79)*, go to **Document Type Association**, use the **New**, **Edit**, **Duplicate**, or **Extend** button *(on page 93)*, and click on the **Extensions** tab.

Libraries containing the implementations must be present in the *classpath (on page 99)* of your document type. The Javadoc available at https://www.oxygenxml.com/InstData/Editor/SDK/javadoc/ contains details about how each API implementation functions.

Document Templates Preferences

Oxygen XML Developer 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 299)* when you create a new document (**New** toolbar button or **File > New**).

You can also create your own templates *(on page 307)* and share them with others. You can store your custom document templates in the existing `templates` folder in the Oxygen XML Developer 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 Developer 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 79)* 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 Developer 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 299)* 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, but if you customize the template *(on page 309)*, you need to set the `type` property to **dita** in the corresponding properties file.

Encoding Preferences

Oxygen XML Developer 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 79)* and go to **Encoding**.

The following encoding options are available:

Fallback character encoding

Specifies 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 Developer 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 Developer 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 `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 495)*, *Base32 (on page 496)*, or *Hex* conversions *(on page 497)*. The default setting is *UTF8*.

Encode non-ASCII characters in URL paths

If selected (default), Oxygen XML Developer 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 Developer 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 79)* 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 2303)*.

Completion proposal foreground

Allows you to set the color of the text in the *Content Completion Assistant (on page 2303)*.

Documentation window background

Allows you to set the background color of the documentation of elements suggested by the *Content Completion Assistant (on page 2303)*.

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 2303)*.

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 Developer 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

Makes the space character visible in the editor.

Can edit read-only files

If this option is selected, Oxygen XML Developer 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 2308)* icon ( ) and *Quick Fix (on page 2308)* 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**, **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 Developer lets you configure which edit mode 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 79)* and go to **Editor > Edit Modes** .

Allow Document Type specific edit mode setting to override the general mode setting

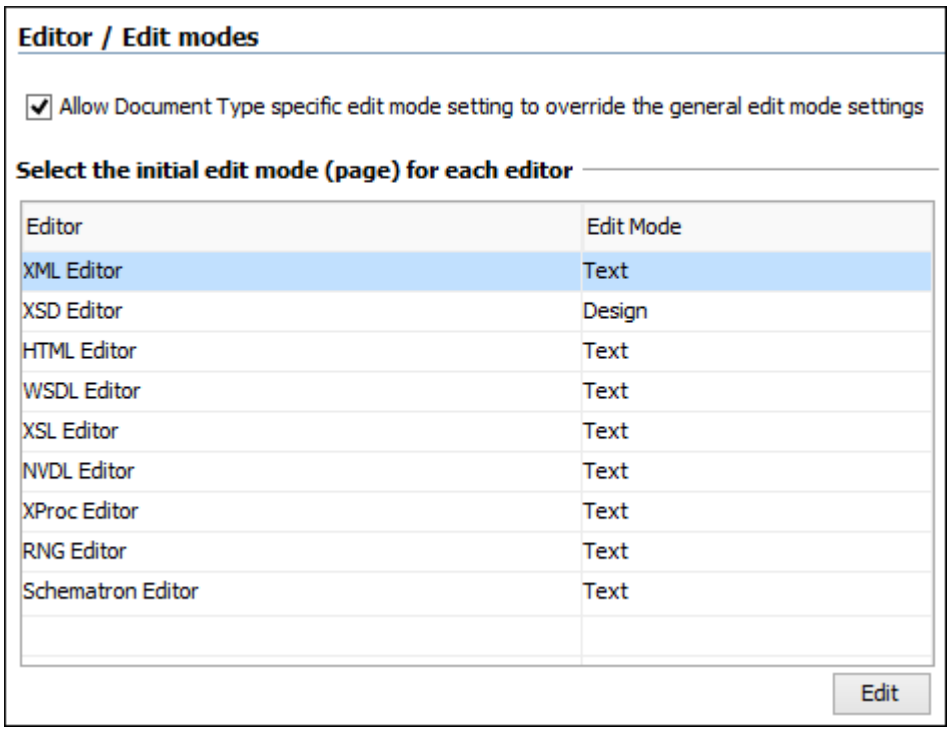
If selected, the initial edit mode setting set in the **Document Type** configuration dialog box *(on page 95)* overrides the general edit mode setting from the table below.

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**
- **Grid**
- **Design** (available only for the XSD editor).

Figure 21. Edit Modes Preferences Page



Text Preferences

Oxygen XML Developer allows you to configure how the **Text** mode editor appears. To configure these options, open the **Preferences** dialog box **(Options > Preferences)** *(on page 79)* 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, and some external tool editors (**Large File Viewer** *(on page 2007)*, **Compare Files** *(on page 402)*, **Compare Directories** *(on page 421)*).

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 1827)* of the debugger *perspectives (on page 2307)*. 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 is displayed in **Text** mode.

XML section

Highlight matching tag

If selected, when you place the cursor on a start or end tag, Oxygen XML Developer 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.

JSON section

Show property name after ending bracket

If enabled (default), property names are displayed after the ending bracket when editing JSON documents in **Text** mode. You can also use the transparency slider to adjust the level of transparency or opaqueness.

YAML section

Show SPACE marks for YAML only

If enabled (default), space characters are visible in the YAML editor.

Diagram Preferences

For certain XML languages, Oxygen XML Developer 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 79)* 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 280)*.



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 468)*, **Model** *(on page 470)*, **Elements** *(on page 472)*, **Outline** *(on page 831)*).

Show Relax NG diagram

Displays the Relax NG schema diagram in the split-screen views (**Full Model View** *(on page 825)* and **Logical Model View** *(on page 826)*).

Enable NVDL diagram and related views

Enables the NVDL schema diagram and synchronization with the related views (**Attributes** *(on page 468)*, **Model** *(on page 470)*, **Elements** *(on page 472)*, **Outline** *(on page 847)*).

Show NVDL diagram

Displays the NVDL schema diagram in the split-screen views (**Full Model View** *(on page 843)* and **Logical Model View** *(on page 844)*).

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 131)* 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 132)* where you can adjust the default zoom level of schema diagrams.

Grid Preferences

Oxygen XML Developer provides a **Grid** view *(on page 279)* of an XML document. To configure the **Grid** mode options, open the **Preferences** dialog box **(Options > Preferences)** *(on page 79)* 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 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 2303)* 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 2303)*.

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.

Schema Design Preferences

Oxygen XML Developer provides a graphical schema design editor *(on page 280)* to make editing XML Schema easier. To configure the **Schema Design** options, open the **Preferences** dialog box **(Options > Preferences)** *(on page 79)* 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 Developer 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 Developer 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 Developer 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 Developer lets you control which properties to display for XML Schema components in the XML Schema **Design** view *(on page 280)*. To configure the schema design properties displayed, open the **Preferences** dialog box **(Options > Preferences)** *(on page 79)* 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 Developer 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 79)* 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 Developer lets you control how files are opened. To configure the options for opening documents, open the **Preferences** dialog box **(Options > Preferences)** *(on page 79)* 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 Developer 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 Developer, it will *lock* all the currently open files. If you deselect this option with files already opened, 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:
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 Developer 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 490)*.
- **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 490)*.
- **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 489)*.

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 134)* 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 397)* for memory-intensive operations.

Show warning when loading large documents

Oxygen XML Developer 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 399)*.

Show warning when loading documents with long lines

When selected, Oxygen XML Developerwill 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 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 399)*.

Save Preferences

Oxygen XML Developer lets you control how files are saved. To configure the options for saving documents, open the **Preferences** dialog box **(Options > Preferences)** *(on page 79)* 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 301)* will not be available in the **New Document** wizard *(on page 299)*, 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 Developer 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.

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 Developer 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.

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 1070)*.

To configure the **Format** options, open the **Preferences** dialog box **(Options > Preferences)** *(on page 79)* and go to **Editor > Format**.

The following options are available:

Detect indent on open

If selected, Oxygen XML Developer detects how a document is indented when it is opened. Oxygen XML Developer 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 Developer 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.

Tip:
If you want to minimize the formatting differences created by the **Format and Indent** operation in a document edited in the **Text** edited mode, make sure that both the **Detect indent on open** and **Detect line width on open** *(on page 138)* options are selected.

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 136)* option.

Indent size

The meaning of this setting depends on the following:

- If the **Detect indent on open** option *(on page 136)* 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 Developer uses the number of detected SPACE characters.
- If the **Indent with tabs** option *(on page 137)* 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 485)*.

For information about when this setting is used, see Where Indent Size and Line Width Settings are Used in Oxygen XML Developer *(on page 138)*.

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 Developer.



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 Developer 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.



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 138)*.

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 Developer will be used for storing the undo states. If this option is selected, Oxygen XML Developer 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 Developer

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.

Resources

For more information about the formatting options offered by Oxygen XML Developer, 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 79)* 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 139)* 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:

```
<root>
  text <parent> <child></child> </parent>
</root>
```

Indent inline elements selected:

```
<root> text <parent>
    <child/>
  </parent>
</root>
```

Indent inline elements not selected:

```
<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>`).

> **Note:**
> When using the **Format and Indent** operation in **Text** mode, if the **Schema-aware format and indent** option *(on page 141)* is enabled, Oxygen XML Developer will use information from the associated schema and avoid expanding tags for elements that are defined as *empty* in the schema.

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 136)*.

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 140)*) 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 Developer formats and indents XML documents, a whitespace normalization process is applied, thus replacing whitespace sequences with single space characters. Oxygen XML Developer 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 79)* 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.

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 79)* 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 79)* 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 Developer can format and indent your CSS files. To configure the **CSS** formatting options, open the **Preferences** dialog box **(Options > Preferences)** *(on page 79)* 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 79)* 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 Developer to format embedded JavaScript code, taking precedence over the Schema-aware format and indent *(on page 141)* option. This option is selected by default.

JSON Preferences

To configure the **JSON** format options, open the **Preferences** dialog box **(Options > Preferences)** *(on page 79)* and go to **Editor > Format > JSON**.

The following option is available:

Sort keys - If enabled, when using the **Format and Indent** action on JSON content, dictionary items are sorted alphabetically by key name.

Content Completion Preferences

Oxygen XML Developer provides a *Content Completion Assistant (on page 2303)* 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 79)* 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 Developer 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 Developer 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 Developer 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 Developer inserts the required elements specified in the DTD, XML Schema, or RELAX NG schema that is associated with the edited XML document.
  - **Add optional content** - If selected, Oxygen XML Developer inserts the optional elements specified in the DTD, XML Schema, or RELAX NG schema.
  - **Add first Choice particle** - If selected, Oxygen XML Developer 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 Developer 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 145)* is not selected.

Show all entities

Oxygen XML Developer 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

If selected, Oxygen XML Developer automatically inserts the required attributes taken from the DTD or XML Schema for an element inserted with the *Content Completion Assistant*. For ID attributes that are required, a unique value is automatically generated for each new ID.

Insert the fixed attributes

If selected, Oxygen XML Developer automatically inserts any `FIXED` attributes from the DTD or XML Schema for an element inserted with the *Content Completion Assistant*.

Show recently used items

When selected, Oxygen XML Developer 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 Developer learns the attribute values used in a document.

Learn on open document

Oxygen XML Developer automatically learns the document structure when the document is opened.

Learn words (Dynamic Abbreviations, available on Ctrl+Space

When selected, Oxygen XML Developer learns the typed words and makes them available in a content completion fashion by pressing **Ctrl + Space** 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.

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 Developer can suggest elements from these vocabularies. To configure the XSLT content completion options, open the **Preferences** dialog box **(Options > Preferences)** *(on page 79)* 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 Developer analyzes the namespaces declared in the root element to find an appropriate schema.

If the detection fails, Oxygen XML Developer uses one of the following options:

- **None** - The *Content Completion Assistant (on page 2303)* suggests only XSLT elements.
- **HTML** - The *Content Completion Assistant (on page 2303)* includes HTML elements, including HTML5 elements (such as `<video>`, `<canvas>`, etc.).
- **Formatting objects** - The *Content Completion Assistant (on page 2303)* 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 Developer 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 79)* 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 634)* that you enter in the `@match`, `@select`, and `@test` XSL attributes and also in the XPath toolbar *(on page 1727)*.
  - **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 638)* 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 Developer provides content completion assistance when you are writing XML Schema (XSD). To configure XSD preferences, open the **Preferences** dialog box **(Options > Preferences)** *(on page 79)* 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 2303)* 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 Developer 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 79)* 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 Developer 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 Developer can also use JavaScript libraries when making suggestions. List the paths (URIs) of any JavaScript files you want Oxygen XML Developer to use when making suggestions.

Note:
The paths can contain editor variables *(on page 252)* such as `${pdu}`, or `${oxygenHome}`. You can make these paths relative to the project directory or installation directory.

JSON Preferences

Oxygen XML Developer 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 79)* 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.

YAML Preferences

Oxygen XML Developer can provide content completion suggestions when you are editing YAML files. To configure content completion support for YAML, open the **Preferences** dialog box **(Options > Preferences)** *(on page 79)* and go to **Editor > Content Completion > YAML**. You can configure the following options:

Generate required content

When invoking content completion over YAML files, all contextual required content is automatically generated according to the specifications from the associated JSON schema.

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, optional properties that are defined in the associated JSON schema will be added when using content completion in YAML 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 YAML 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 Developer can display these annotations when offering content completion suggestions. To configure the **Annotations** preferences, open the **Preferences** dialog box **(Options > Preferences)** *(on page 79)* and go to **Editor > Content Completion > Annotations**.

The following options are available:

Show annotations in Content Completion Assistant

If selected, Oxygen XML Developer displays the schema annotations of an element, attribute, or attribute value currently selected in the *Content Completion Assistant (on page 2303)* proposals list.

Show annotations in tooltip

If selected, Oxygen XML Developer 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. 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 Developer 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 Developer 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`, is considered annotation and is displayed in the annotation window next to the *Content Completion Assistant (on page 2303)* window and in the **Model** view *(on page 470)*. When this option is not selected, only elements from the Relax NG annotations namespace, that is `http://relaxng.org/ns/compatibility/annotations/1.0` are considered annotations.

Related information

Schema Annotations in Text Mode *(on page 460)*

Code Templates Preferences

Code templates are code fragments that can be inserted at the current editing position. Oxygen XML Developer 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 462)* for any type of file and share them with your colleagues *(on page 464)* using the template export and import functions.

To configure **Code Templates**, open the **Preferences** dialog box **(Options > Preferences)** *(on page 79)* 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 2303)*. 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 252)* 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 2303)*. 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 252)* 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

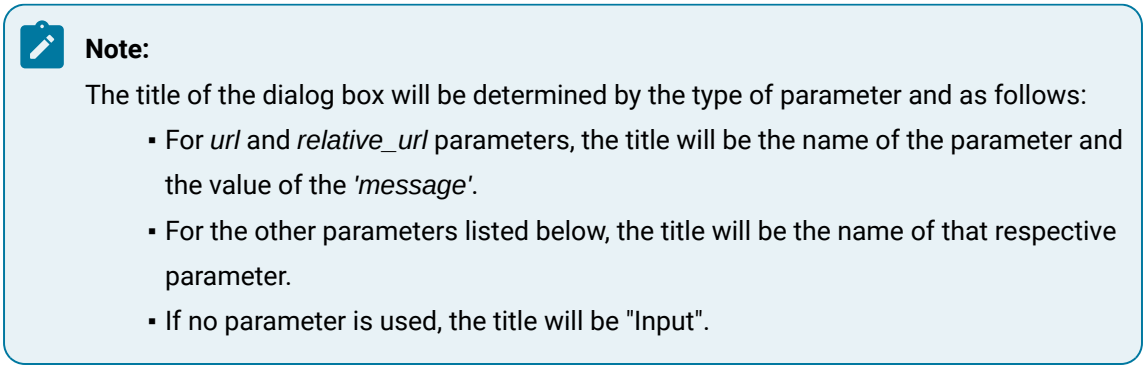
Exports a file with code templates.

Import

Imports a file with code templates that was created by the **Export** action.

You can use the following editor variables *(on page 252)* when you define a code template in the **Content** text box:

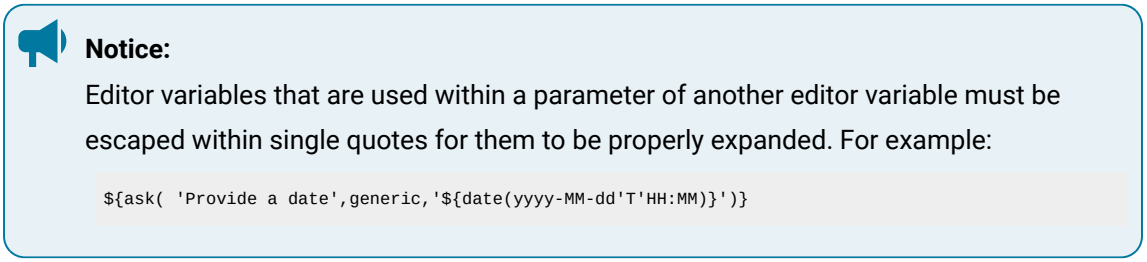
- **${caret}** - The position where the cursor is located. This variable can be used in a code template, in **Author** mode operations, or in a **selection** *plugin*.
- **${selection}** - The currently selected text content in the currently edited document. This variable can be used in a code template, 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".



Notice:

Editor variables that are used within a parameter of another editor variable must be escaped within single quotes for them to be properly expanded. For example:

```
${ask( 'Provide a date',generic,'${date(yyyy-MM-dd'T'HH:MM)}')}
```

Parameter	
generic (default)	**Format**: *${ask('message', generic, 'default')}*
	Description: The input is considered to be generic text that requires no special handling.
	Example: ▪ *${ask('Hello world!')}* - The dialog box has a *Hello world!* message displayed. ▪ *${ask('Hello world!', generic, 'Hello again!')}* - The dialog box has a *Hello world!* message displayed and the value displayed in the input box is *'Hello again!'*.
url	**Format:** *${ask('message', url, 'default_value')}*

Parameter	
	Description: Input is considered a URL. Oxygen XML Developer checks that the provided URL is valid.
	Example: ▪ *${ask('Input URL', url)}* - The displayed dialog box has the name *Input URL*. The expected input type is URL. ▪ *${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*.
relative_url	**Format:** *${ask('message', relative_url, 'default')}*
	Description: Input is considered a URL. This parameter provides a file chooser, along with a text field. Oxygen XML Developer tries to make the URL relative to that of the document you are editing.  **Note:** If the *$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 Developer 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')}*

Parameter	
	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 *${xpath_eval()}*. **Note:** The *'default'* parameter specifies the default-selected value and can match either a key or a value.
	Example: ▪ *${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.

Parameter	
	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', 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 || '":"' || $pair?2 || '";")}), 'ios')}*
radio	**Format:** *${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:**

Parameter	
	▪ *${ask('Operating System', radio, ('win':'Microsoft Windows';'macos':'macOS';'lnx':'Linux/UNIX'), 'macos')}* - The dialog box has the name *'Operating System'*. The radio button group allows you to choose between the three operating systems. **Note:** In this example, `macOS` is the default-selected value and if selected, it would return *macos* for the output. ▪ *${ask('Operating System', radio, (${xpath_eval(for $pair in (['win', 'Microsoft Windows'], ['macos', 'macOS'], ['lnx', 'Linux/UNIX']) return '"' || $pair?1 || '":"' || $pair?2 || '";')}), 'ios')}*
textarea	**Format:** *${ask('message', textarea, 'default')}*
	Description: The input is a text area that requires no special handling.
	Example: ▪ *${ask('Hello world!', textarea, 'Hello again!')}* - The dialog box would have a *Hello world!* message displayed and the value of *Hello again!* would be displayed in the text area.

Here is an example of a complex XPath expression for constructing a radio button set:

```
${ask('Bundle for keydefs', radio, (${xpath_eval(let $family := 'utah', $bundles := doc(environment-variable('HOME')
 || '/git/dita-author-tools/resource/properties/bundle-data.xml')/*/family[@name eq $family]/*, $pairs :=
 $bundles ! concat('''',./@bundle-code, ''','':',  ''',./@map-filename,''')  return string-join($pairs, ';'))}),
 @bundle-code)}
```

- **${timeStamp}** - The timestamp, which is the current time in Unix format. For example, it can be used to save transformation results in multiple output files on each transformation.
- **${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.
- **${cfd}** - Current file folder as file path, that is the path of the currently edited document up to the name of the parent folder.
- **${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 Developer 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 462)*

Syntax Highlight Preferences

Oxygen XML Developer supports syntax highlighting in the **Text** mode editors for numerous types of documents, including XML, XHTML, JavaScript, XQuery, XPath, PHP, PowerShell, CSS, LESS, Markdown, Text, DTD, RNC, Java, JSON, Ant, and more.

To configure syntax highlighting, open the **Preferences** dialog box **(Options > Preferences)** *(on page 79)* 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 Developer allows you to specify syntax highlighting colors for elements and attributes with specific namespace prefixes. To configure the **Elements/Attributes by Prefix** preferences, open the **Preferences** dialog box **(Options > Preferences)** *(on page 79)* 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 492)* feature activated. To configure these options, open the **Preferences** dialog box **(Options > Preferences)** *(on page 79)* 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 492)* feature in XML files.
- **XSLT files** - Activates the Highlight Component Occurrences *(on page 972)* feature in XSLT files.
- **XML Schema files** - Activates the Highlight Component Occurrences *(on page 972)* feature in XSD files.
- **WSDL files** - Activates the Highlight Component Occurrences *(on page 972)* feature in WSDL files.
- **RNG files** - Activates the highlight component occurrences feature in RNG files.

- **Schematron files** - Activates the Highlight Component Occurrences *(on page 972)* feature in Schematron files.
- **Ant files** - Activates the Highlight Component Occurrences *(on page 686)* 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 79)* 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 and manual validation.

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 Developer.

Delay after the last key event (s)

The period of keyboard inactivity before starting a new validation (in seconds).

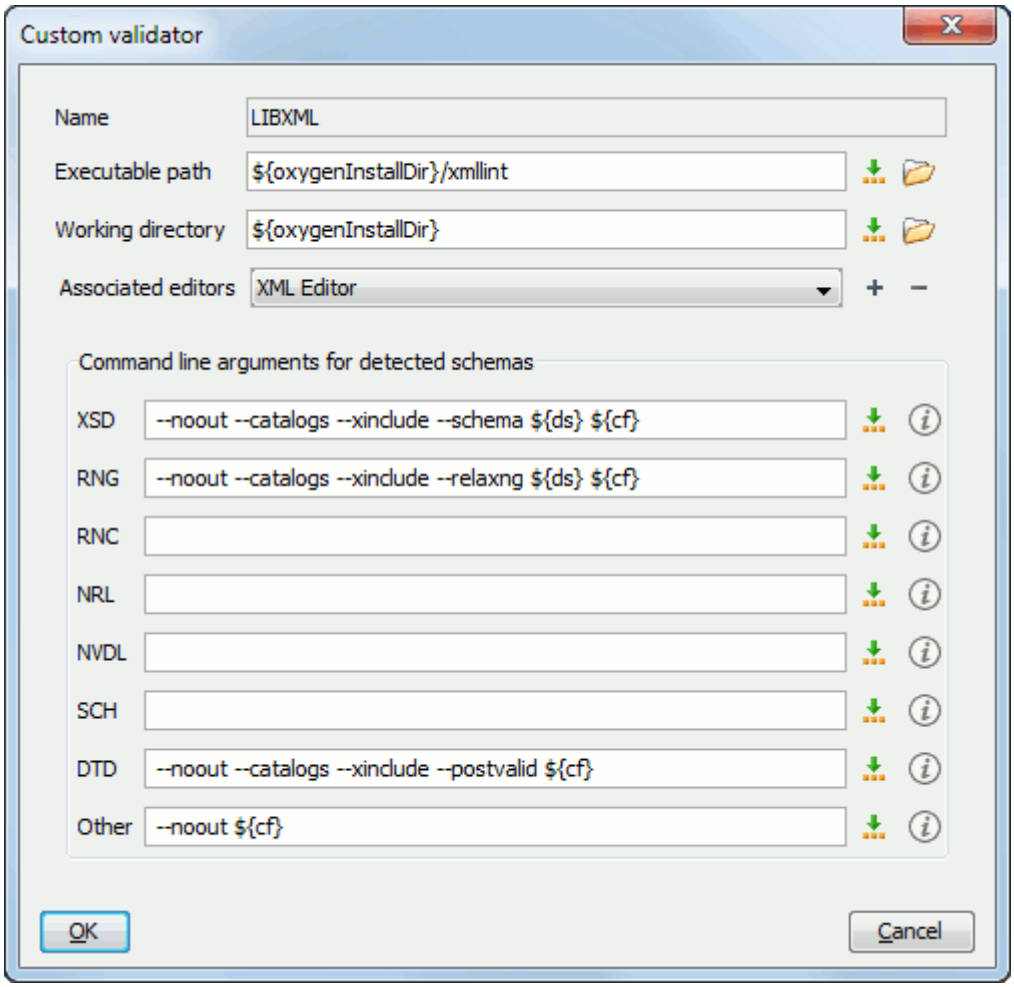
At the bottom of the preferences page you can choose whether or not the saved options will be shared with other users by selecting **Global** or **Project** storage options *(on page 240)*.

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 79)* 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.

Figure 22. Custom Validator Dialog Box



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 252)* 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 252)* 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 252)*:

- **${cf}** - Current file as file path, that is the absolute file path of the currently edited document.
- **${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.

Related information
Editor Variables *(on page 252)*

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 Developer *(on page 267)*.

- Increase the value of the **-Xss** parameter.

> **Note:**
> Increasing the value of the **-Xss** parameter affects all the threads of the application.

Related information

Setting a Java Virtual Machine Parameter when Launching Oxygen XML Developer *(on page 267)*

Ignored Validation Problems Preferences

Some validation issues include a Quick Fix *(on page 551)* 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 79)* 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 549)*, 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 549)*. 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 241)* so that you can easily share your ignored problems configuration with others.

Spell Check Preferences

Oxygen XML Developer provides support for spell checking in the **Text** mode. To configure the **Spell Check** options, open the **Preferences** dialog box **(Options > Preferences)** *(on page 79)* 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 Developer 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 380)*.

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 164)* 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.



Note:
When such problems are reported, they cannot be learned and ignored by the application as words stored in dictionaries, term lists, and the list of learned words are not handled as case-sensitive.

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*.

Spell Check Dictionaries Preferences

To set the Dictionaries preferences, open the **Preferences** dialog box **(Options > Preferences)** *(on page 79)* 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 Developer uses and to choose where to save new learned words.

The following options are valid when Oxygen XML Developer 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 Developer 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 Developer User Guide that explains how to add more dictionaries and term lists *(on page 383)*.

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 383)*
Adding Custom Spell Check Term Lists *(on page 385)*

Print Preferences

Oxygen XML Developer lets you configure how files are printed out of the editor. Note that these settings cover how files are printed directly from Oxygen XML Developer 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 79)* 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 279)* editor. These settings do not apply to the **Grid** *(on page 279)* and schema **Design** *(on page 280)* 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 79)* and go to **CSS Validator**.

You can configure the following options for the built-in **CSS Validator** of Oxygen XML Developer:

- **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 and the CSS extensions specific for **Oxygen**. That means all **Oxygen**-specific extensions are accepted in a CSS stylesheet by the built-in CSS validator *(on page 817)* 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 Developer recognizes browser-specific CSS properties (no validation is performed). The *Content Completion Assistant (on page 2303)* 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 2309)*, open the **Preferences** dialog box **(Options > Preferences)** *(on page 79)* 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 Developer 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 Developer 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 Developer Determines which Catalog to Use *(on page 565)*.

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 252)* 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 252)* 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 517)* so that the changes take full effect.

You can also add or configure catalogs at *framework* level from the **Catalogs** tab *(on page 119)* in the **Document Type** configuration dialog box *(on page 95)*.

Related information

Controlling the Catalog Resolver
Working with XML Catalogs *(on page 564)*

XML Parser Preferences

To configure the **XML Parser** options, open the **Preferences** dialog box **(Options > Preferences)** *(on page 79)* 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.

Enable secure processing

Enables the configuration of the number of entities that are allowed to be expanded and the maximum occurrence limit for nodes.

Entity expansion limit

The maximum number of entities that are allowed to be expanded by the parser.

Nodes maximum occurrence limit

The maximum occurrence limit for XML schema nodes.

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 Developer is turned on for validation 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 79)* 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 299)*.

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 79)* 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.



Note:
Selecting this option disables the **Check ID/IDREF** option.

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 79)* 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 2308)* in Schematron files. This option is selected by default.

Restrict "Apply All" to quick fixes marked as "sqf:default-fix"

Allows you to restrict the *Schematron Quick Fixes (on page 2308)* to be applied in bulk only for those that have the `sqf:default-fix` attribute explicitly set.



Note:
If you want to bulk apply quick fixes for documents that are not validated against Schematron, this option should not be selected.

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 81)* 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 2305)* or *plugin (on page 2307)*. 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 748)* (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 79)* 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 `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.

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.

Related information

Generating Sample XML Files *(on page 748)*

XProc Preferences

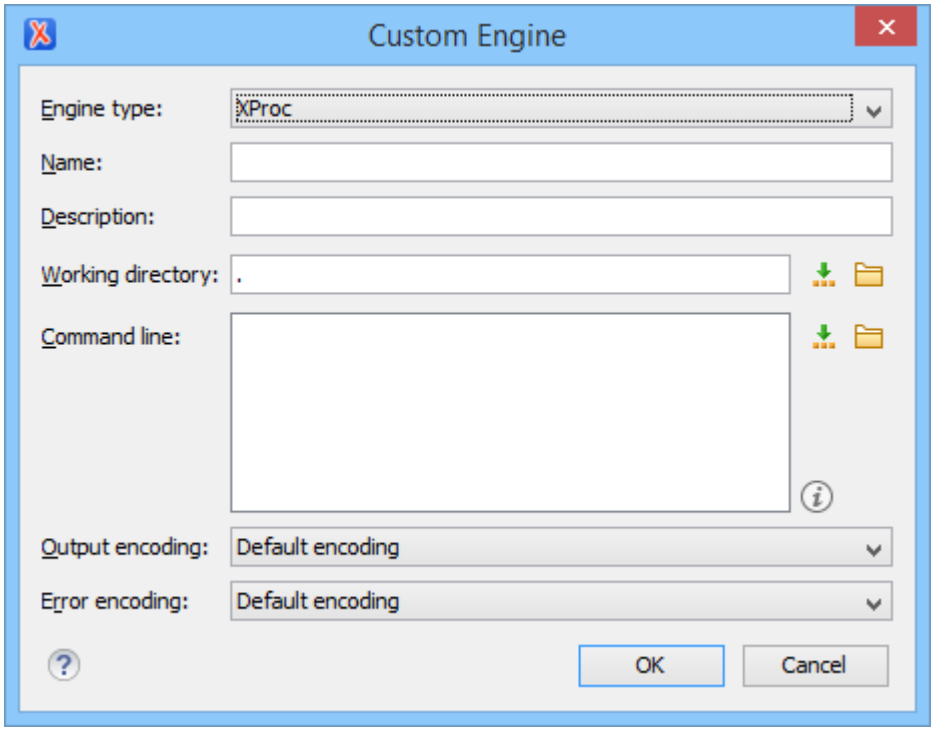
Oxygen XML Developer includes a bundled version of the *Calabash* XProc engine that can be used for XProc transformations and validation, but you also have several ways to integrate other external XProc engines.

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 1189)*.

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 79)* 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.

Figure 23. Creating an XProc 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 252)* button, or the **Browse** 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 252)* button, or the **Browse** 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 474)*.

XSLT/XQuery Preferences

To configure options regarding XSLT and XQuery processors, open the **Preferences** dialog box **(Options > Preferences)** *(on page 79)* 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 79)* 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 536)*.

Note:
If no specific engine is specified in the validation scenario and the XSLT file has a transformation scenario associated, Oxygen XML Developer 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 146)*, 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 79)* 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 79)* and go to **XML > XSLT-XQuery > XSLT > Saxon > Saxon-HE/PE/EE**.

Saxon-HE/PE/EE Options

Oxygen XML Developer allows you to configure the following XSLT options for the Saxon 12.9 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 12.9 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.

> **Attention:**
> Oxygen XML Developer does not support the `ALLOWED_PROTOCOLS` Saxon property (*http://saxon.sf.net/feature/allowedProtocols*). This feature is specific to Saxon and controls access by Saxon (but not by underlying software, such as the XML parser). For more information, see https://www.saxonica.com/documentation12/index.html#!configuration/config-features.

Debugger trace into XPath expressions (applies to debugging sessions)

Instructs the XSLT Debugger *(on page 1845)* 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 12.9 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 12.9 processors.

> **Note:**
> Saxon-CE, being JavaScript-based, was designed to run inside a web browser. This means that you will use Oxygen XML Developer 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 options available specifically for Saxon 12.9 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.

Saxon-HE/PE/EE Advanced Preferences

To configure the **Saxon HE/PE/EE Advanced** preferences, open the **Preferences** dialog box **(Options > Preferences)** *(on page 79)* and go to **XML > XSLT-XQuery > XSLT > Saxon > Saxon-HE/PE/EE > Advanced**.

You can configure the following advanced XSLT options for the Saxon 12.9 transformer (all three editions: Home Edition, Professional Edition, Enterprise Edition):

- **URI Resolver class name ("-r")** - Specifies a custom implementation for the URI resolver used by the XSLT Saxon 12.9 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 XSLT extension *(on page 1105)* 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 12.9 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 XSLT extension *(on page 1105)* for the particular transformation scenario.

XSLTProc Preferences (Deprecated)

To configure **XSLTProc** options, open the **Preferences** dialog box **(Options > Preferences)** *(on page 79)* 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 1070)*.
- **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 Developer 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 Developer will display in the **Warnings** view the URL of all the files loaded during transformation.
- **Show profile information** - If selected, Oxygen XML Developer 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 Developer 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.

XQuery Preferences

To configure the **XQuery** options, open the **Preferences** dialog box **(Options > Preferences)** *(on page 79)* 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 Developer 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 1123)*) 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 786)* 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 2307)*).



Note:
This option is ignored if you choose **Present as a sequence** *(on page 1123)* (lazy extract data from a database) from the associated transformation scenario.

Create structure indicating the type nodes

If selected, Oxygen XML Developer 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 1123)* (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 79)* and go to **XML > XSLT-XQuery > XQuery > Saxon-HE/PE/EE**.

Oxygen XML Developer allows you to configure the following XQuery options for the Saxon 12.9 Home Edition (HE), Professional Edition (PE), and Enterprise Edition (EE):

Use a configuration file ("-config")

Sets a Saxon 12.9 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.

The following option is available for Saxon 12.9 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 12.9 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.

Saxon HE/PE/EE Advanced Preferences

To configure **Saxon HE/PE/EE Advanced** preferences, open the **Preferences** dialog box **(Options > Preferences)** *(on page 79)* and go to **XML > XSLT-XQuery > XQuery > Saxon-HE/PE/EE > Advanced**.

The advanced XQuery options that can be configured for the Saxon 12.9 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 12.9 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 1105)* 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 564)*.

- **Collection URI Resolver class name** - Allows you to specify a custom implementation for the Collection URI resolver used by the XQuery Saxon 12.9 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 1105)* for the particular transformation scenario.

Debugger Preferences

To configure the **Debugger** preferences, open the **Preferences** dialog box **(Options > Preferences)** *(on page 79)* 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 Developer 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 79)* and go to **XML > XSLT-XQuery > Profiler** (see Debugger Preferences *(on page 188)*).

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 1853)*.

Ignore invocation less than

Invocations are ignored below this specified amount (in microseconds). For more information, see Invocation Tree View *(on page 1852)*.

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 79)* and go to **XML > XSLT-XQuery > XPath**.

Oxygen XML Developer 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 1729)* and the XPath toolbar *(on page 1727)* are unescaped during their execution. For example, the expression:

```
//varlistentry[starts-with(@os,'&#x73;')]
```

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 474)*.

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 Developer 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 Developer 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 Developer 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.

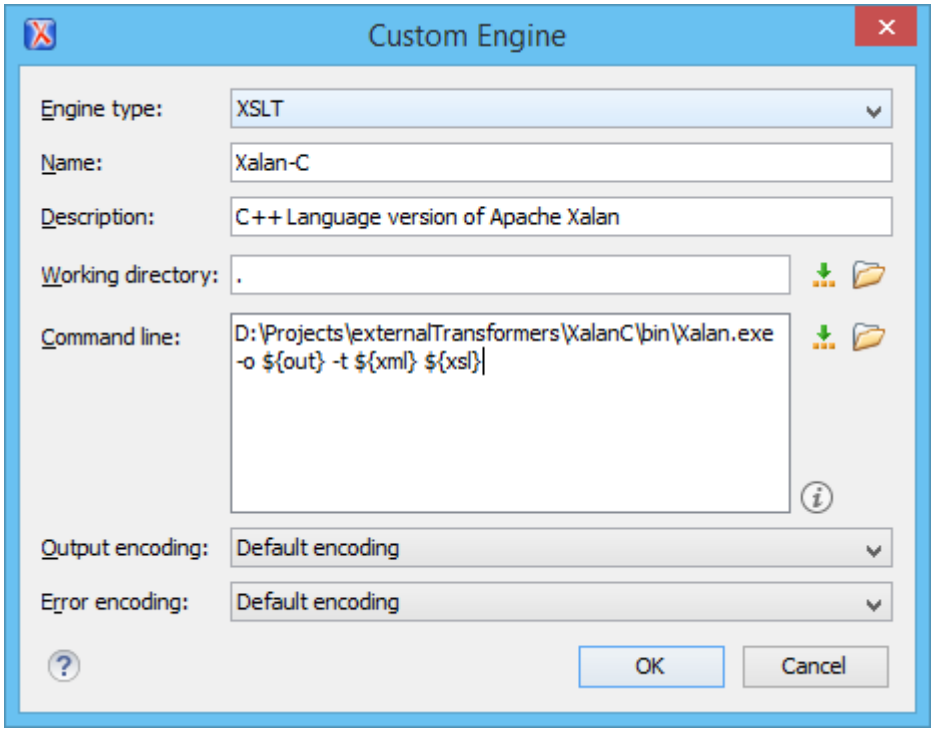
Custom Engines Preferences

Oxygen XML Developer allows you to configure custom processors to be used for running XSLT and XQuery transformations.

To configure the **Custom Engines** preferences, open the **Preferences** dialog box **(Options > Preferences)** *(on page 79)* 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.

Figure 24. Parameters of 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 252)* 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 Developer install directory.

Command line

The command line that must be executed by Oxygen XML Developer to perform a transformation with the engine. The following editor variables *(on page 252)* 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 Developer 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 Developer 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 Developer 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 1127)* and XML with XSLT transformation scenarios *(on page 1102)*.

To configure the options for the FO processors, open the **Preferences** dialog box **(Options > Preferences)** *(on page 79)* 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 Developer 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 Developer 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 252)* 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 Developer 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 252)* 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 1176)*.
- 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)

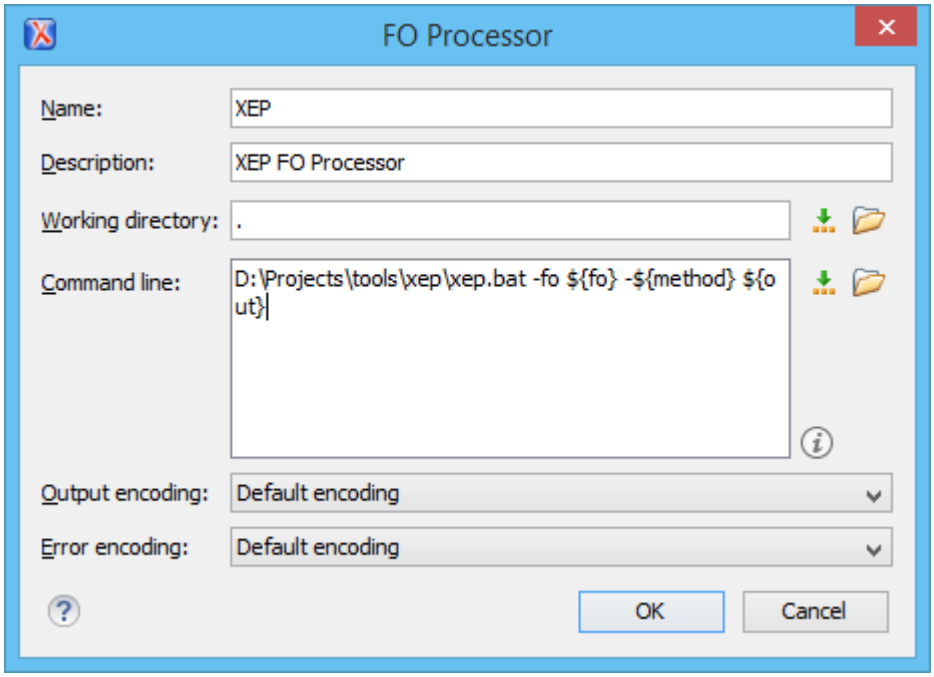
If **Antenna House Formatter** is already installed on your computer, you can use this button to choose the Antenna House executable script (`AHFCmd.exe` or `XSLCmd.exe` for Windows, and `run.sh` for Linux/macOS).



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 195)*, 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. A list with all AntennaHouse command line parameters can be found here: https://www.antenna.co.jp/AHF/help/en/ahf-xslcmd.html#cl-parameter.

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 25. 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 252)* button, or the **Browse** button. You can use one of the following editor variables *(on page 252)*:

- **${homeDir}** - The path to the user home directory.
- **${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 Developer 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 252)* button, or the **Browse** button. You can use one of the following editor variables *(on page 252)*:

- **${method}** - The FOP transformation method: **pdf**, **ps**, or **txt**.
- **${fo}** - The input FO file.
- **${out}** - The output file.
- **${pd}** - The project directory.
- **${frameworksDir}** - The path of the `frameworks` subdirectory of the Oxygen XML Developer installation directory.
- **${oxygenInstallDir}** - The Oxygen XML Developer 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 474)* at the bottom of the Oxygen XML Developer window.

Error Encoding

The encoding of the FO processor error stream that is displayed in a **Results** panel *(on page 474)* at the bottom of the Oxygen XML Developer window.

CSS-based Processors Preferences

Oxygen XML Developer 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 79)* 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 79)* and go to **XML > Ant**. This panel allows you to choose the directory containing the *Apache Ant (on page 2303)* libraries (the so-called *Ant Home*) that Oxygen XML Developer 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 Developer 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 79)* 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 Developer 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 2. Pattern Letters

Letter	Date or Time Component	Presentation	Examples
G	Era designator	Text	AD
y	Year	Year	1996; 96
M	Month in year	Month	July; Jul; 07
w	Week in year	Number	27
W	Week in month	Number	2
D	Day in year	Number	189
d	Day in month	Number	10
F	Day of week in month	Number	2
E	Day in week	Text	Tuesday; Tue
a	Am / pm marker	Text	PM
H	Hour in day (0-23)	Number	0
k	Hour in day (1-24)	Number	24
K	Hour in am / pm (0-11)	Number	0

Table 2. Pattern Letters (continued)

Letter	Date or Time Component	Presentation	Examples
h	Hour in am / pm (1-12)	Number	12
m	Minute in hour	Number	30
s	Second in minute	Number	55
S	Millisecond	Number	978
z	Time zone	General time zone	PST; GMT-08:00
Z	Time zone	RFC 822 time zone	-0800

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.

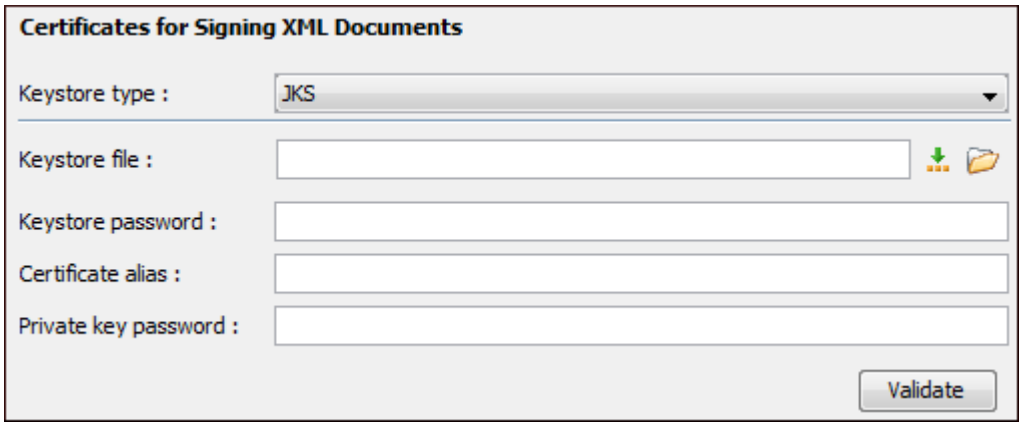
- **RFC 822 time zone**: The RFC 822 4-digit time zone format is used:
    - **RFC822TimeZone**
    - **TwoDigitHours** (must be between 00 and 23)

XML Signing Certificates Preferences

Oxygen XML Developer provides two types of *keystores (on page 2306)* 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 79)* and go to **XML > XML Signing Certificates**. You can customize the following parameters of a *keystore*:

Figure 26. Certificates Preferences Panel



- **Keystore type** - The type of *keystore (on page 2306)* that Oxygen XML Developer 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 2306)*.
- **Private key password** - The private key password of the certificate (required only for JKS *keystores (on page 2306)*).
- **Validate** - Click this button to verify the configured *keystore (on page 2306)* 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)** *(on page 79)* 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** *(on page 252)* button, or the **Browse** button. Oxygen XML Developer 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)** *(on page 79)* 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 Developer 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 4.x)

Oxygen XML Developer comes bundled with the **Oxygen Publishing Engine** (based on DITA-OT 4.4.0). 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-OT*.

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** *(on page 252)* 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)* or DITA Map PDF - based on HTML5 & CSS *(on page)* 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 Developer default distribution, you might encounter validation-related issues.



CAUTION:
Oxygen XML Developer support engineers do not officially offer support and troubleshooting assistance for custom DITA-OT distributions or custom installed DITA-OT plugins. If you discover any issues or inconsistent behavior while using a custom DITA-OT or a DITA-OT that

contains custom DITA-OT plugins, 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 299)* 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 New Topics Preferences

To access the DITA New Topics preferences page, open the **Preferences** dialog box **(Options > Preferences)** *(on page 79)* 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 Document Wizard (on page 299)*. 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.

DITA Publishing Preferences

To access the DITA Publishing preferences page, open the **Preferences** dialog box **(Options > Preferences)** *(on page 79)* 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 79)* 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 79)* 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 564)* for one of the following URIs:

- **http://www.oxygenxml.com/schematron/validation/markdown-as-html**
- **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 79)* 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 210)*.

Connection Wizards Section

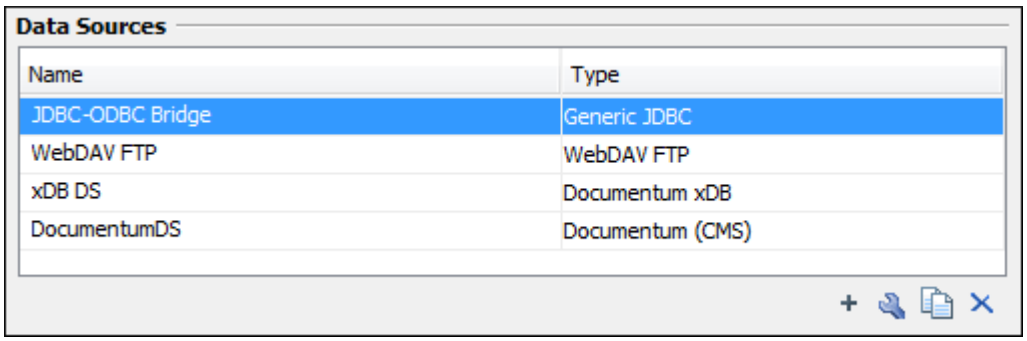
Create eXist-db XML connection

Click this link to open the dedicated **Create eXist-db XML connection** dialog box *(on page 1761)* that provides a quick way to create an eXist connection.

Data Sources Section

This section allows you to add and configure data sources.

Figure 27. Data Sources Preferences Panel

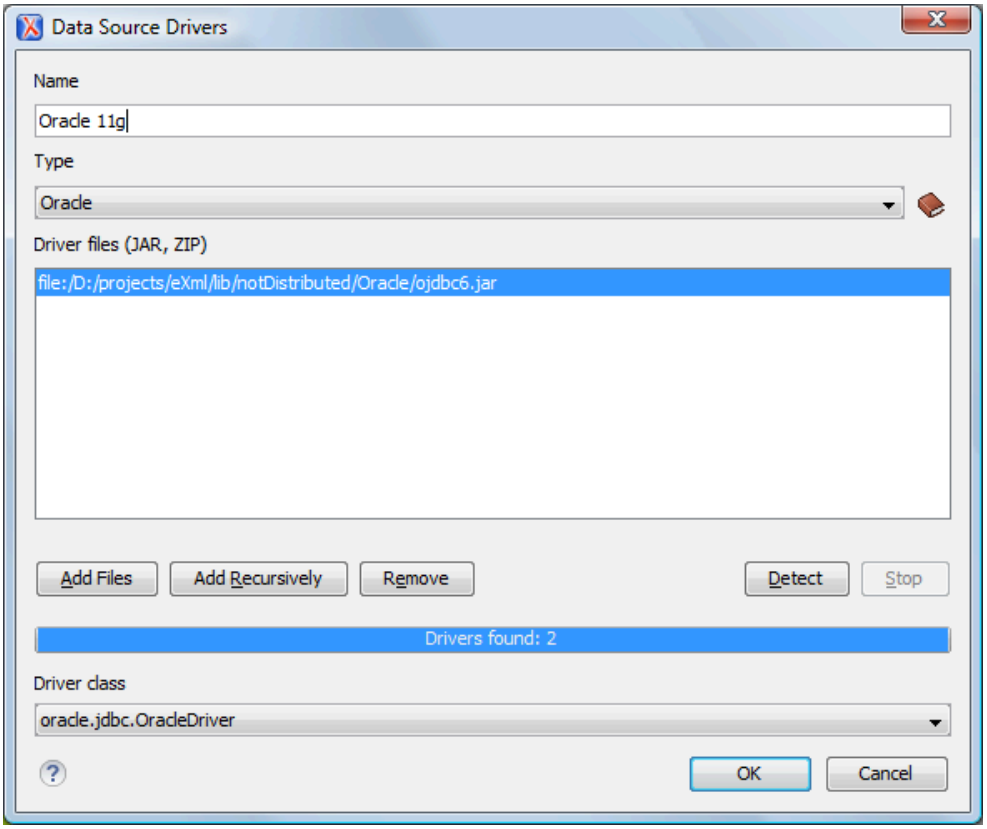


The following buttons are available at the bottom of the **Data Sources** panel:

+ New

Opens the **Data Sources Drivers** dialog box that allows you to configure a new database driver.

Figure 28. 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 210)* 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 210)* that are necessary for accessing databases in Oxygen XML Developer.
- **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.

Edit

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.

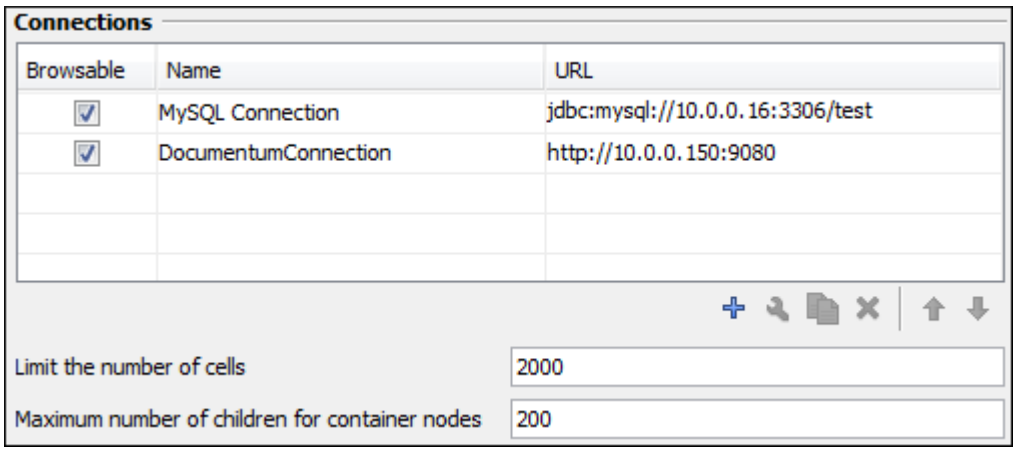
Delete

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 29. Connections Preferences Panel

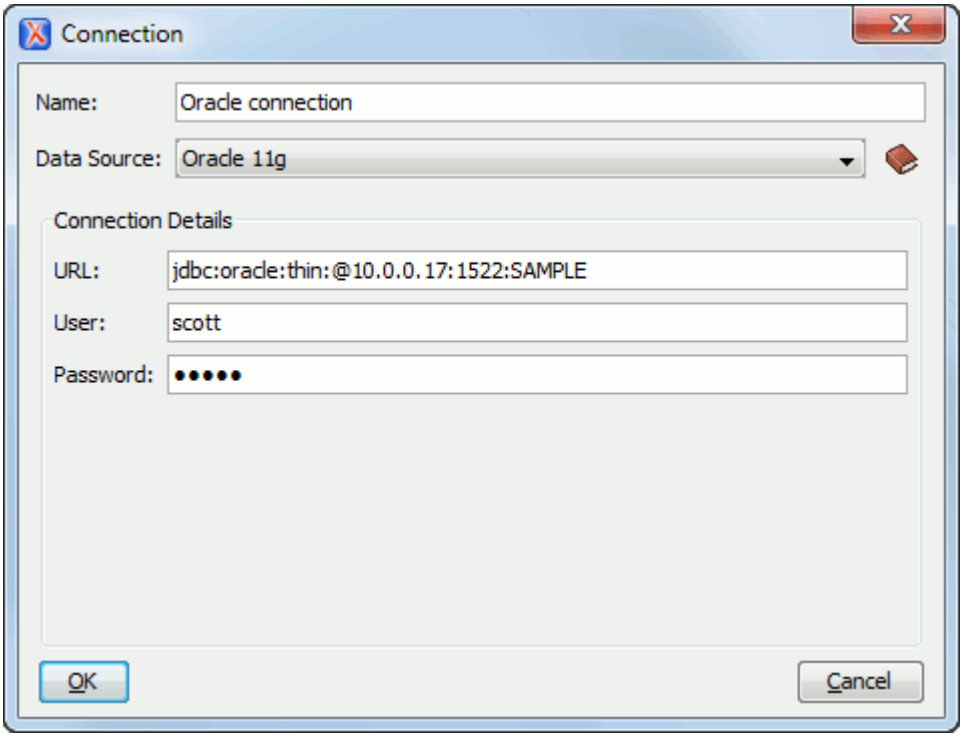


The following buttons and options are available at the bottom of the **Connections** panel:

New

Opens the **Connection** dialog box that allows you to configure a new database connection.

Figure 30. Connection Dialog Box



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.
- **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 1744)* 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 1744)*, 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 1742)* 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 1742)*. 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 1742)*. To open this preferences page, open the **Preferences** dialog box **(Options > Preferences)** *(on page 79)* 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**
- **Table**
- **View**

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.

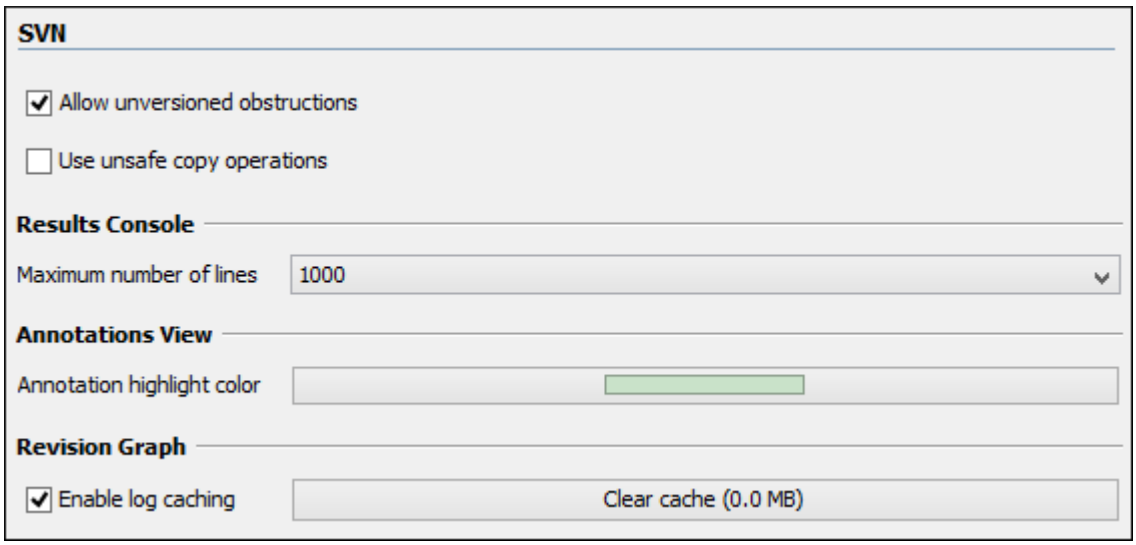
In addition, the following is a list of other popular databases along with instructions for getting the drivers that are necessary to access the databases in Oxygen XML Developer:

- **IBM DB2 Pure XML database** (Deprecated) - 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 zip file and use the `db2jcc.jar` and `db2jcc_license_cu.jar` files in Oxygen XML Developer for configuring a DB2 data source *(on page 1785)*.
- **eXist database** - Copy the *jar* files from the eXist database install directory to the Oxygen XML Developer install directory as described in the procedure for configuring an eXist data source *(on page 1762)*.
- **MarkLogic database** (Deprecated) - Download the MarkLogic driver from MarkLogic Community site.
- **Oracle 11g database** (Deprecated) - 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** (Deprecated) - Go to https://jdbc.postgresql.org/download/ and download the PostgreSQL JDBC driver specific for your server version.
- **Microsoft SQL Server 2019 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 79)* 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.

Figure 31. SVN Preferences Panel



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.

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 79)* 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 Developer 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. 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 79)* 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 **${firstFile}** and **${secondFile}** specify the positions of the two compared files in the command line for the external diff application. The parameter **${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 Developer 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 79)* 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.

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 79)* 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

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 `a b` sequence with `a b`, Oxygen XML Developer finds no differences, because after normalization, all whitespaces from the first sequence are collapsed into a single space character. However, when comparing `a b` with `ab` (no whitespace between `a` and `b`), Oxygen XML Developer 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.
- **Words** - Computes the differences at word level, meaning that it compares two files or fragments looking for identical words.

- **Lines** - Computes the differences at line level, meaning that it compares two files or fragments looking for identical lines of text.
- **Syntax Aware** - Computes differences for the file types or fragments known by Oxygen XML Developer, taking the syntax (the specific types of tokens) into consideration.
- **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.
- **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

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

If selected, you can enter an XPath expression *(on page 1726)* 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

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

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

If selected, empty elements in both expansion states are considered matched (that is `<element/>` and `<element></element>` are considered equal).

Note:
The File Comparison preferences are also accessible through the **Directory Compare Tool** *(on page 421)* and are taken into account when directory comparisons are invoked.

Appearance Preferences

To configure the appearance options for the Files Comparison tool, open the **Preferences** dialog box **(Options > Preferences)** *(on page 79)* 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.

Show comment callouts

Displays any existing comment callouts for both compared files.

Comment callout width

Specifies the width used for rendering the comment callouts. The default value is 200 pixels.

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.

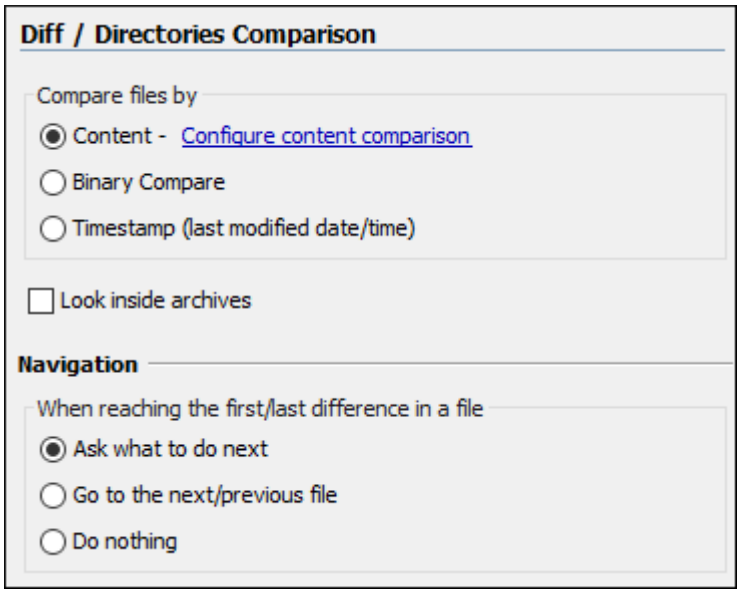
Conflict color

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 79)* and go to **Diff > Directories Comparison**.

Figure 32. Diff Preferences Page



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 215)*. 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 215)* 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 220)* are considered directories and their content is compared just like regular files.

Navigation

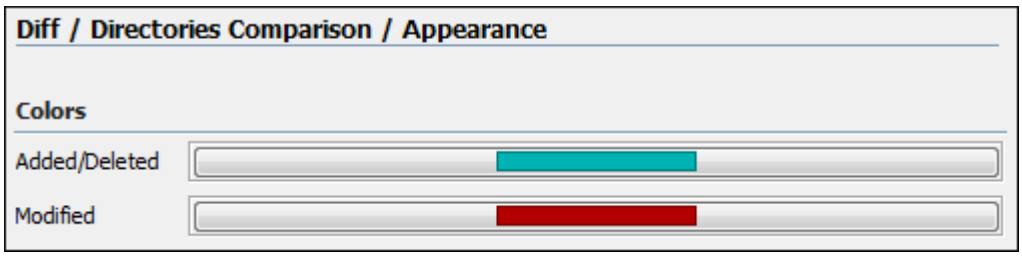
This options control the behavior of the differences traversal actions (**Go to previous modification**, **Go to next modification**) when the first or last difference in a file is reached:

- **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 79)* and go to **Diff > Directories Comparison > Appearance**.

Figure 33. 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 79)* 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.

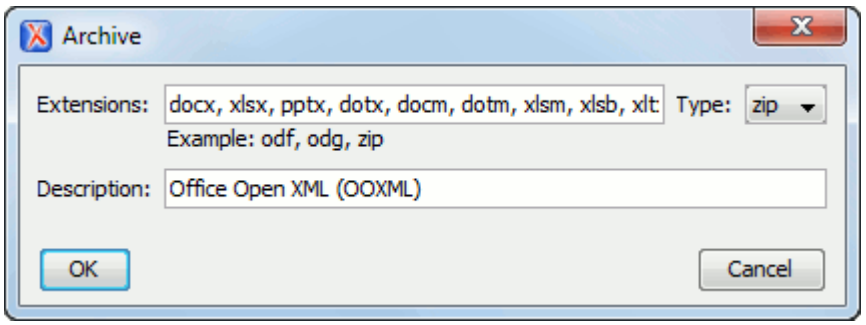


Note:
Backup files have the name `originalArchiveFileName.bak` and are located in the same folder as the original archive.

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 Developer. 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 34. Edit Archive Extension Mappings





Important:
You have to restart Oxygen XML Developer 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 (on page 2307)* that extend the functionality of Oxygen XML Developer. The *plugins* are shipped as separate packages. To check for new *plugins*, go to http://www.oxygenxml.com/oxygen_sdk.html.

A *plugin* consists of a separate sub-folder in the `Plugins` folder of the Oxygen XML Developer 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 Developer 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 79)* 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 1891)*

External Tools Preferences

A command-line tool can be started in the Oxygen XML Developer 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 Developer. To access this preferences page, open the **Preferences** dialog box **(Options > Preferences)** *(on page 79)* 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 291)*).

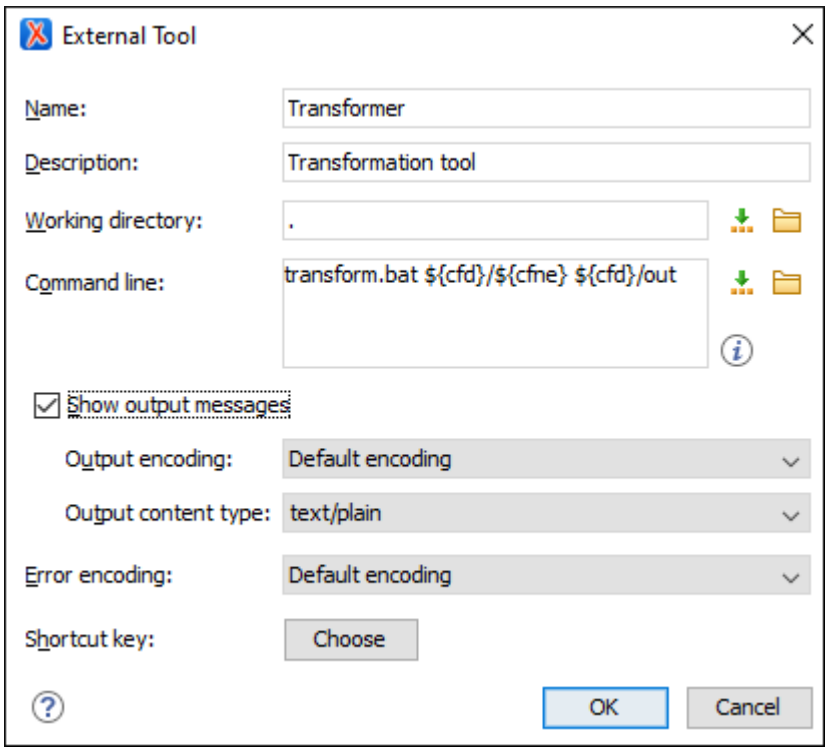
How to Configure an External Tool

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.

Figure 35. 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** *(on page 252)* button, or the **Browse** button. You can use one of the following editor variables: **${cfd}** *(on page 258)*, **${pd}** *(on page 259)*, **${oxygenInstallDir}** *(on page 259)*, **${homeDir}** *(on page 259)*, **${system(var.name)}** *(on page 260)*, **${date(pattern)}** *(on page 258)*, **${xpath_eval(expression)}** *(on page 260)*.

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** *(on page 252)* button, or the **Browse** button. You can use one of the following editor variables: **${homeDir}** *(on page 259)*, **${home}** *(on page 259)*, **${cfn}** *(on page 258)*, **${cfne}** *(on page 258)*, **${cf}** *(on page 257)*, **${currentFileURL}** *(on page 258)*, **${cfd}** *(on page 258)*, **${cfdu}** *(on page 258)*, **${tsf}** *(on page 260)*, **${pd}** *(on page 259)*, **${pdu}** *(on page 259)*, **${oxygenInstallDir}** *(on page 259)*, **${oxygenHome}** *(on page 259)*, **${frameworksDir}** *(on page 259)*, **${frameworks}** *(on page 259)*, **${ps}** *(on*

page 260), **${timeStamp}** *(on page 260)*, **${uuid}** *(on page 260)*, **${id}** *(on page 259)*, **${afn}** *(on page 252)*, **${afne}** *(on page 252)*, **${af}** *(on page 252)*, **${afu}** *(on page 252)*, **${afd}** *(on page 252)*, **${afdu}** *(on page 252)*, **${ask('message', type, 'default_value')}** *(on page 253)*, **${dbgXML}** *(on page 258)*, **${dbgXSL}** *(on page 258)*, **${env(VAR_NAME)}** *(on page 258)*, **${system(var.name)}** *(on page 260)*, **${date(pattern)}** *(on page 258)*, and **${xpath_eval(expression)}** *(on page 260)*.

Show output messages

When this option is selected, all the messages emitted by the external tool are displayed in the **Results** view *(on page 474)* view. 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 byOxygen XML Developer to read the output of the tool.
- **Output content type** - A list of predefined content type formats that instructOxygen XML Developer 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 Developer to read the error stream.

Shortcut key

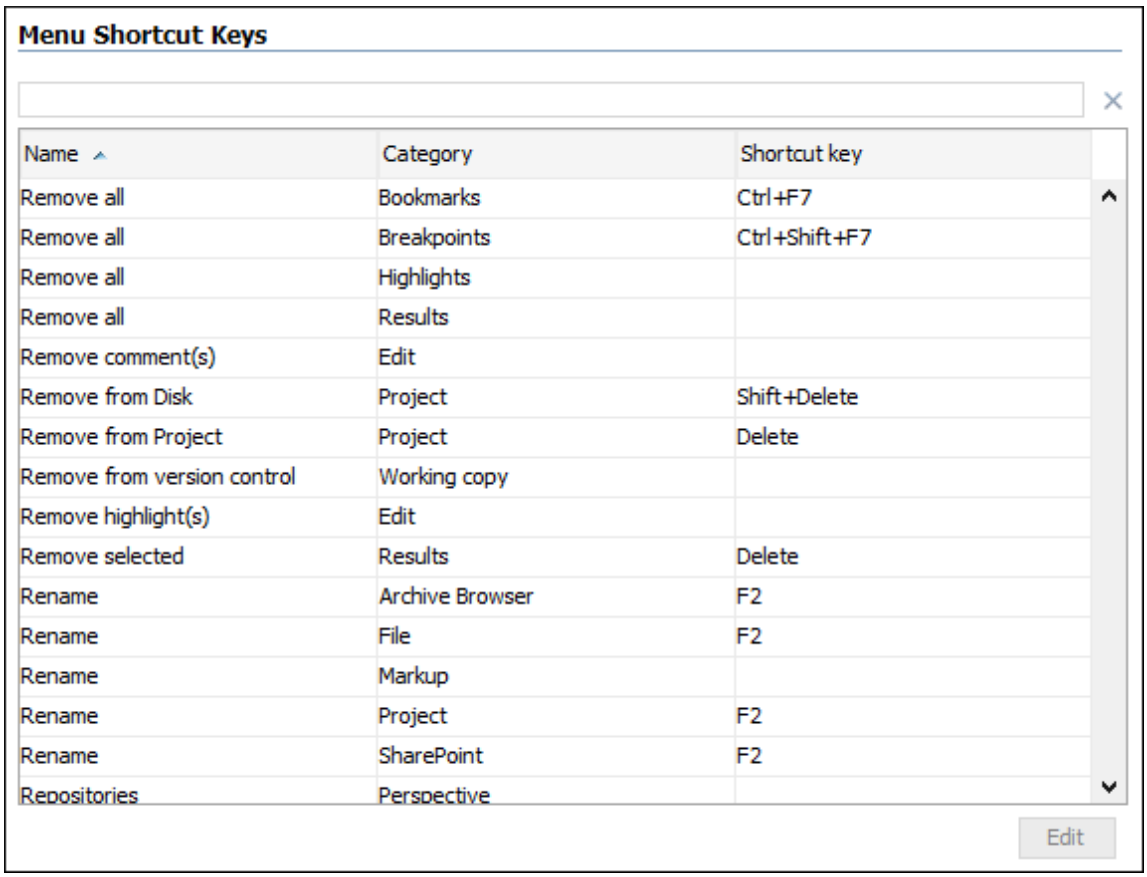
You can choose a keyboard shortcut that can be used to launch the external tool.

Menu Shortcut Keys Preferences

You can use the **Menu Shortcut Keys** preferences page to configure shortcut keys for the actions available in Oxygen XML Developer. 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 79)* 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 293)*.

Figure 36. Menu Shortcut Keys Preferences Page



The **Menu Shortcut Keys** preferences page also contains the shortcuts that you define at *document type* level *(on page 105)*.

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 plug-ins.

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.

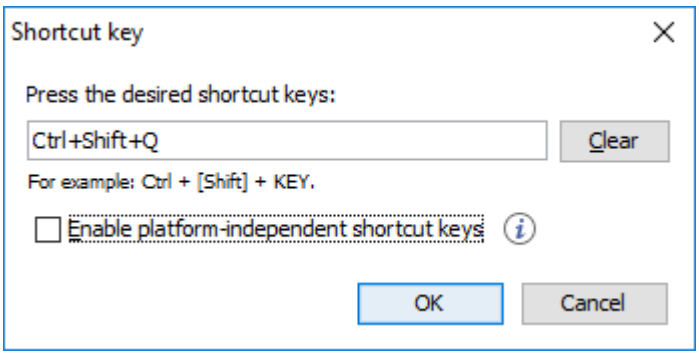
How to Assign a Shortcut Key or Edit an Existing Shortcut

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.

   Figure 37. Shortcut Key Configuration Dialog Box

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 2226)* or Keyboard Shortcuts Do Not Work At All *(on page 2225)*.

Related information

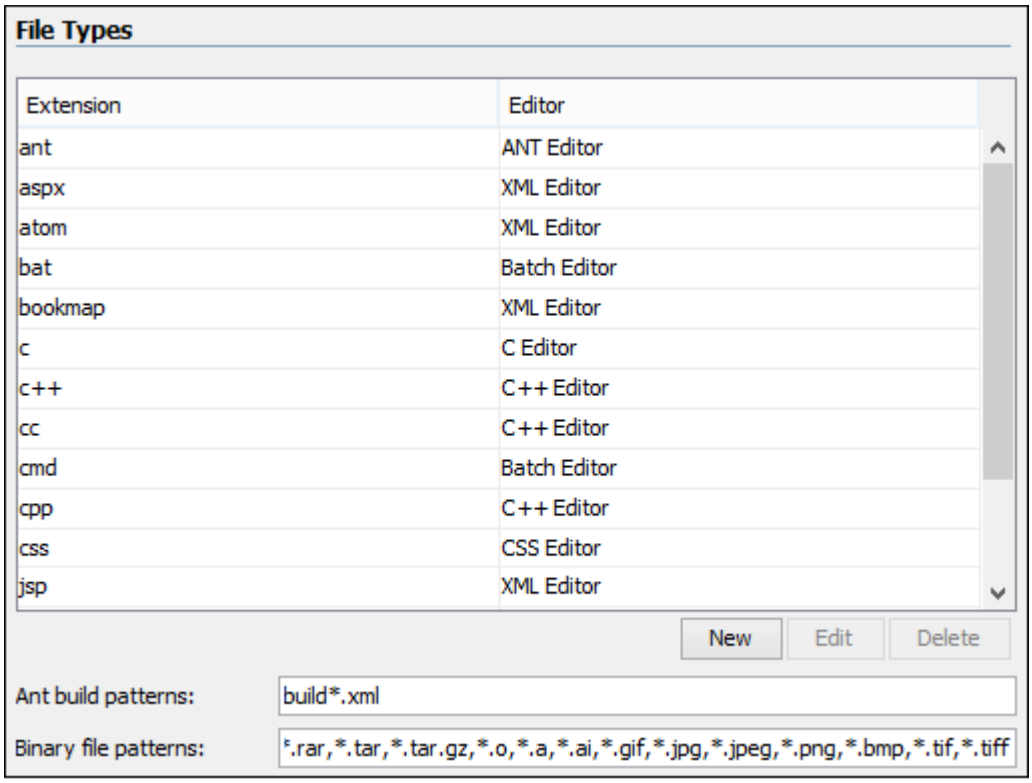
Frequently Used Shortcut Keys *(on page 293)*

File Types Preferences

Oxygen XML Developer 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 79)* and go to **File Types**.

Figure 38. File Types Preferences Page



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 123)* 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 334)*: **File/Replace in Files**, **Check Spelling in Files**, **Validate**.



Note:
If you associate an empty extension to a content type, all files without extensions will be opened with that specific content type.

Open/Find Resource Preferences Page

You can configure various options that pertain to the **Open/Find Resource** dialog box *(on page 358)* and **Open/Find Resource** view *(on page 354)*. To access these options, open the **Preferences** dialog box **(Options > Preferences)** *(on page 79)* 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.

Only index files referenced in the context DITA map as "resource-only"

If selected, only files referenced as "resource-only" in the context DITA map or in its referenced sub maps are indexed. By default, the setting is not selected.



Tip:
It makes sense to enable this setting if the current opened project contains lots of resources to speed up computation of the reusable components displayed in the **DITA Reusable Components** view's **Components** tab.

Limit search results to

Specifies the maximum number of results that are displayed in the **Open/Find Resource** dialog box/view *(on page 358)*.

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 358)* 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 358)* 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 2304)* 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 81)*. 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 Developer 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 360)* 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 229)* 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 354)*
Open/Find Resource Dialog Box *(on page 358)*

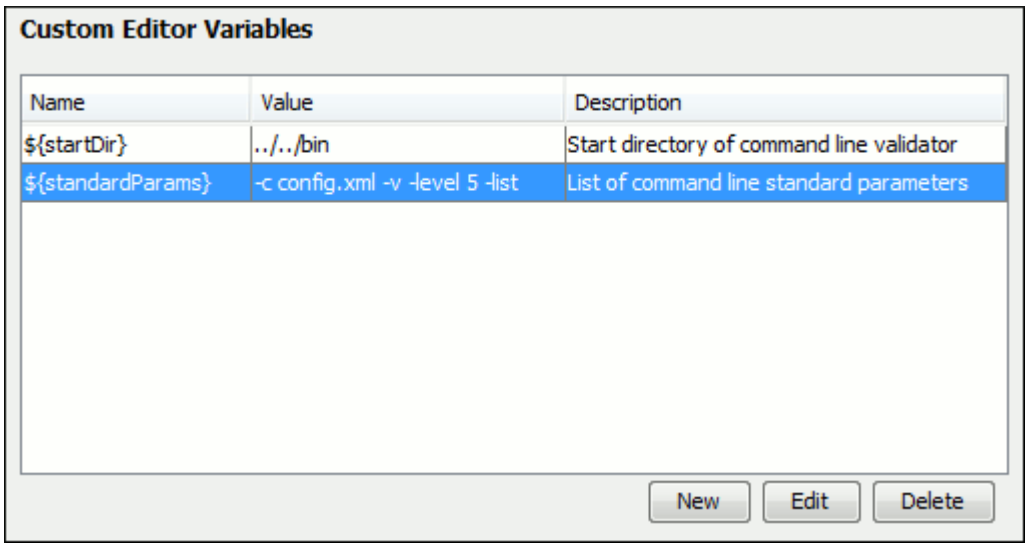
Custom Editor Variables Preferences

An editor variable *(on page 252)* 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 252)*.

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 79)* 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.

Figure 39. Custom Editor Variables Table



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 79)* 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.

HTTP(S)/WebDAV Preferences

To set the **HTTP(S)/WebDAV** preferences, open the **Preferences** dialog box **(Options > Preferences)** *(on page 79)* 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 Developer 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 Developer 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 Developer tries to re-send the request to the server.

Cache content for similar HTTP requests when opening and validating documents

When opening XML documents that contain lots of `xi:include` elements over HTTP (for example), depending on how content is reused, there may be lots of HTTP requests to the same target files during the validation or opening of the XML document. When this setting is selected,

HTTP calls to the same target file are cached and the opening and validation of such XML documents may take less time.

Automatically accept a security certificate, even if invalid

When selected, the HTTPS connections that Oxygen XML Developer 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.

SFTP Preferences

To configure the **SFTP** options, open the **Preferences** dialog box **(Options > Preferences)** *(on page 79)* and go to **Network Connection Settings > SFTP**. You can customize the following options:

Encoding for FTP control connection

The encoding used to communicate with SFTP servers: either ISO-8859-1 or UTF-8. If the server supports the UTF-8 encoding, Oxygen XML Developer 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 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 318)*.

Passphrase

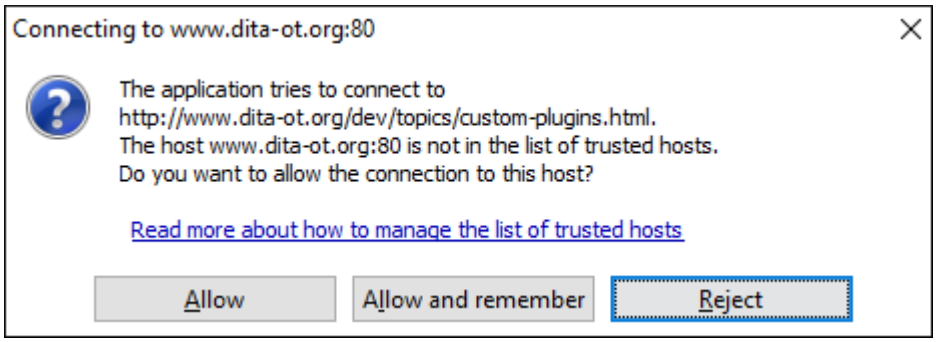
The passphrase used for the private key method of authentication of the SFTP protocol. The user / password method of authentication has precedence if it is used in the Open URL dialog box *(on page 318)*.

Trusted Hosts Preferences

Oxygen XML Developer 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.

Figure 40. Trusted Hosts Confirmation Dialog Box



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 Developer will allow connections to the listed hosts without requesting user confirmation.

Note:
Connections defined in the **Data Sources** preferences page *(on page 205)* or accepted by add-ons are also considered trusted.

To add or remove domains, open the **Preferences** dialog box **(Options > Preferences)** *(on page 79)* 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 79)* 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 465)*, open the **Preferences** dialog box **(Options > Preferences)** *(on page 79)* 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 79)* 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 334)*. 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 437)*.

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 79)* 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 Developer 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 334)*, **Data Source Explorer** view *(on page 1742)*, and **Archive Browser** *(on page 1735)*. 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 1729)* view when editing long XPath expressions.

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, Base 32, 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 495)*, *Base32 (on page 496)*, or *Hex* conversions *(on page 497)*. 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.

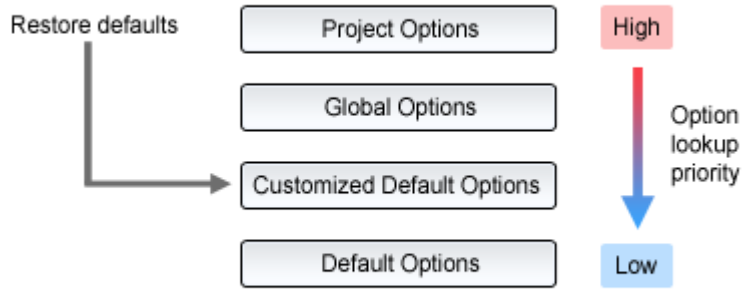
Convert DB Structure to XML Schema

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 766)* 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 Developer, 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 Developer includes several distinct layers of option values.

Figure 41. Option Lookup Priority



The option layers are as follows (sorted from high priority to low):

- **Project Options** *(on page 241)*

  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 241)*

  Allows individual users to personalize Oxygen XML Developer according to their specific needs.

- **Customized Default Options** *(on page 238)*

  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 Developer when an end-user selects the **Restore defaults** *(on page 80)* or **Reset Global Options** *(on page 242)* actions.

- **Default Options**

  The predefined default values, tuned so that Oxygen XML Developer 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 Developer has an extensive set of options that you can configure. When Oxygen XML Developer in 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 Developer and open the **Preferences** dialog box **(Options > Preferences)** *(on page 79)*.
3. Go through the options and set them to the desired defaults. Make sure that **Global Options** *(on page 2305)* 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 334)*, create a project or open an existing one.
3. Open the **Preferences** dialog box **(Options > Preferences)** *(on page 79)*.
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 2308)* 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 Developer 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 240)*.

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`).

  Note:
  The XML file placed in the `preferences` folder can have any name as long as it has the extension `xml` or `xpr`. If there are multiple such files in the `preferences` folder only the first one found is used.

- **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 `com.oxygenxml.default.options` system property in the startup parameters *(on page 267)*. The path can be specified with any of the following:
  - A URL or file path relative to the application installation folder. For example:

    ```
    -Dcom.oxygenxml.default.options=options/default.xml
    ```

  - A system variable that specifies the file path. For example:

```
com.oxygenxml.default.options=${system(CONFIG)}/default.xml
```

- An environmental variable that specifies the file path. For example:

```
com.oxygenxml.default.options=${env(CONFIG)}/default.xml
```

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 Developer.

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 Developer starts.

Related information
Sharing Application Settings *(on page 241)*

Storing Global and Project Level Options

When you configure the Oxygen XML Developer 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 241)* or by sharing the stored project-level files *(on page 241)*. The same is true with transformation and validation scenarios.

For each preferences page, you can choose between **Global Options** *(on page 241)* and **Project Options** *(on page 241)* 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 42. 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 241)*).

Global options are stored locally in option files (for example, `oxyOptionsSa19.1.xml` for a standalone distribution of Oxygen XML Developer version 19.1) located in the following directories:

- **Windows (7, 8, 10)** - `[user_home_directory]\AppData\Roaming\com.oxygenxml.developer`
- **macOS** - `[user_home_directory]/Library/Preferences/com.oxygenxml.developer`
- **Linux/Unix** - `[user_home_directory]/.com.oxygenxml.developer`

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 241)*.

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 241)*
Customizing Default Options *(on page 238)*
Importing/Exporting/Resetting Global Options *(on page 242)*

Sharing Application Settings

There are a variety of ways that you can share the settings in Oxygen XML Developer 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 Developer include a **Project Options** *(on page 2308)* button that allows you to pass changes to the settings to the current project file that is opened in the **Project** view *(on page 334)*. 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 347)*.

Share Settings by Exporting/Importing Global Options

Oxygen XML Developer includes actions in the **Options** menu that allow you to export and import the global settings *(on page 2305)*. 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 242)*.

Share Settings with a Custom Options File During Installation

When Oxygen XML Developer in 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 Developer 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 238)*.

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 347)*
Sharing Transformation Scenarios *(on page 1216)*
Sharing Validation Scenarios *(on page 546)*
Customizing Default Options *(on page 238)*
Importing/Exporting/Resetting Global Options *(on page 242)*

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 2305)* 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 240)*.

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 238)*. 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 347)* (`.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 Developer 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.developer` for Windows
- `[user-home-folder]/Library/Preferences/com.oxygenxml.developer` for macOS
- `[user-home-folder]/.com.oxygenxml.developer` for Linux

The name of the `options` file of Oxygen XML Developer 28.1 is oxyDeveloperOptionsSa28.1.xml.

Configuring the Layout of the Views and Editors

All of the side-views available in Oxygen XML Developer are *dockable (on page 2304)* 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 291)*.

To open a view, select it from the **Window > Show View** menu. You can hide a view by closing it with the ✕ 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 Developer window. Once you create a layout that suits your needs, you can save it from **Window > Export Layout**. Oxygen XML Developer 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**.



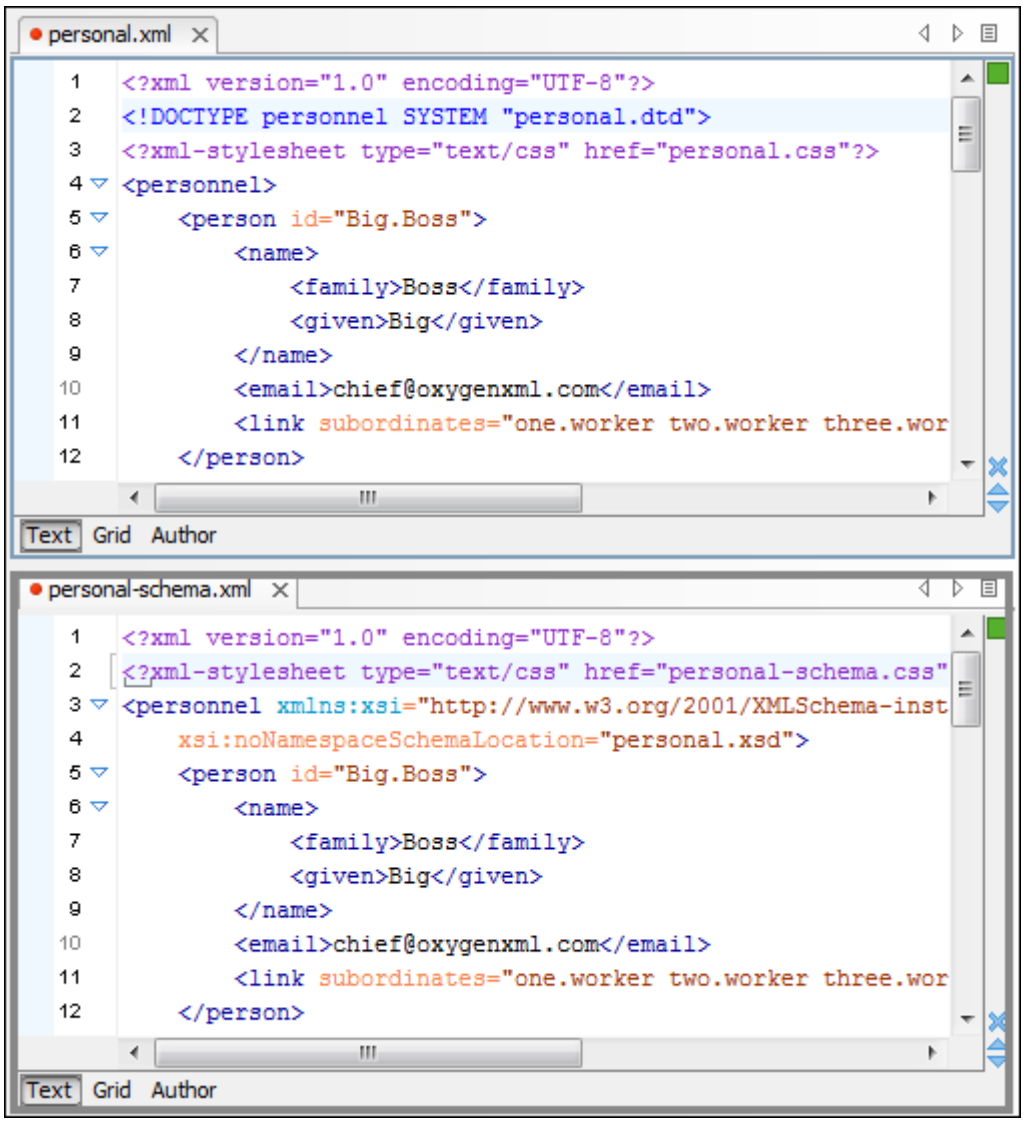
Note:
The **Load Layout** menu lets you select between the default layout, a predefined layout, or a custom layout. The changes you make using the **Load Layout** menu are also reflected in the **Application Layout** preferences page.

The changes you make to any layout are preserved between working sessions. The predefined layout files are saved in the preferences directory *(on page 80)* of Oxygen XML Developer.

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 43. Drag and Drop Editors



Hide or Float Views

Hide

To gain more editing space in the Oxygen XML Developer window, click **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 Developer 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 Developer 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 Developer 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 82)*.

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 + Option + 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 83)*.

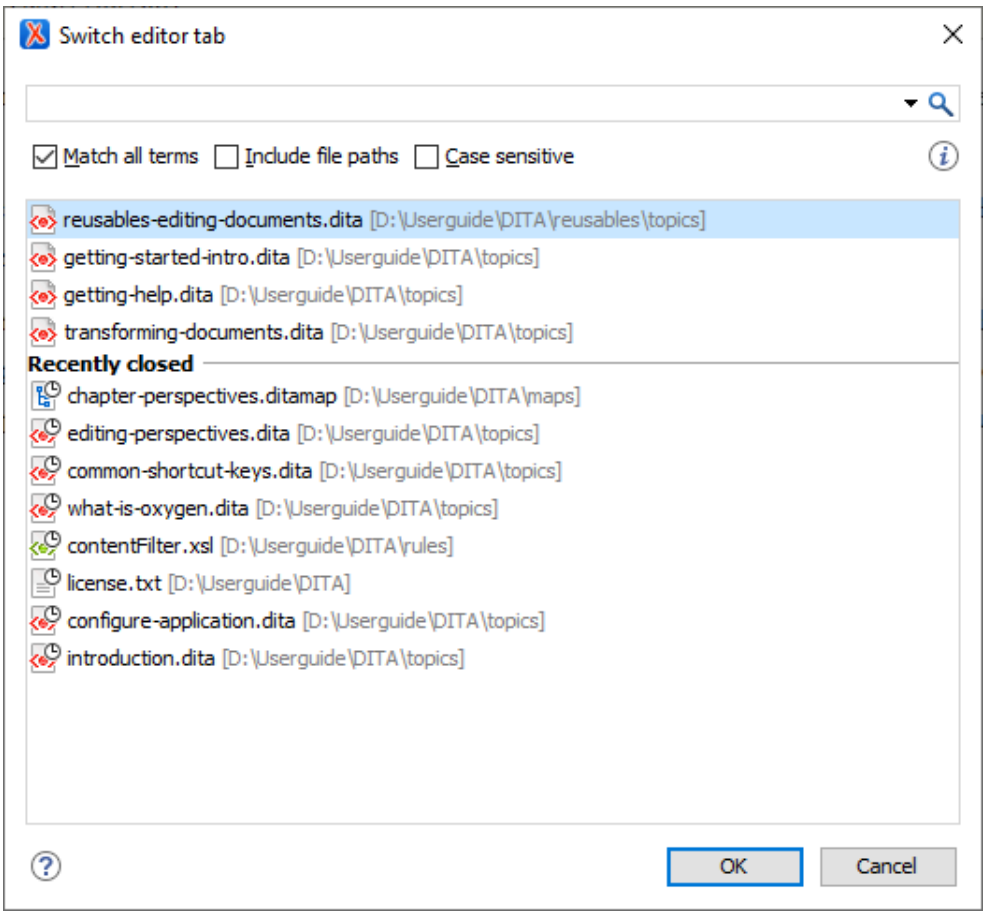
Ctrl + Shift + Tab (Command + Shift + Tab on macOS) [NOTE: Ctrl + Page Up (Ctrl + Option + Left Arrow on macOS) does the same]

Switches to the previous open tab in the order specified in the **Order of switching between editor tabs** option *(on page 83)*.

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.

Figure 44. Switch Editor Tab Dialog Box



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.

Resources

For more information about configuring the interface of Oxygen XML Developer, 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 291)*

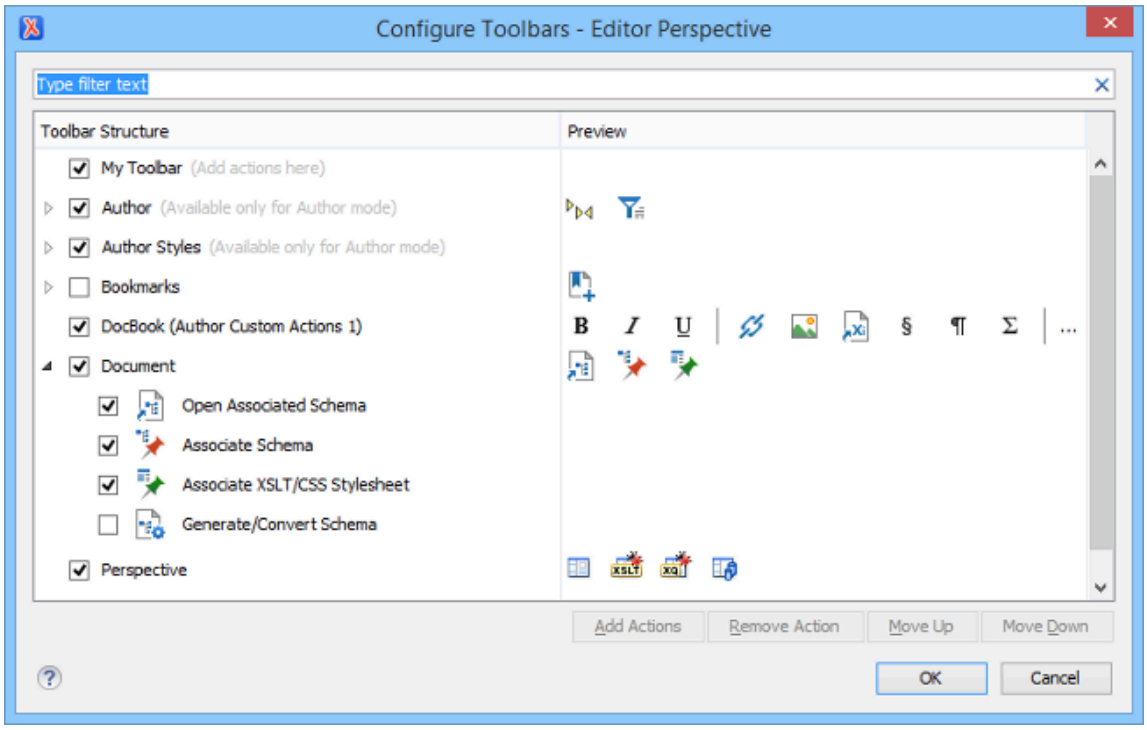
Configuring Toolbars

You can configure the toolbars in Oxygen XML Developer to personalize the interface for your specific needs. You can choose which toolbars to show or hide in the current editor mode (**Text**, **Design**, or **Grid**) and in the current *perspective (on page 2307)* (**Editor**, **XSLT Debugger**, **XQuery Debugger**, or **Databse**). 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 45. 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 2307)* 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 2305)* or 3rd party *plugins (on page 2307)*.

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 Developer.

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 285)*

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 2305)* 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 Developer includes a variety of built-in editor variables. You can also create your own custom editor variables by using the **Custom Editor Variables** preferences page *(on page 229)*.

Editor variables are evaluated and automatically expanded in many places in the application, when:

- Creating new documents from file templates *(on page 307)*.
- Inserting code templates *(on page 307)* in the **Text** mode.
- Running custom configured External Tools *(on page 2197)*.
- Running validation scenarios *(on page 536)* that use editor variables inside to reference various resources.
- Executing transformation scenarios (of type ANT, DITA-OT *(on page)*, XSLT *(on page 1102)*, etc.) that have editor variables set as parameter values or as values for references to various resources.
- Using specific Java API `UtilAccess.expandEditorVariables(String, URL)` from plugins and framework extensions.

You can use the following editor variables in Oxygen XML Developer commands of external engines or other external tools, and in various places in the application, such as in transformation scenarios, and validation scenarios:

- **${activeConditionSet}** - Current active profiling condition set name. If there is no active condition set, the variable will be replaced with an *empty string*.
- **${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 version="1.0" encoding="UTF-8"?>
  <!DOCTYPE topic PUBLIC "-//OASIS//DTD DITA Topic//EN" "topic.dtd">
  ```

```
<topic id="topic_lcf_1c4_tdb">
    <title></title>
    <body>
        <data name="${ask('Set a data name', String, 'name', @name)}"></data>
        <p>The name is: ${answer(@name)}</p>
    </body>
</topic>
```

Here is a more complex example of using the answer to an ask in the context of an XPath evaluation:

```
<topicref keys="${xpath_eval(tokenize('${answer(@firstTopic)}', '/')[last()]
 ! substring-before(., '.dita'))}" href="${answer(@firstTopic)}"/>
```

- **${ask('message', type, ('real_value1':'rendered_value1'; 'real_value2':'rendered_value2'; ...), 'default_value', @id)}** - To prompt for values at runtime, use the *ask('message', type, ('real_value1':'rendered_value1'; 'real_value2':'rendered_value2'; ...), 'default-value')* editor variable. You can set the following parameters:
  - **'message'** - The displayed message. Note the quotes that enclose the message.
  - **'default-value'** - Optional parameter. Provides a default value.
  - **@id** - Optional parameter. Used for identifying the variable to reuse the answer using the **${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".



Notice:

Editor variables that are used within a parameter of another editor variable must be escaped within single quotes for them to be properly expanded. For example:

```
${ask( 'Provide a date',generic,'${date(yyyy-MM-dd'T'HH:MM)}')}
```

Parameter	
generic (default)	**Format**: *${ask('message', generic, 'default')}*
	Description: The input is considered to be generic text that requires no special handling.
	Example:

Parameter	
	▪ *${ask('Hello world!')}* - The dialog box has a *Hello world!* message displayed. ▪ *${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!'*.
url	**Format:** *${ask('message', url, 'default_value')}*
	Description: Input is considered a URL. Oxygen XML Developer checks that the provided URL is valid.
	Example: ▪ *${ask('Input URL', url)}* - The displayed dialog box has the name *Input URL*. The expected input type is URL. ▪ *${ask('Input URL', url, 'http://www.example.com')}* - The displayed dialog box has the name *Input URL*. The expected input type is URL. The input field displays the default value *http://www.example.com*.
relative_url	**Format:** *${ask('message', relative_url, 'default')}*
	Description: Input is considered a URL. This parameter provides a file chooser, along with a text field. Oxygen XML Developer tries to make the URL relative to that of the document you are editing. **Note:** If the *$ask* editor variable is expanded in content that is not yet saved (such as an *untitled* file, whose path cannot be determined), then Oxygen XML Developer will transform it into an absolute URL.
	Example: *${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:

Parameter	
	▪ *${ask('Input password', password)}* - The displayed dialog box has the name *'Input password'* and the input is hidden with bullet symbols. ▪ *${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()}*. **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')}*

Parameter	
	▪ *${ask('Mobile OS', combobox, ('ios':'iOS';'and':'Android'), 'Android')}* ▪ *${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 *${xpath_eval()}*. **Note:** The *'default'* parameter specifies the default-selected value and can match either a key or a value.
	Example: ▪ *${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 || '":"' || $pair?2 || '";')}), 'ios')}*
radio	**Format:** *${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*.

Parameter	
	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. ▪ *${ask('Operating System', radio, (${xpath_eval(for $pair in (['win', 'Microsoft Windows'], ['macos', 'macOS'], ['lnx', 'Linux/UNIX']) return '"' || $pair?1 || '":"' || $pair?2 || '";')}), 'ios')}*
textarea	**Format:** *${ask('message', textarea, 'default')}*
	Description: The input is a text area that requires no special handling.
	Example: ▪ *${ask('Hello world!', textarea, 'Hello again!')}* - The dialog box would have a *Hello world!* message displayed and the value of *Hello again!* would be displayed in the text area.

Here is an example of a complex XPath expression for constructing a radio button set:

```
${ask('Bundle for keydefs', radio, (${xpath_eval(let $family := 'utah', $bundles := doc(environment-variable('HOME')
 || '/git/dita-author-tools/resource/properties/bundle-data.xml')/*/family[@name eq $family]/*, $pairs :=
$bundles ! concat('''',./@bundle-code, ''','':',  ''',./@map-filename,''')  return string-join($pairs, ';'))}),
@bundle-code)}
```

- **${caret}** - The position where the cursor is located. This variable can be used in a code template, in **Author** mode operations, or in a **selection** *plugin*.
- **${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 200)*.
- **${cp}** - Current page number. Used to display the current page number on each printed page in the **Editor / Print** Preferences page.
- **${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).
- **${dita.dir.url}** - A special local contextual editor variable that gets expanded only in the **Libraries** dialog box that is accessible from the **Advanced** tab of DITA transformation scenarios. The **Libraries** dialog box allows you to specify additional libraries (*JAR (on page 2306)* files or additional class paths) to be used by the transformer. This *${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 Developer 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 96)* setting from the **Document Type** configuration dialog box *(on page 95)*. 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 Developer options.
  - Additional *frameworks* added in the **Locations** preferences page *(on page 94)*.
  - *Frameworks* installed using the add-ons support.
  - *Frameworks* found in the main *framework* location *(on page 94)* (**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.
- **${id}** - Application-level unique identifier. It is a short sequence of 10-12 letters and digits that is not guaranteed to be universally unique.
- **${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)}
  ```

- **${oxygenHome}** - Oxygen XML Developer installation folder as URL.c
- **${oxygenInstallDir}** - Oxygen XML Developer 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 semi-colons (;).
- **${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)}')}
  ```

Custom Editor Variables

An editor variable *(on page 252)* 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 229)*. To create a custom editor variable, follow these steps:

1. Open the **Preferences** dialog box **(Options > Preferences)** *(on page 79)* 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 252)*

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 269)*.

com.oxygenxml.disable.http.protocol.handlers

- **Allowed Values:** `true` or `false`
- **Default Value:** `false`
- **Purpose:** By default, Oxygen XML Developer 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 Developer 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 Developer 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 Developer 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 238)*.

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 242)*.

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 94)*.

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 192)*.

com.oxygenxml.additional.classpath

- **Allowed Values:** A list of *JAR (on page 2306)*-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.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 Developer saves empty file content over shared network drives.

com.oxygenxml.format.indent.files.parallel

- **Allowed Values:** `true` or `false`
- **Default Value:** `true`
- **Purpose:** By default, when using the **Format and Indent Files** action from the **Project** view, up to four parallel threads are used by the operation, speeding up the processing when formatting very large or a large amount of documents. If the system property is set to `false`, only one thread will be used for the formatter.

Related information

Setting a System Property *(on page 269)*

Localizing the User Interface

Oxygen XML Developer 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 Developer 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 Developer support team and ask them to send you the `translation.xml` sample file.
3. Open the `translation.xml` file in Oxygen XML Developer. 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:

```
<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:

```
<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 79)*, go to **Global**, and select the **Other language** option *(on page 81)*. Browse for the `translation.xml` file.
7. Restart the application.

Adding New Languages to the Interface

Oxygen XML Developer 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 81)*.

Setting a Java Virtual Machine Parameter when Launching Oxygen XML Developer

You can set Java Virtual Machine parameters (for example, if you want to increase the maximum amount of memory available) for the Oxygen XML Developer application launchers *(on page 267)* or command-line scripts *(on page 270)*. You can also create a custom startup parameters file *(on page 270)*.

Setting Parameters for the Application Launchers

Increasing the Amount of Memory that Oxygen XML Developer Uses on Windows and Linux

For Windows and Linux installations of Oxygen XML Developer, 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 Developer application on Windows:

1. Browse the installation directory of Oxygen XML Developer.
2. Locate the **-Xmx** parameter in the `oxygenDeveloper28.1.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 Developer with the *oxygen* launcher (or with the desktop shortcut). If you use the command-line script (`<oxygen.bat>` or `<oxygen.sh>`), make sure you use the following procedure instead: Setting Parameters in the Command-Line Scripts *(on page 270)*.

   Tip:
   By default, the maximum memory available to the application is about a quarter of the internal memory available on the machine. It is recommended to not use more than half of your existing physical RAM.

3. Restart Oxygen XML Developer. 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 Developer 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 Developer Uses on macOS

To increase the memory available to Oxygen XML Developer on macOS:

1. Create a file named `vmoptions.txt`.
2. Add the **-Xmx** argument (or other Java VM arguments), one per line, and do not add extra new lines at the beginning or end of the file. For example:

   ```
   -Xmx4g

   -Dcom.oxygenxml.editor.plugins.dir="$OXYGEN_HOME/plugins"
   ```

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 for the main application launcher (i.e. `Oxygen XML Developer.app/ContentsOxygen XML Developer.app/Contents`). To show the `Contents` folder for the application launcher, right-click (or **Command+Single-Click**) the Oxygen XML Developer icon in **Finder**, and choose **Show Package Contents**.

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 270)* 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 268)*. Add each system property (or Java VM argument) on a separate line. For example:

  ```
  -DpropertyName1=value1
  -DpropertyName2=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 270)*.



Note:
You can also set a system property through a parameter prefixed with `-Doxy` in the command line used to start the application:

```
oxygen25.1.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 Developer system properties, go to Custom System Properties *(on page 261)*.

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 269)*:

```
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 Developer with a command-line script (`oxygenDeveloper.bat`/`oxygenDeveloper.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 2 GB in **Windows**, add the **-Xmx** parameter to the last line of the `.bat` file like this:

```
%OXYGEN_JAVA% -Xmx2g -Dsun.java2d.noddraw=true ...
```

On **macOS**/**Linux**, add the **-Xmx** parameter (followed by a '\') to a new line just above the `ro.sync.exml.Oxygen\` line (at approximately line 100) in the `.bat` file like this:

```
-cp "$CP"\
 -Xmx2g\
 ro.sync.exml.Oxygen\
 ..
```

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:

```
-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:

```
-Xmx2g
```

To be recognized and included, these custom startup parameter files must be saved in the installation directory of Oxygen XML Developer.

How to Increase the Amount of Available Memory

Determining how to increase the amount of memory that is allocated to Oxygen XML Developer depends on how you launch the application.

- **Windows/Linux Application Launcher** - If you start Oxygen XML Developer using the default startup launcher that was created during a Windows or Linux installation, see Increasing the Amount of Memory that Oxygen XML Developer Uses on Windows and Linux *(on page 267)*.
- **macOS Application Launcher** - If you start Oxygen XML Developer using the default startup launcher that was created during a macOS installation, see Increasing the Amount of Memory that Oxygen XML Developer Uses on macOS *(on page 268)*.
- **Command-Line Script** - If you start Oxygen XML Developer using a command-line script, see Setting Parameters in the Command-Line Scripts *(on page 270)*.
- **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 270)*.

5.

Perspectives

An Oxygen XML Developer *perspective (on page 2307)* is a layout geared towards a specific use. The Oxygen XML Developer 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 Developer. You can change the *perspective* by selecting the respective icon () in the top-right corner of Oxygen XML Developer or by selecting the *perspective* from the **Window > Open Perspective** menu.

Editor Perspective

The **Editor** *perspective (on page 2307)* is the most commonly used *perspective* and it is the default *perspective* when you start Oxygen XML Developer 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 Developer (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 Developer. Most of the menus are common for all types of documents. However, Oxygen XML Developer also includes some context-sensitive and *framework (on page 2305)*-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.The toolbars can be configured *(on page 291)* 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 Developer includes a large variety of *dockable (on page 2304)* 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 *(on page 285)* 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 334)* - Enables the definition of projects and logical management of the documents they contain.
- **Open/Find Resource** view *(on page 354)* - Designed to offer advanced search capabilities in various scopes.
- **Outline** view *(on page 465)* - 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 474)* - Displays the messages generated as a result of user actions such as validations *(on page 514)*, transformation scenarios *(on page 1070)*, spell checking in multiple files *(on page 389)*, search operations, and others. Each message is a link to the location related to the event that triggered the message.
- **Attributes** view *(on page 468)* - 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.
- **Model** view *(on page 470)* - Presents the currently edited element structure model and additional documentation as defined in the schema.
- **Elements view** *(on page 472)* - Presents a list of all defined elements that you can insert at the current cursor position according to the document's schema.
- **Entities** view *(on page 472)* - Displays a list with all entities declared in the current document as well as built-in ones.
- **Transformation Scenarios** view *(on page 1217)* - Displays a list with all currently configured transformation scenarios.
- **XPath/XQuery Builder** view *(on page 1729)* - Displays the results from running an XPath expression.
- **WSDL SOAP Analyzer** view *(on page 813)* - Provides a tool that helps you test if the messages defined in a Web Service Descriptor (WSDL) are accepted by a Web Services server.

Related information

Editing Supported Document Types *(on page 441)*

Editing Modes *(on page 279)*

Configuring the Layout of the Views and Editors *(on page 285)*

XSLT Debugger Perspective

The **XSLT Debugger** *perspective (on page 2307)* 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 188)* 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 279)* 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 291)* 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 1832)*

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 1832)* 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).

Resources

For more information about the XSLT debugging capabilities in Oxygen XML Developer, watch our video demonstration:

https://www.youtube.com/embed/m9d8c4V-LJw

Related information
Debugging XSLT Stylesheets and XQuery Documents *(on page 1826)*
XQuery Debugger Perspective *(on page 275)*

XQuery Debugger Perspective

The **XQuery Debugger** *perspective (on page 2307)* 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 188)* 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 279)* 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 291)* 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 1832)*

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 1832)* 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).

Resources

For more information about the XQuery debugging capabilities in Oxygen XML Developer, watch our video demonstration:

https://www.youtube.com/embed/o5_M2kbyipU

Related information
Debugging XSLT Stylesheets and XQuery Documents *(on page 1826)*
XSLT Debugger Perspective *(on page 274)*

Database Perspective

The **Database** *perspective (on page 2307)* allows you to manage databases. To switch the focus to this *perspective*, select the  **Database** button in the top-right corner of Oxygen XML Developer 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 Developer 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)
- 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 Developer. For a detailed feature matrix that compares the Academic, Professional, and Enterprise editions of Oxygen XML Developer go to the Oxygen XML Developer 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 1744)*.
- 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 Developer 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 1747)*.

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 Developer editor panel.
- Handling the values from **XML Type** columns as XML instance documents that can be opened and edited in an Oxygen XML Developer 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 291)* 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 1742)*

Provides browsing support for the configured connections.

Table Explorer View *(on page 1744)*

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 1742)*
Data Source Explorer View *(on page 1742)*
Table Explorer View *(on page 1744)*

6.
Editing Modes

The main editing area in Oxygen XML Developer 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 Developer offers the following editing modes:

- **Text** *(on page 279)* - This mode presents the source of the document.
- **Grid** *(on page 279)* - This mode displays the document as a structured grid of nested tables.
- **Design** *(on page 280)* - 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 126)* is selected in the **Edit Modes** preferences page, then the edit mode specified in the **Document Type** configuration dialog box *(on page 95)* 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 126)* is not selected, then the edit mode specified in the table in the **Edit Modes** preferences page *(on page 126)* is used when that particular type of document is initially opened.

Text Editing Mode

The **Text** mode editor in Oxygen XML Developer 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 Developer also includes specialized editing actions, a powerful *Content Completion Assistant*, 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 442)*.

Related information
Editing XML Documents in Text Mode *(on page 442)*

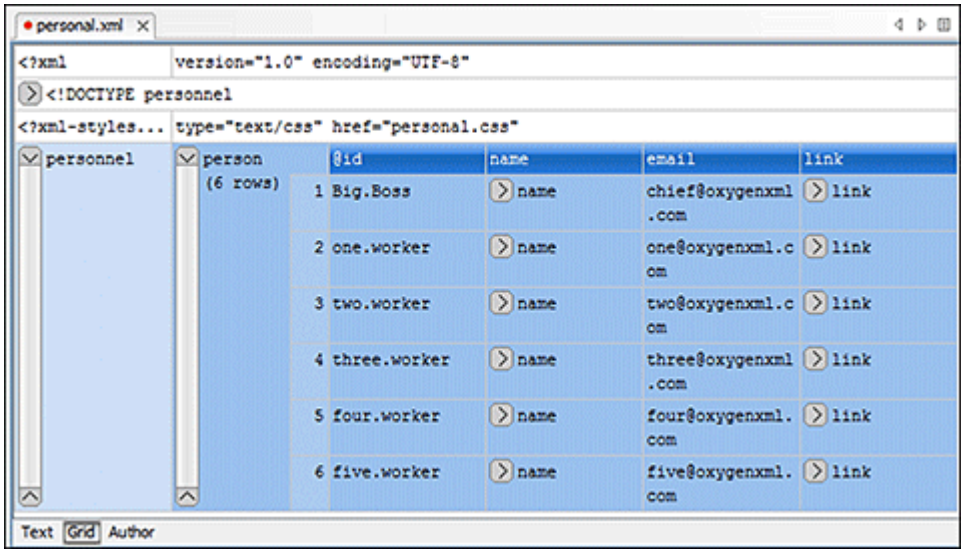
Grid Editing Mode

The Oxygen XML Developer **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 46. 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 505)*.

Related information

Editing XML Documents in Grid Mode *(on page 505)*

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 Developer 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 and **Facets** view) to further enhance the diagram editor, various contextual menu actions, validation support, and much more.

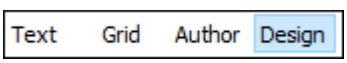
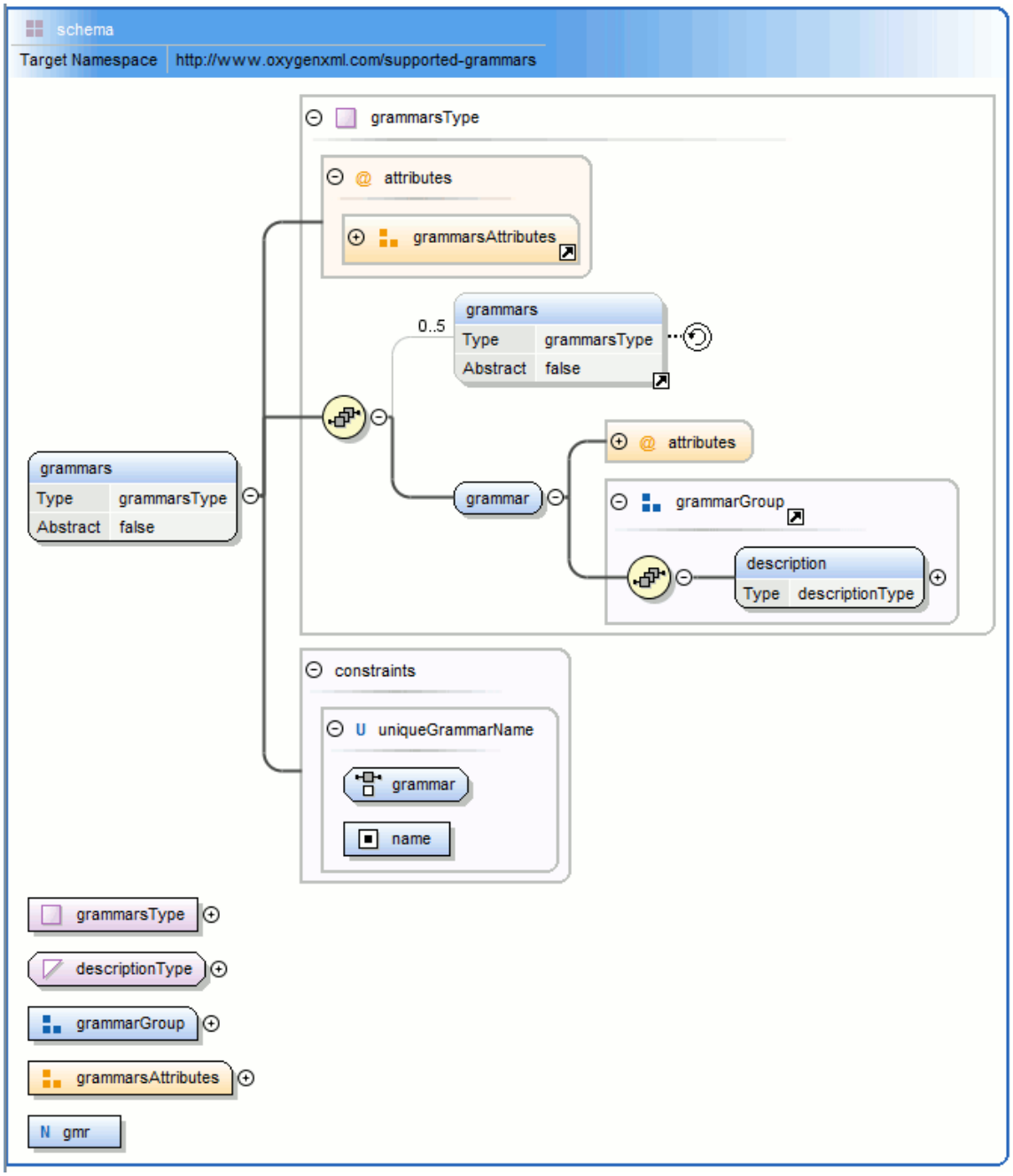
To switch to this mode, select **Design** at the bottom of the editing area. 

Figure 47. XML Schema Diagram



XML Schema Resources

For more information about designing and editing XML schemas, and all the details about the features that are available in the Oxygen XML Developer **Design** mode for XML schema documents, see the **Editing XML Schemas** section *(on page 689)* and the **XML Schema Design Mode (XML Schema Diagram Editor)** subsection *(on page 689)*.

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 Developer **Design** mode for JSON schema documents, see the **Editing JSON**

Schemas section *(on page 893)* and the **JSON Schema Design Mode (JSON Schema Diagram Editor)** subsection *(on page 894)*.

7.

Working With Documents

Oxygen XML Developer includes a variety of general features that can be used when working with most types of documents. More specialized features are available when working with specific types of documents, such as the various types of XML documents, CSS *(on page 817)*, JavaScript *(on page 940)*, Markdown *(on page 1016)*, and more. For details about those specialized features for specific types of documents, see Editing Supported Document Types *(on page 441)*.

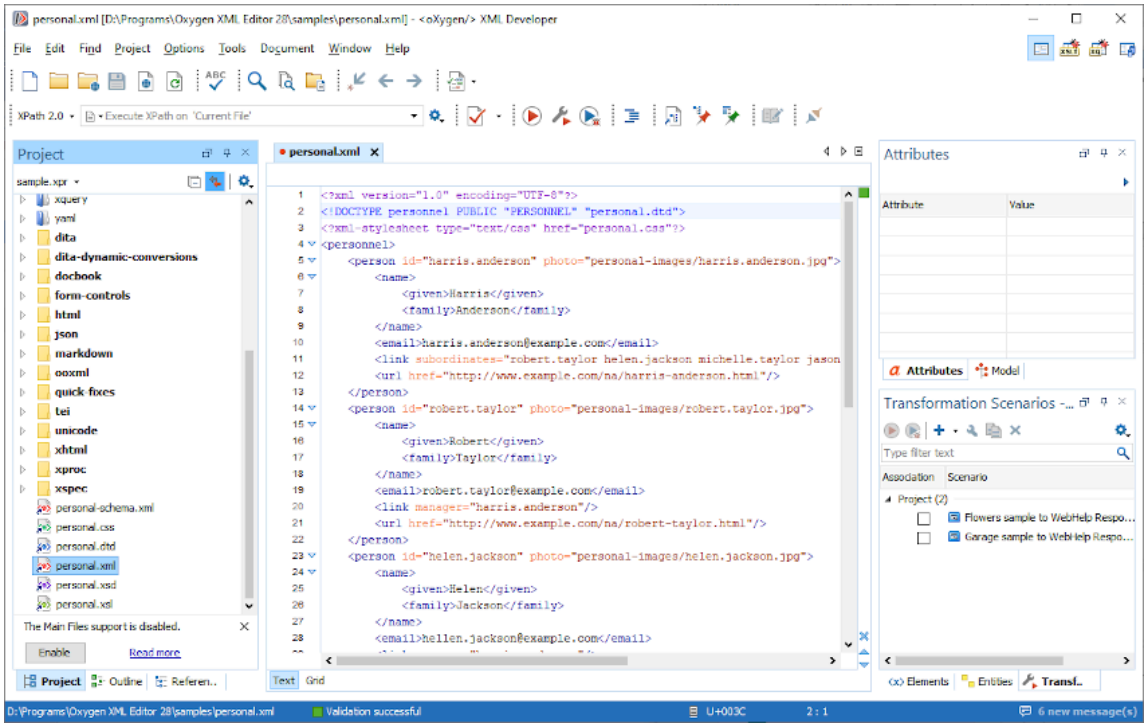
This chapter includes information about how to create and work with documents, working with projects, and various general editing features. This chapter also includes information about some of the special tools that are included in Oxygen XML Developer, such as the file and directory comparison tools *(on page 401)*.

Getting Familiar with the Interface

Oxygen XML Developer includes several *perspectives (on page 2307)* and editing modes *(on page 279)* 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 285)*, and you can customize the toolbars *(on page 291)*.

Figure 48. Clickable Image Map of the Oxygen XML Developer Interface



1. Menus *(on page 284)*
2. Toolbars *(on page 284)*
3. Left-side Helper Views *(on page 284)*
4. Right-side Helper Views *(on page 284)*
5. Editor Pane *(on page 284)*
6. Perspectives *(on page 284)*
7. Status Bar *(on page 285)*
8. Editing Modes *(on page 279)*

Regardless of the *perspective (on page 2307)* or editing mode 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 Developer. Most of the menus are common for all types of documents, but Oxygen XML Developer 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 291)* to suit your specific needs.

Helper Views

Oxygen XML Developer includes a large variety of *dockable (on page 2304)* 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 285)* according to your preferences.

Editor Pane

The main editing area in the center of the application. Each editing mode 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 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 325)*.

Perspectives

Oxygen XML Developer includes several different *perspectives (on page 272)* 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 Developer for the first time. Oxygen XML Developer 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 391)* 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, the current line and character position is displayed.
- If the **Check for notifications** option *(on page 81)* 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 Developer website, the announcement of upcoming webinars and conferences where the Oxygen XML Developer 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 83)* 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 Developer are *dockable (on page 2304)* 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 291)*.

To open a view, select it from the **Window > Show View** menu. You can hide a view by closing it with the × 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 Developer window. Once you create a layout that suits your needs, you can save it from **Window > Export Layout**. Oxygen XML Developer 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**.

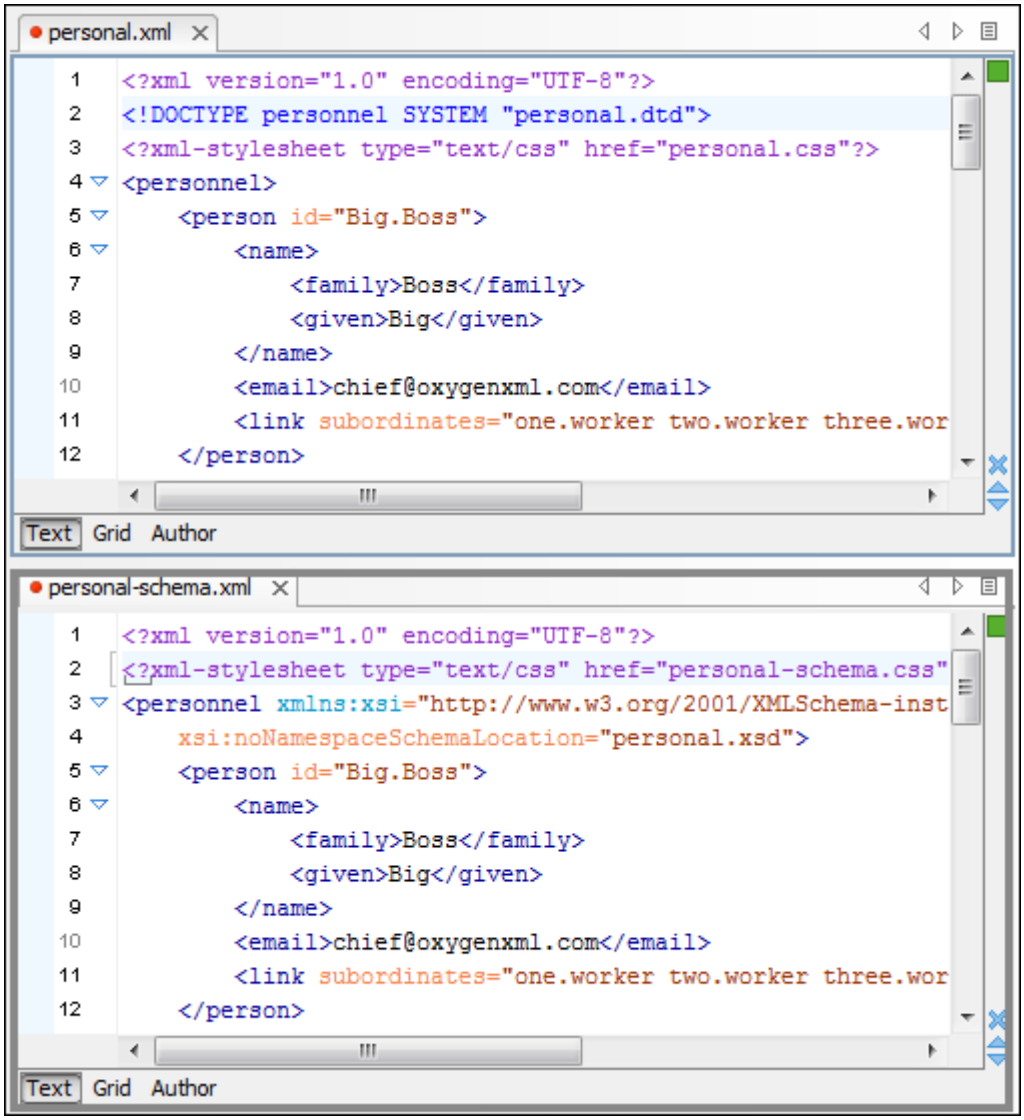
Note:
The **Load Layout** menu lets you select between the default layout, a predefined layout, or a custom layout. The changes you make using the **Load Layout** menu are also reflected in the **Application Layout** preferences page.

The changes you make to any layout are preserved between working sessions. The predefined layout files are saved in the preferences directory *(on page 80)* of Oxygen XML Developer.

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 49. Drag and Drop Editors



Hide or Float Views

Hide

To gain more editing space in the Oxygen XML Developer window, click **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 Developer 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 Developer 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 Developer 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 82)*.

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 + Option + 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 83)*.

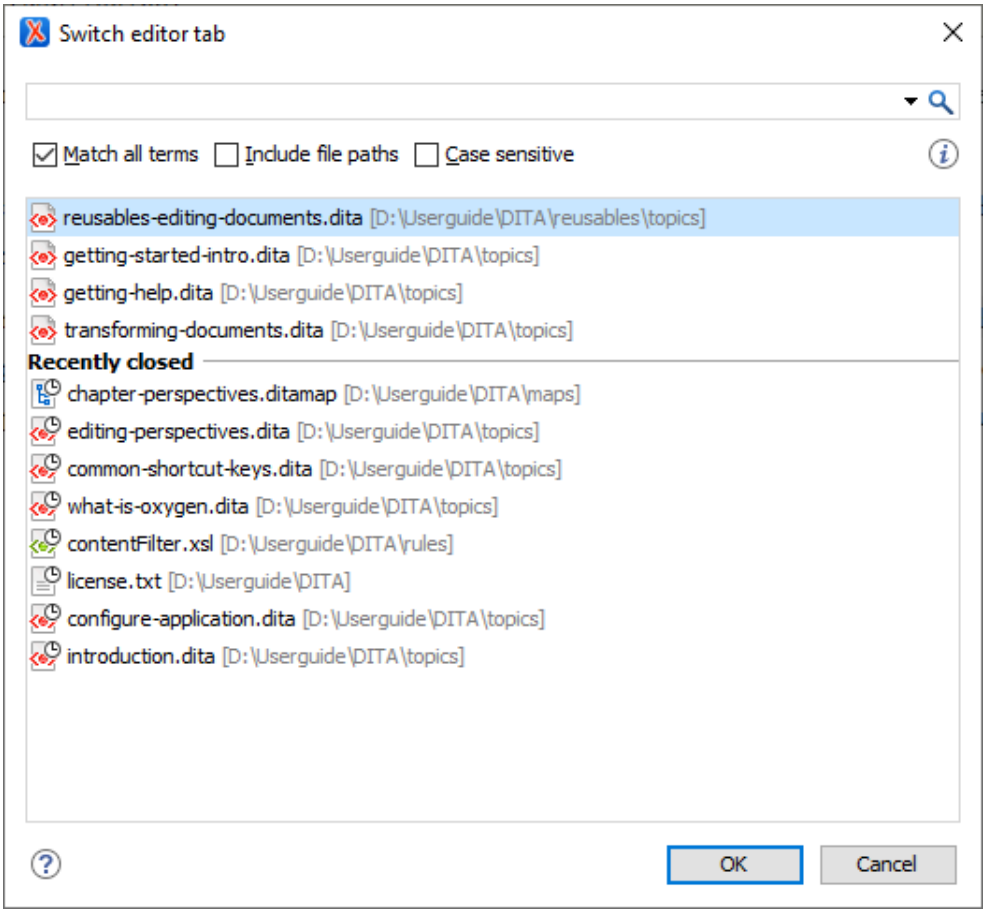
Ctrl + Shift + Tab (Command + Shift + Tab on macOS) [NOTE: Ctrl + Page Up (Ctrl + Option + Left Arrow on macOS) does the same]

Switches to the previous open tab in the order specified in the **Order of switching between editor tabs** option *(on page 83)*.

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.

Figure 50. Switch Editor Tab Dialog Box



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.

Resources

For more information about configuring the interface of Oxygen XML Developer, 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 291)*

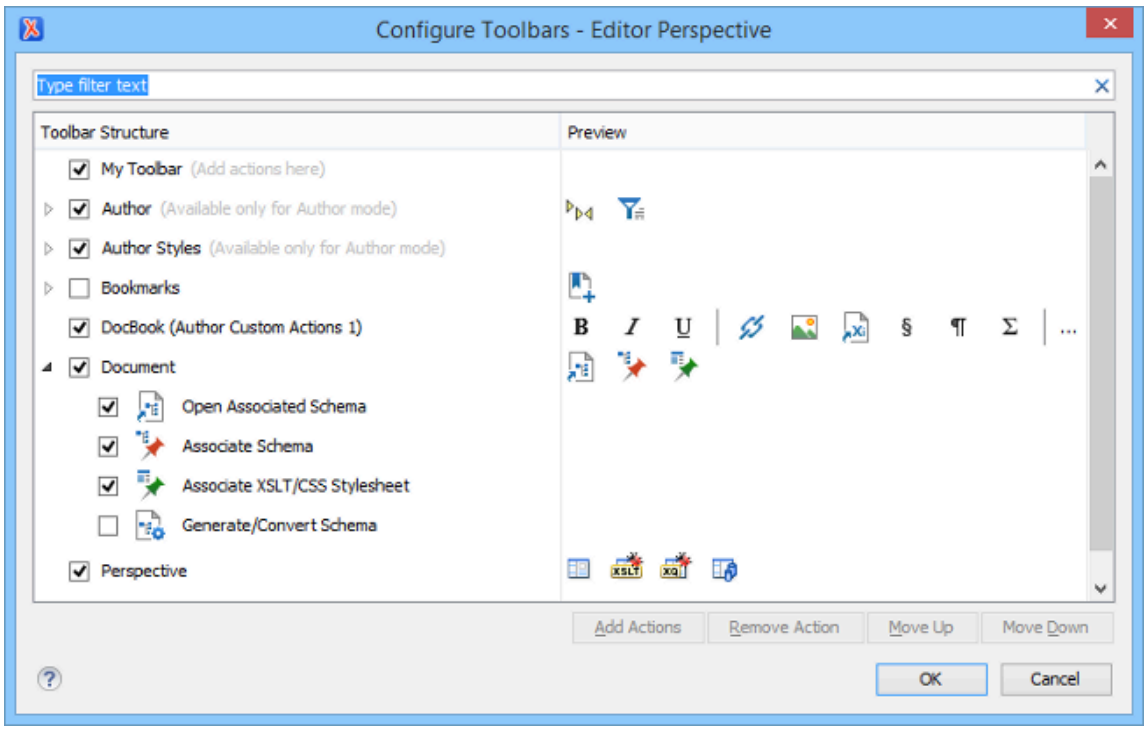
Configuring Toolbars

You can configure the toolbars in Oxygen XML Developer to personalize the interface for your specific needs. You can choose which toolbars to show or hide in the current editor mode (**Text**, **Design**, or **Grid**) and in the current *perspective (on page 2307)* (**Editor**, **XSLT Debugger**, **XQuery Debugger**, or **Databse**). 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 51. 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 2307)* 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 2305)* or 3rd party *plugins (on page 2307)*.

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 Developer.

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 285)*

Frequently Used Shortcut Keys

Oxygen XML Developer 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 223)*.

For information about how to assign or configure shortcut keys, see How to Assign a Shortcut Key or Edit an Existing Shortcut *(on page 225)*.

Table 3. Frequently Used Shortcut Keys in Oxygen XML Developer

Action	Windows/Linux Shortcut Keys	macOS Shortcut Keys	Description of Default Assigned Action
Attribute Editor	**Alt + Enter**	**Option + Enter**	Opens the in-place attribute editor
Beginning	**Ctrl + Home**	**Command + Home**	Navigates to the beginning of the document

Table 3. Frequently Used Shortcut Keys in Oxygen XML Developer (continued)

Action	Windows/Linux Shortcut Keys	macOS Shortcut Keys	Description of Default Assigned Action
Check Spelling	**F7**	**F7**	Opens the spell checking dialog box
Check Well-Formedness	**Ctrl + Shift + W**	**Command + Shift + W**	Check well-formedness of current document
Configure Transformation	**Ctrl + Shift + C**	**Command + Shift + C**	Opens the Configure Transformation Scenario dialog box
Content Completion / New Line	**Enter**	**Enter**	• **Author** mode - Opens the content completion window • **Text** mode - Moves cursor to the next line
Content Completion (**Text** Mode)	**Ctrl + Space**	**Command + Space**	Opens the content completion window in **Text** mode
Create Bookmark #	**Ctrl + Shift + 1-9**	**Command + Shift + 1-9**	Create bookmarks numbered 1 through 9
Create Next Bookmark	**F9**	**F9**	Create bookmark numbered whatever is next in sequence
Delete Next Word	**Ctrl + Delete**	**Command + Delete**	Deletes the next word or whitespace
Delete Previous Word	**Ctrl + Backspace**	**Command + Backspace**	Deletes the previous word or whitespace
Delete Tags	**Alt + Shift + X**	**Command + Option + X**	Deletes the start and end tag of the current element
Duplicate Lines Up (**Text** Mode)	**Ctrl + Shift + UpArrow**	**Option + Shift + UpArrow**	Duplicates the selected lines (or current line) and inserts it above the current selection/line

Table 3. Frequently Used Shortcut Keys in Oxygen XML Developer (continued)

Action	Windows/Linux Shortcut Keys	macOS Shortcut Keys	Description of Default Assigned Action
Duplicate Lines Down (**Text** Mode)	**Ctrl + Shift + DownArrow**	**Option + Shift + DownArrow**	Duplicates the selected lines (or current line) and inserts it below the current selection or line
End	**Ctrl + End**	**Command + End**	Navigates to the end of the document
Exit	**Ctrl + Q**	**Command + Q**	Exit the application
Find	**Ctrl + F**	**Command + F**	Opens Find/Replace dialog box
Find Next	**F3**	**Command + G**	Finds next occurrence of the last searched term
Find Previous	**Shift + F3**	**Command + Shift + G**	Finds previous occurrence of the last searched term
Go To Bookmark	**Ctrl + 1-9**	**Command + 1-9**	Go to specific bookmark
Go To Definition	**Shift + Ctrl + Enter**	**Shift + Command + Enter**	Go to the definition of the selected item in the associated schema.
Help	**F1**	**F1**	Opens help documentation
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 + Option + Comma**	Moves the current file tab one position to the left
Move Tab Right	**Ctrl + Alt + Period**	**Ctrl + Option + Period**	Moves the current file tab one position to the right
Move Node Down (**Author**)	**Alt + DownArrow**	**Option + DownArrow**	Moves the selected XML node down in **Author** mode

Table 3. Frequently Used Shortcut Keys in Oxygen XML Developer (continued)

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

Table 3. Frequently Used Shortcut Keys in Oxygen XML Developer (continued)

Action	Windows/Linux Shortcut Keys	macOS Shortcut Keys	Description of Default Assigned Action
Reset Zoom	**Ctrl + NumPad0**	**Command + NumPad0**	Resets zoom (default font size)
Save	**Ctrl + S**	**Command + S**	Saves current document
Save All	**Ctrl + Shift + S**	**Command + Shift + S**	Saves all open files
Scroll Down	**Ctrl + DownArrow**	**Command + DownArrow**	Scrolls the editor down
Scroll Up	**Ctrl + UpArrow**	**Command + Up Arrow**	Scrolls the editor up
Select Content of Element	**Alt + [Mouse Triple Click]**	**Option + [Mouse Triple Click]**	Selects the content of an element in **Author** mode.
Shift Left	**Shift + Tab**	**Shift + Tab**	• **Author** mode - Moves the cursor to the previous XML node • **Text** mode - Shifts content to the left
Shift Right	**Tab**	**Tab**	• **Author** mode - Moves cursor to the next XML node • **Text** mode - Shifts content to the right
Split Element	**Alt + Shift + D**	**Ctrl + Option + D**	Splits the element the cursor position
Surround With	**Ctrl + E**	**Command + E**	Surrounds selected content with specified tag
Switch Tabs	**Ctrl + Tab** / **Ctrl + Shift + Tab**	**Command + Tab** / **Command + Shift + Tab**	Switches between open tabs
Transform	**Ctrl + Shift + T**	**Command + Shift + T**	Opens a dialog box for selecting a transformation scenario

Table 3. Frequently Used Shortcut Keys in Oxygen XML Developer (continued)

Action	Windows/Linux Shortcut Keys	macOS Shortcut Keys	Description of Default Assigned Action
Underline / Open URL	**Ctrl + U**	**Command + U**	• Underlines selected content (in the main editor) • Opens the URL (when focus is outside the main editor)
Undo	**Ctrl + Z**	**Command + Z**	Undo last editing action
Zoom In	**Ctrl + NumPad+**	**Command + NumPad+**	Zooms in (increase font size)
Zoom Out	**Ctrl + NumPad-**	**Command + NumPad-**	Zooms out (decrease font size)

Troubleshooting:
If you encounter problems with keyboard shortcuts not working as expected, see Keyboard Shortcuts Result in Unexpected Behavior *(on page 2226)* or Keyboard Shortcuts Do Not Work At All *(on page 2225)*.

Table 4. Supported Emacs Key Bindings for macOS

Action	macOS Shortcut Key	Editing Mode Availability
Move cursor to start of line	**Ctrl + A**	**Text** and **Author** mode
Move cursor to end of line	**Ctrl + E**	**Text** and **Author** mode
Move cursor up one line	**Ctrl + P**	**Text** and **Author** mode
Move cursor down one line	**Ctrl + N**	**Text** and **Author** mode
Move cursor left one character	**Ctrl + B**	**Text** and **Author** mode
Move cursor right one character	**Ctrl + F**	**Text** and **Author** mode
Delete character to the right of cursor	**Ctrl + D**	**Text** and **Author** mode
Delete character to the left of cursor	**Ctrl + H**	**Text** and **Author** mode
Scroll Down	**Ctrl + V**	**Text** and **Author** mode
Delete the selection (cut text)	**Ctrl + K**	**Text** mode
Retrieve deleted text (paste)	**Ctrl + Y**	**Text** mode
Insert line break w/o moving cursor	**Ctrl + O**	**Text** mode

Creating, Opening, Saving, and Closing Documents

Oxygen XML Developer 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 Developer 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 307)* and share them with others *(on page 313)*.

To 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 Developer 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)** *(on page 79)* and go to **File Types** *(on page 226)*.

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 307)* and select them from this wizard.

> **Tip:**
> If the **Oxygen AI Positron** add-on (version 8.0 or later) is installed, an **AI prompt** input box appears in the new document wizard, where you can provide the AI with details and instructions for creating a new document from the chosen file template. You can also paste snippets of content and attach documents (text, Markdown, Word, PDF) or images by using the **Attach** button (or by using copy/paste or drag/drop). The AI processing is shown in the **AI Positron** view, where you can then continue the conversation to refine the generated content.

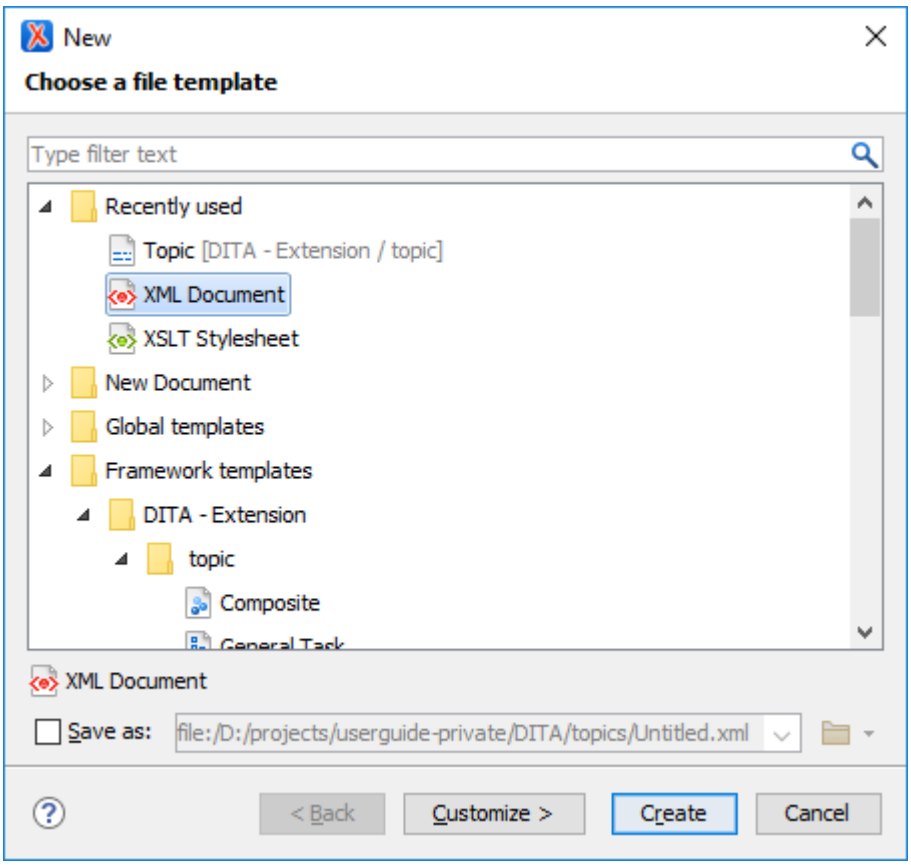
New Document Wizard

To create a new document using this wizard, follow these steps:

1. Click the **New** button on the toolbar or select **File > New**.

Result: The **New Document** wizard is displayed:

Figure 52. New Document Wizard



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 307)* in a directory and then adding that directory to the list of template directories that Oxygen XML Developer uses in the **Document Templates** preferences page *(on page 122)*. 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 309)*.

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, PowerShell, JavaScript, Java, C, C++, Batch, Shell, Properties, SQL, *XML Catalog*, PERL, JSON, and more.

Global Templates

Contains the list of built-in templates along with user-defined custom templates. You can create your own custom document templates *(on page 307)* and add them to the `templates` folder of the Oxygen XML Developer installation directory.

Framework Templates

Contains the list of templates defined in the **Document Type** configuration dialog box (**Templates** tab) *(on page 118)* 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 135)* 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 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.

4. If you want to use the default settings in the creation process, select **Create** at the bottom of the dialog box.

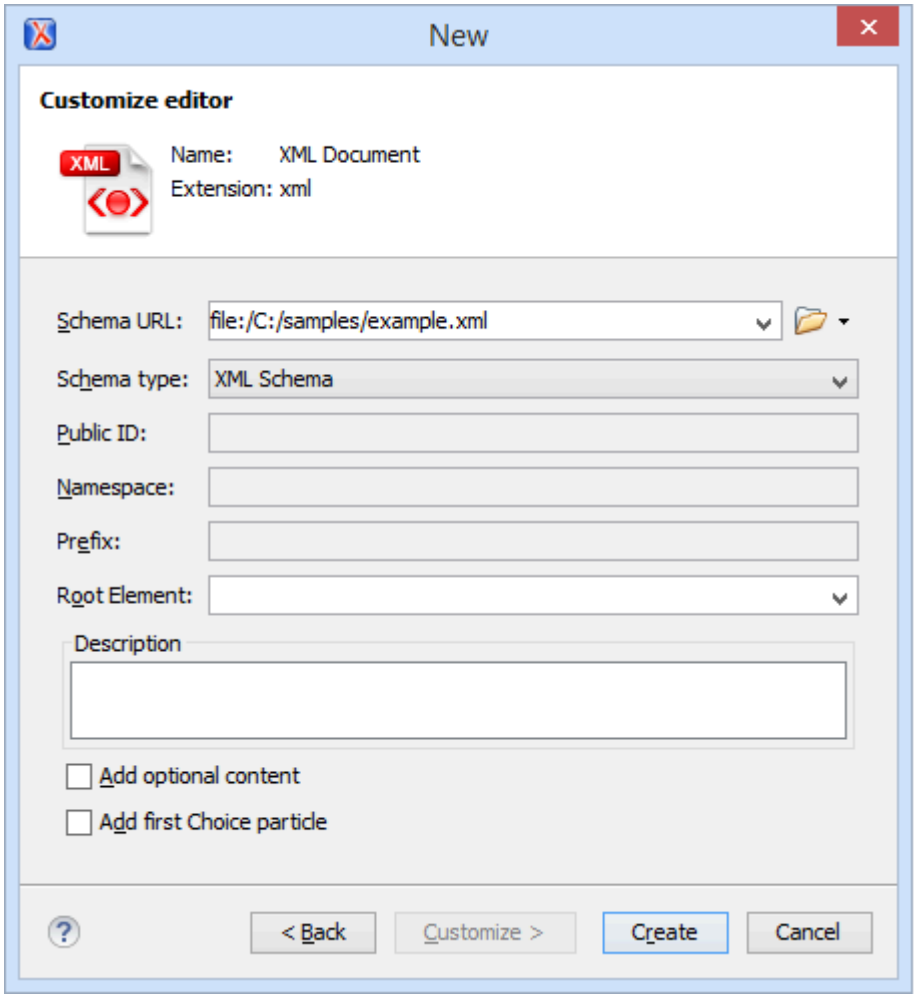
   Result: The document is created using the default settings and the new file is opened in the appropriate editor.

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 53. 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 Developer 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

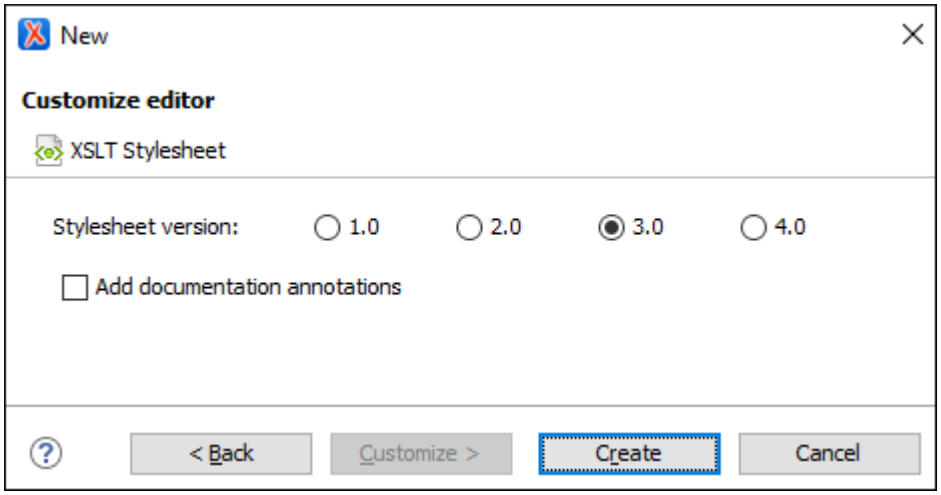
If you select this option, the elements and attributes defined in the XML Schema as optional are generated in the skeleton XML document.

Add First Choice Particle

If you select this option, Oxygen XML Developer generates the first element of an `<xs:choice>` schema element in the skeleton XML document. Oxygen XML Developer creates this document in a new editor panel when you click **OK**.

XSLT Document Configuration Page

Figure 54. 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

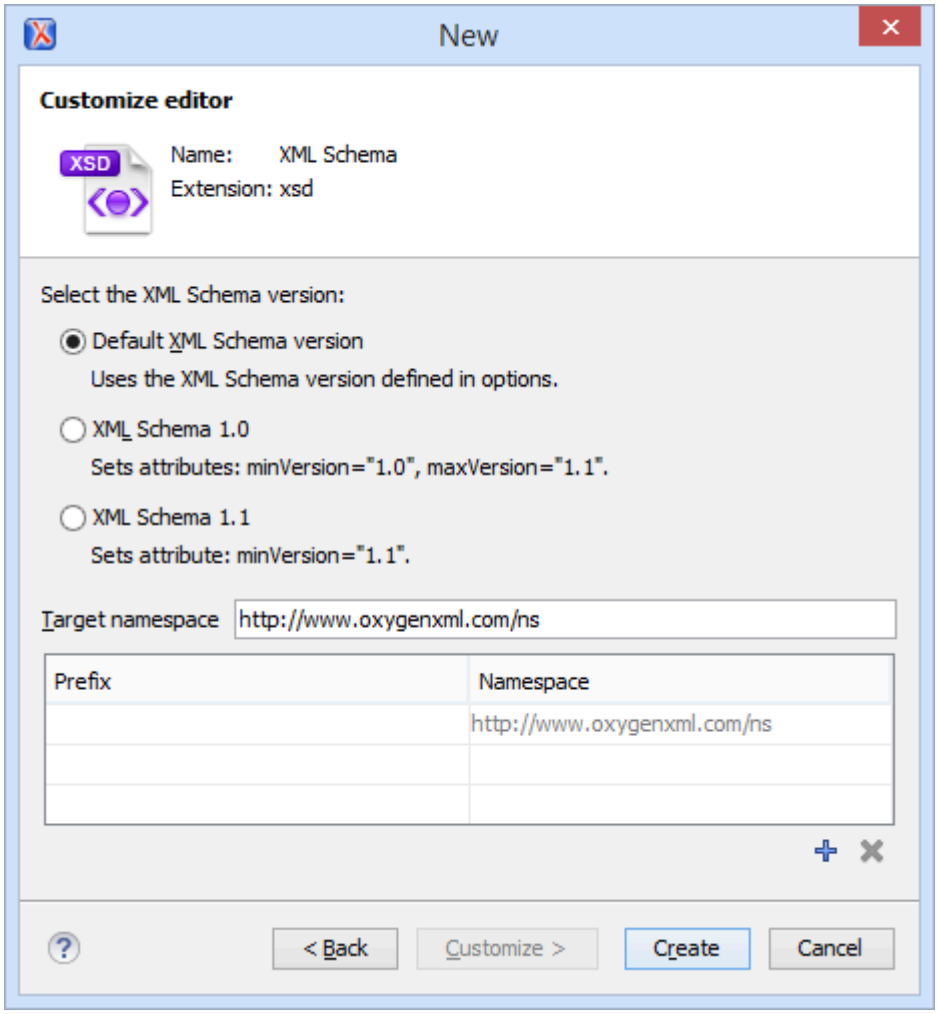
Allows you to select the **Stylesheet version** number. You can select from: `1.0`, `2.0`, `3.0`, and `4.0`.

Add documentation annotations

Select this option to generate the stylesheet annotation documentation.

XML Schema Document Configuration Page

Figure 55. 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 172)*.

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.

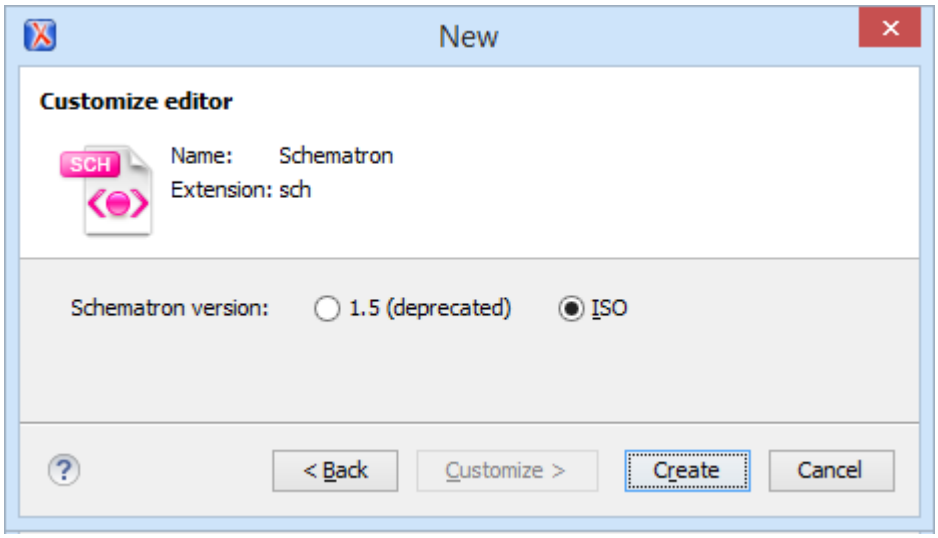
Namespace prefix declaration table

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 772)*.

Schematron Document Configuration Page

Figure 56. 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 Developer, the support for Schematron 1.5 is deprecated. It is recommended to use ISO Schematron instead.

XProc Configuration Page

Figure 57. New XProc Configuration Wizard Page



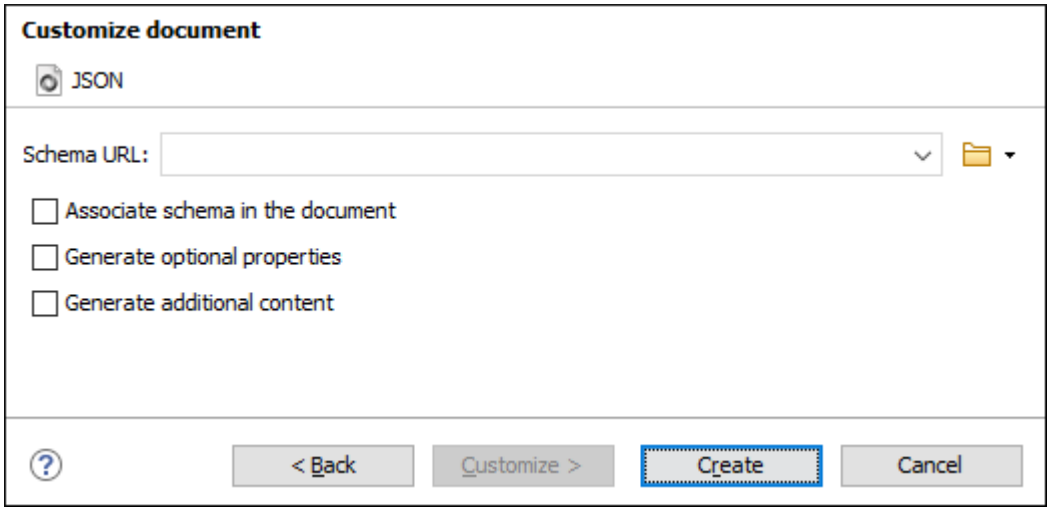
If you selected **XProc script** for the type of file you want to create and selected the **Customize** option, the configuration dialog box will include the following option:

XProc version

Specifies the XProc version. Possible options: `1.0`, `3.0` and `3.1`.

JSON Document Configuration Page

Figure 58. New JSON Configuration Wizard Page



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

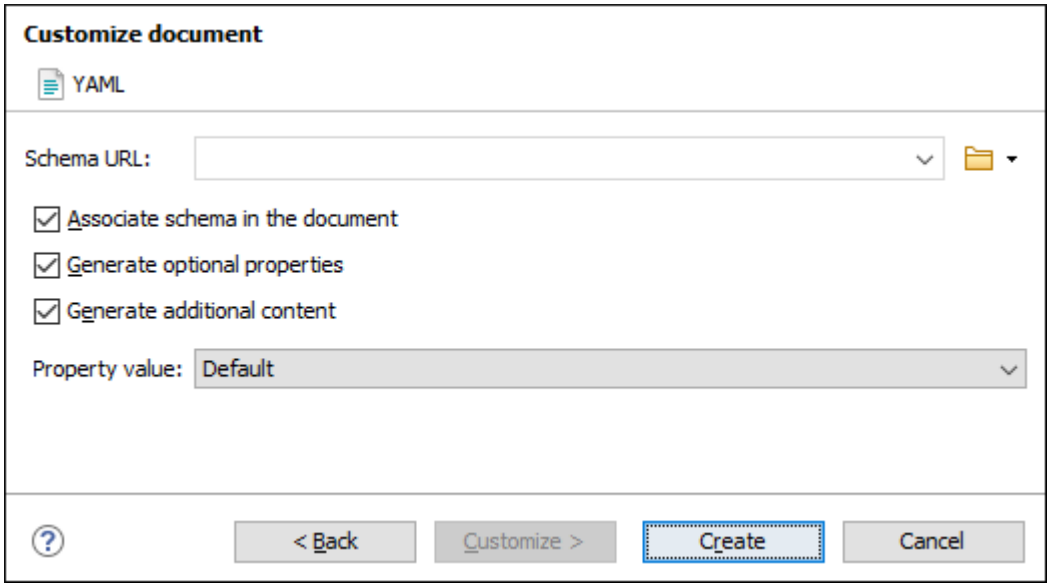
If you select this option, the JSON instance will be generated with optional properties that are defined in the JSON schema. Otherwise, only the required properties will be generated.

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).

YAML Document Configuration Page

Figure 59. New YAML Configuration Wizard Page



If you select **YAML** 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 YAML instance will be generated with the JSON schema associated directly in the document.

Generate Optional Properties

If you select this option, the YAML 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 YAML 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).

Property Value

Specifies the method of generating values for the properties. Possible values: **None**, **Default**, **Random**.

Creating New Document Templates

Oxygen XML Developer allows you to create your own custom document templates and they will appear in the **New** document wizard *(on page 299)*.

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 252)* in the template file content and they will be expanded when the files are opened. Also, see Customizing Document Templates *(on page 309)* for other template customization tips (for example, you could add placeholders or hints *(on page 312)* to assist authors).

2. Save the new document template and reference that location in Oxygen XML Developer. 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. Then open the **Document Type** configuration dialog box *(on page 95)* for that specific framework, go to the **Templates** tab *(on page 118)*, 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 Developer 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 299)*.
   - **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 122)*. This user-defined directory will appear in the **New** document wizard *(on page 299)* 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 311)* 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, but if you customize the template *(on page 309)*, you need to set the `type` property to **dita** in the corresponding properties file.

Related information
Customizing Document Templates *(on page 309)*
Sharing Custom Document Templates *(on page 313)*

Customizing Document Templates

Oxygen XML Developer 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 299)*.

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 299)*. 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:

```
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. 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 308)*), 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 334)*. The prefix or suffix is added to the name of the file in the **File name** field.

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:

      ```
      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 310)*.

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 310)*) 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 252)* 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 2305)*-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 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 299)* 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**.

Related information

Creating New Document Templates *(on page 307)*

Sharing Custom Document Templates *(on page 313)*

Sharing Custom Document Templates

Your custom document templates *(on page 307)* 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 299)*. The best way to share them is by integrating them in an extended *framework (on page 2305)* (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, if you have not already done so.
2. Create the new document template *(on page 307)*, 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 95)* for that specific framework, go to the **Templates** tab *(on page 118)*, 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.

Opening Documents

To open a document in Oxygen XML Developer, 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 81)*
- 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 334)*.

Related information

Opening Local Files at Start-up *(on page 314)*

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 79)*, 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 (`oxygenDeveloper.bat` on Windows, `oxygenDeveloper.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 315)*

Opening a Document at a Specific Location Using a Command-Line Interface

Oxygen XML Developer offers support for opening a file at a specific position using a command-line interface to transmit parameters to the Oxygen XML Developer application launching script file (`oxygenDeveloper.bat`/`oxygenDeveloper.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 Developer 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 Developer from a Windows command-line interface:

   ```
   developer.bat  file:samples/personal.xml#line=4
   developer.bat  file:samples/personal.xml#line=4column=5
   developer.bat  file:samples/personal.xml#line=4;column=5
   developer.bat  file:samples/personal.xml#char=334
   ```

2. **Simplified XPath index path**

   Oxygen XML Developer 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 Developer 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 Developer and select the third child of the root element using a Windows command-line interface:

   ```
   developer.bat  file:samples/personal.xml#element(1/3)
   ```

3. **Anchors identified by ID attribute values**

   Oxygen XML Developer will open an XML file and select the element whose `@id` attribute value is an exact match of the *anchor (on page 2303)* attached to a command-line instruction.

   Examples for Windows:

The following example shows how you can open an XML document in Oxygen XML Developer and select the element that has the `@id` set to `titleID` using a Windows command-line interface:

```
developer.bat  file:samples/personal.xml#titleID
```

Related information

Opening Local Files at Start-up *(on page 314)*

Saving Documents

You can save the document you are editing with one of the following actions:

- **File >** **Save**.
- **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 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.

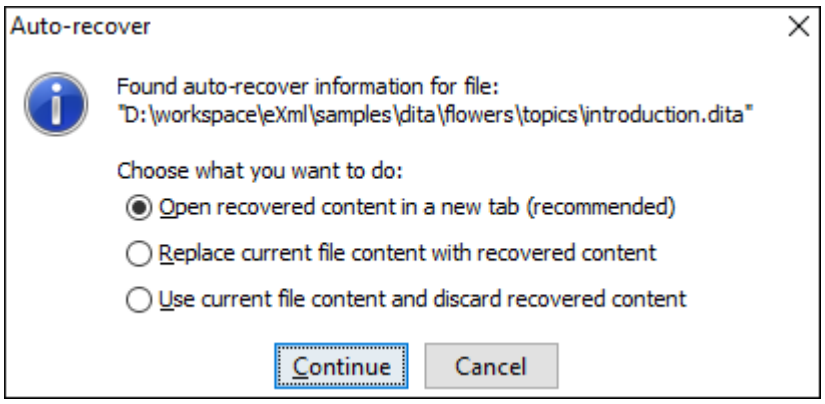
Auto Recover Documents

Oxygen XML Developer 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 135)*, and you can specify the location of the saved documents in the **Auto-recover file location** option *(on page 135)*.

In the event of an application or system crash, once you restart the application, Oxygen XML Developer 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.

Figure 60. Auto Recover Dialog Box



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 135)*.
- The *Auto Recover* feature does NOT work on files opened in the *huge file editor (on page 399)* (if you select the **Optimize loading for huge files** option when opening large documents *(on page 397)*).

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 Developer supports editing remote files, using the WebDAV, SharePoint Online for Office 365, or SFTP 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 318)*.

A WebDAV resource can be locked when it is opened in Oxygen XML Developer by selecting the **Lock WebDAV files on open** option *(on page 233)* to prevent other users to modify it concurrently on the server. If a user tries to edit a locked file, Oxygen XML Developer 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 Developer.

To avoid conflicts with other users when you edit a resource stored on a SharePoint server, you can **Check Out** the resource.

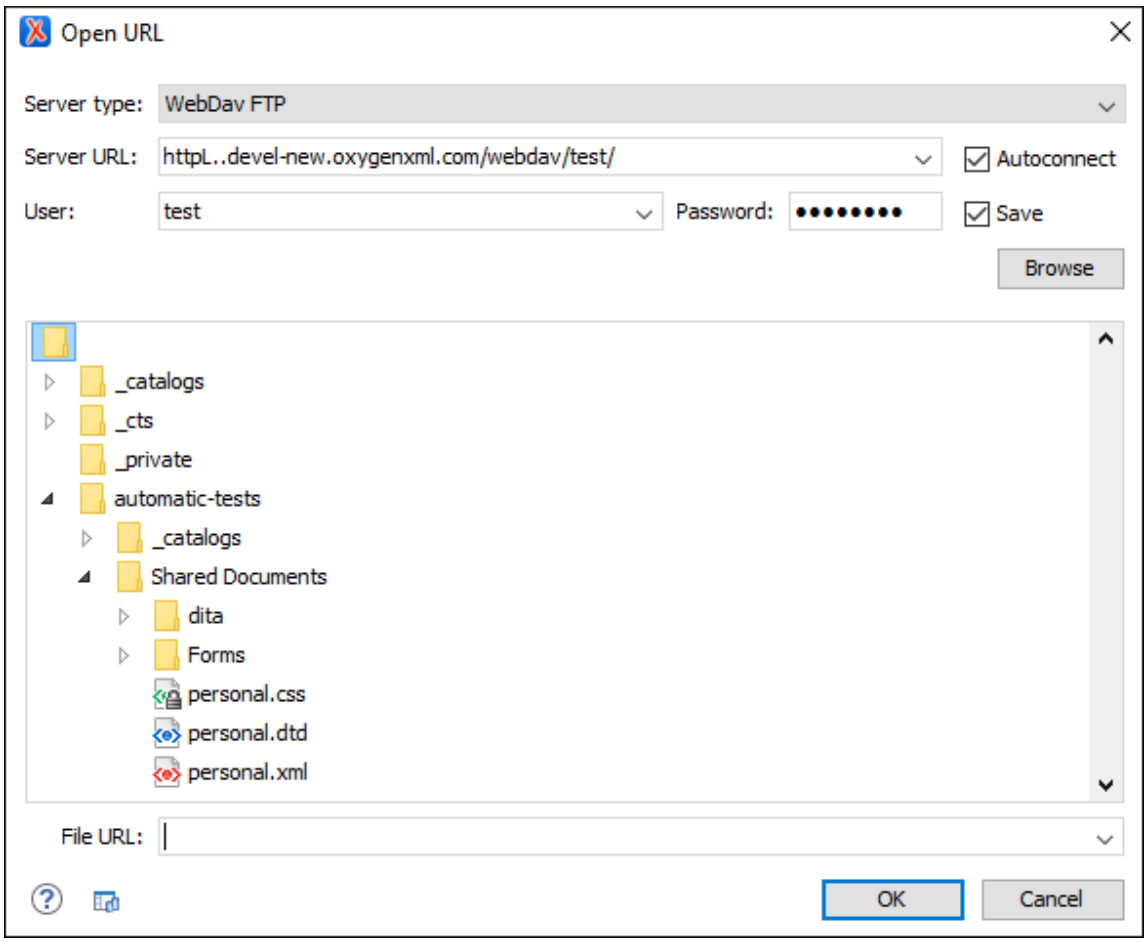
To improve the transfer speed, the content exchanged between Oxygen XML Developer and the HTTP / WebDAV server is compressed using the GZIP algorithm.

The current WebDAV Connection *(on page 1789)* details can be saved by switching to the **Database** *perspective (on page 2307)* and then you can browse and manage the connection in the **Data Source Explorer** view *(on page 1742)*.

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 61. 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/SFTP/WebDav and other servers.
- **SharePoint Online** - For SharePoint Online servers.

Server URL

Specifies the protocol (HTTP, HTTPS, or SFTP) and the host name or IP of the server.

Tip:
When specifying a URL, follow these rules:

- 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 SFTP 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.

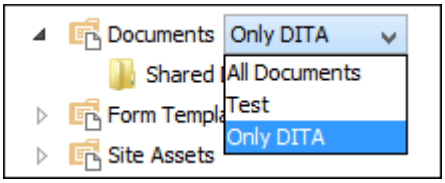
  The left side navigation area presents the SharePoint site structure in a tree-like fashion with various node types (such as *sites*, *libraries*, and *folders*).

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.

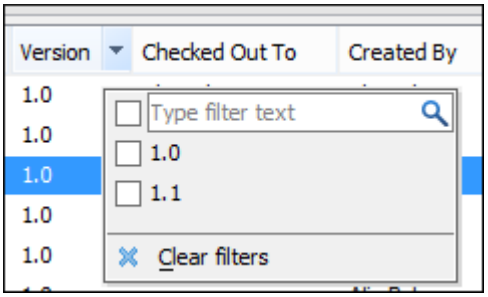
Figure 62. Drop-Down Menu to Select Which Items to Display



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 63. Column Filter



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).

This combo box also displays the current selection when the user changes selection by browsing the tree of folders and files on the server.

WebDAV over HTTPS

If you want to access a WebDAV repository across a non-secure network, Oxygen XML Developer 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 Developer 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 Developer. This means that Oxygen XML Developer 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 Developer cannot connect to an HTTPS-capable server and an error message appears stating that it is "unable to find a valid certification path to the requested target", the HTTPS server is most likely either configured to use a self-signed certificate or to use a certificate issued by an unknown authority that the *Java Runtime Environment (JRE)* used by Oxygen XML Developer does not trust.

Note:
For Windows, starting with version 26.0, by default, Oxygen XML Developer uses the trusted root certificates from the Windows certificate store instead of the JRE cacerts store. To trust a certificate, the root certificate should be imported in the Windows Trusted Root certificates store.

Tip:
To make Oxygen XML Developer accept a certificate even if it is invalid, open the **Preferences** dialog box **(Options > Preferences)** *(on page 79)*, 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 **Export** button.

g. In the resulting dialog box, for the **Save as type** option, select **DER-encoded binary, single certificate (*.der)**.
h. Save the certificate to the local file `server.der`.

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 Developer:
   a. Open a text-mode console with administrative rights.If Oxygen XML Developer 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 Developer. 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).

   > **Note:**
   > On macOS, for the distribution of Oxygen XML Developer that bundles the JRE from Oracle, the JRE uses the `.install4j/jre.bundle/Contents/Home/jre/lib/security/cacerts` path within its installation directory.

   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 2306)*. 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:
   >
   > ```
   > ..\..\bin\keytool -import -alias myalias1 -trustcacerts -file
   > server1.cer -keystore cacerts
   > ```

```
..\..\bin\keytool -import -alias myalias2 -trustcacerts -file

server2.cer -keystore cacerts
```

3. Restart Oxygen XML Developer.

Related information

HTTP(S)/WebDAV Preferences *(on page 232)*

HTTP Authentication Schemes

Oxygen XML Developer 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 (Deprecated)** *(on page 324)* - 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 (Deprecated)

Oxygen XML Developer 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 Developer obtains from the operating system.

Restriction:
This *Single sign-on* support is not available for SharePoint integrations.

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
```

Related information

Setting a Java Virtual Machine Parameter when Launching Oxygen XML Developer *(on page 267)*

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 Developer 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 82)*.

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 + Option + 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 83)*.

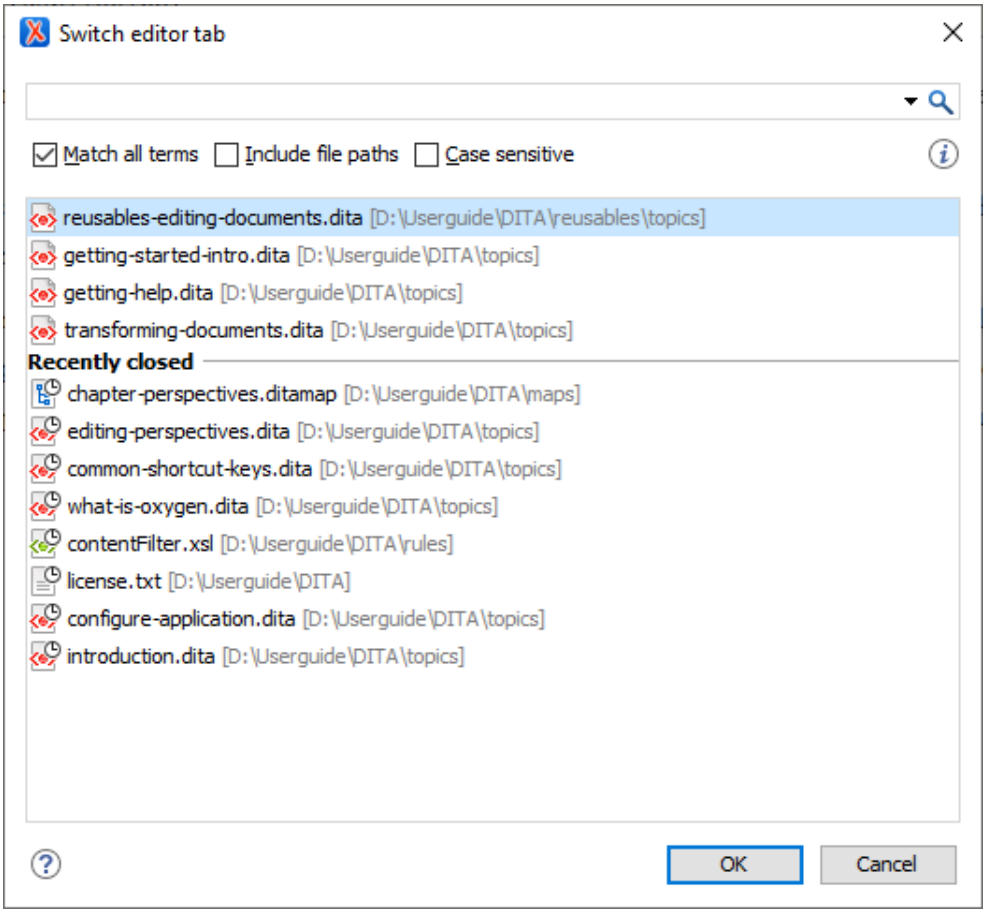
Ctrl + Shift + Tab (Command + Shift + Tab on macOS) [NOTE: Ctrl + Page Up (Ctrl + Option + Left Arrow on macOS) does the same]

Switches to the previous open tab in the order specified in the **Order of switching between editor tabs** option *(on page 83)*.

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.

Figure 64. Switch Editor Tab Dialog Box



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

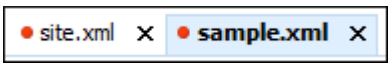
Moves the current file tab one position to the right.

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 + Option + T on macOS))

Reopens the last closed editor.

Maximize Editing Area

Collapses all the side views and spans the editing are 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

Oxygen XML Developer 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 Developer **Text** editing mode are available in this simple editor.

Types of Non-XML Files That are Supported in the Simple Text Editor

The types of files that can be created and edited in the simple text editor include:

- Java
- C++
- C
- Dockerfile
- PHP
- Perl
- Properties
- SQL
- PowerShell
- 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 330)* are available for all types of files.
- **Shortcut Actions** - Many of the shortcut actions that are available in **Text** mode 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 455)* 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 444)* 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 494)*.
- **Encoding/Decoding Actions** - Contextual menu actions are available to encode or decode Base 64, Base 32, and Hex schemes *(on page 495)*.
- **Code Templates** - You can define your own code templates *(on page 462)* for any type of file and use the *Content Completion Assistant (on page 2303)* 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 79)* and go to

Editor > Syntax Highlight *(on page 158)*. 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 364)* 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 402)* can also be used to compare non-XML files.

Using Projects to Group Documents

Oxygen XML Developer includes a **Project** view *(on page 334)* that helps you organize your projects. Oxygen XML Developer 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 331)*

- **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

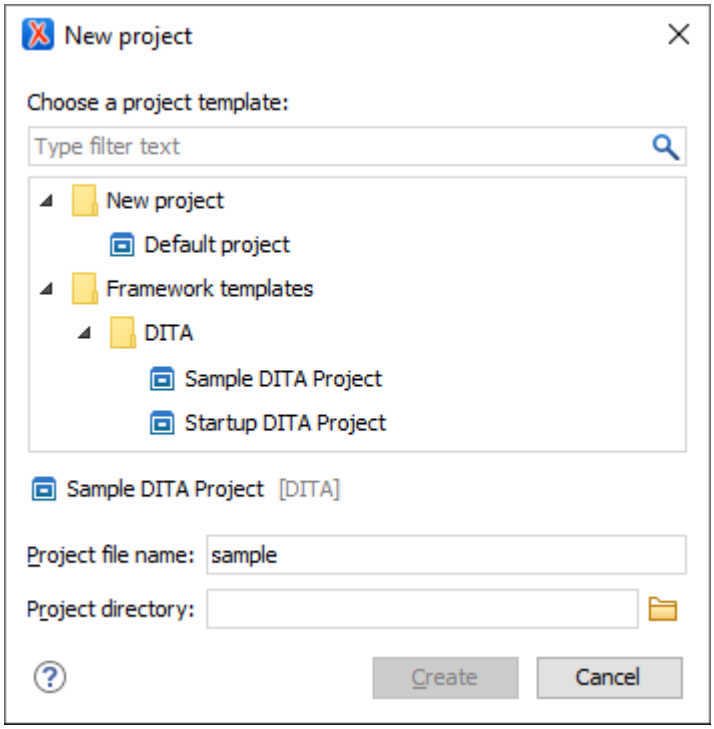
Oxygen XML Developer allows you to organize your files into projects. Projects help you manage and organize your files and allow you to perform batch operations (such as validations and transformations) over multiple files. You can also share your project settings and transformation/validation scenarios *(on page 347)* with other users. Use the **Project** view *(on page 334)* 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 65. 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 331)*

- **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 1735)*.

After selecting a project template, you can specify the following:

Project file name

Specifies the name of the new project file. Oxygen XML Developer 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 Developer 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 309)* 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 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 334)* 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 334)* 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 334)* 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 334)*.

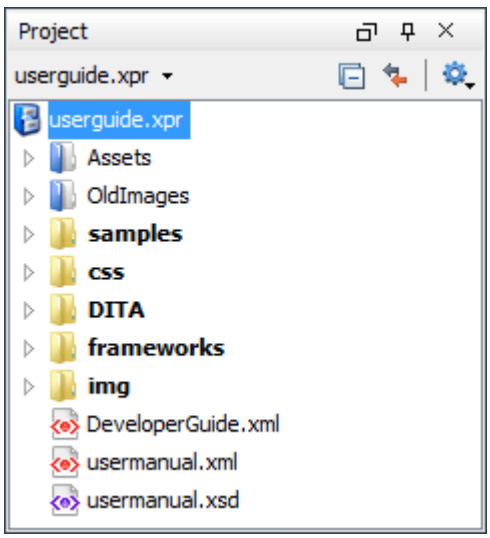
For more details about how you can share projects with other users, see Sharing a Project - Team Collaboration *(on page 347)*.

Related information
Using Projects to Group Documents *(on page 330)*

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.

Figure 66. Project View



By default, the view is positioned on the left side of Oxygen XML Developer, above the **Outline** view. 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 91)*, 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 currently edited file (from the main editor or from the DITA Maps Manager view) is highlighted in the project tree, if the file is found in the project.

> **Note:**
> This button is disabled automatically when you move to the **Debugger** *perspective (on page 2307)*.

Settings

A submenu that contains the following actions:

Filters

Opens a **Project filters** dialog box where you can choose to show hidden files/ folders in the **Project** view and specify filter patterns. It contains the following options:

Show hidden files and directories

If selected, the hidden files and directories are shown in the **Project** view.

File filter patterns

- **Matching files will be shown** - You can specify a comma-separated list of file patterns to be shown in the **Project** view.
- **Matching files will be hidden** - You can specify a comma-separated list of file patterns to be hidden in the **Project** view.

Directory filter patterns

You can specify a comma-separated list of specific directories to be hidden in the **Project** view.

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 350)*.

Change Search and Refactor operations scope

Allows you to change the collection of documents that define the context of the search and refactor operations.

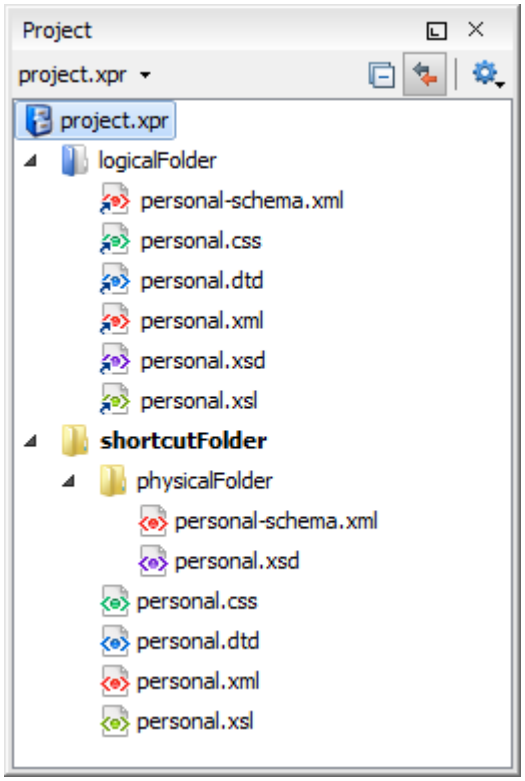
- **Use only Main Files, if enabled** - Restricts Oxygen XML Developer to perform the search and refactor operations starting from the *main files (on page 2306)* that are defined for the current resource. This option is available when you select **Project** in the **Select the scope for Search and Refactor 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 (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 67. 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 330)*.

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 wizard *(on page 299)* 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 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). If the **Main Files** support is enabled and the resources about to be deleted are referenced elsewhere, Oxygen XML Developer shows a warning dialog box where you can see all references to the selected resources.

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 235)*).

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) or folders 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 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) or folders 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 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 Developer 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 or folders)

Opens the **Rename resource** dialog box *(on page 344)* 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 or folders)

Opens the **Move resource** dialog box *(on page 344)* 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 578)* that presents refactoring operations to assist you with managing the structure of your XML documents.

Apply all default quick fix proposals

Opens the **Apply all default quick fix proposals** tool *(on page 615)* that can be used to apply quick fix proposals for all reported validation errors in the selected documents when one or more quick fix proposals have been detected for reported validation errors.

Other XML Refactoring Actions

For your convenience, the last 5 XML Refactoring tool operations *(on page 578)* that were finished or previewed will also appear in this submenu.

Show referenced resources

Opens the **Referenced/Dependent Resources** view *(on page 570)* that allows you to see the referenced resource hierarchy for an XML document.

Show dependent resources

Opens the **Referenced/Dependent Resources** view *(on page 570)* that allows you to see the resource dependencies for an XML document. When working with DITA XML projects, this action is available when right-clicking images to find direct references to them.

 **Refresh**

Refreshes the content and the dependencies between the resources in the **Main Files** directory *(on page 350)*.

 **Find/Replace in Files**

Opens the **Find/Replace in Files** dialog box *(on page 368)* that allows you to find and replace text in multiple files.

 **XPath in Files**

Opens the **XPath/XQuery Builder** view *(on page 1729)* 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 358)*.

 **Check Spelling in Files**

Allows you to check the spelling of multiple files *(on page 389)*.

 **Format and Indent Files**

Opens the **Format and Indent Files** dialog box *(on page 486)* that allows you to configure the format and indent (*pretty-print (on page 2307)*) action that will be applied on the selected documents.

 **Open in SVN Client**

Syncro SVN Client *(on page 2060)* 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 402)* or **Compare Directories** *(on page 421)* 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 405)*. 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.

Configure Transformation Scenario(s)

Opens a dialog box 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

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 1738)* 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.

Generate Documentation submenu

Generate Documentation > XML Schema Documentation

Opens the XML Schema Documentation dialog box *(on page 753)*.

Generate Documentation > XSLT Stylesheet Documentation

Opens the XSLT Stylesheet Documentation dialog box *(on page 665)*.

Generate Documentation > XQuery Documentation

Opens the XQuery Documentation dialog box *(on page 785)*.

Generate Documentation > WSDL Documentation

Opens the WSDL Documentation dialog box *(on page 808)*.

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 351)* for each project.

Detect Main Files (Available from the project container when Main Files Support is enabled)

Opens the **Detect Main Files** wizard *(on page 352)* 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 353)*.

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 330)*.

Open Project (Ctrl + F2 (Command + F2 on macOS))

Opens an existing project. Alternatively, you can open a project by dropping an Oxygen XML Developer 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 351)* 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 334)* 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 by using the  button.

Renaming Resources

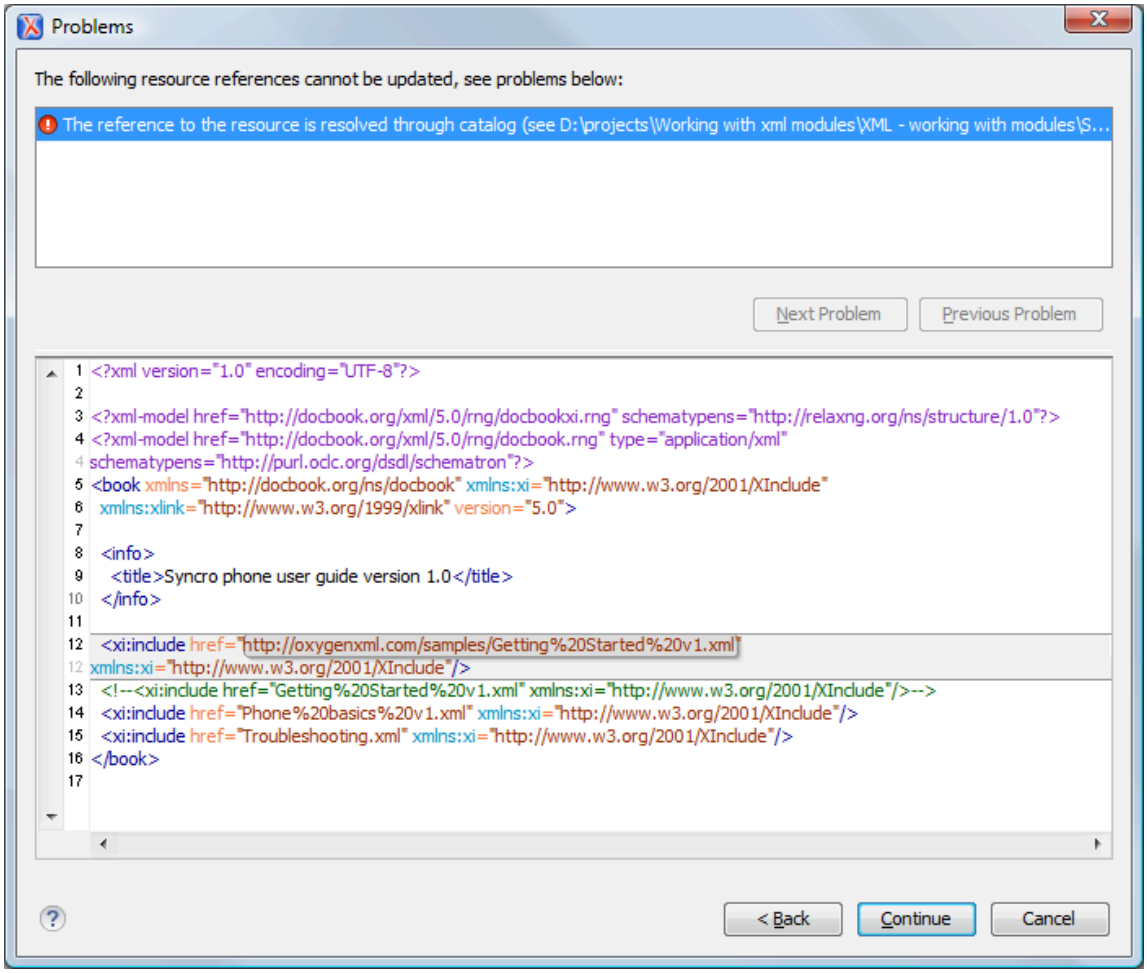
You can rename certain types of files (such as XML, XML Schema, Relax NG, WSDL, and XSLT) by using the **Refactoring > Rename resource** action from the contextual menu. This action opens the **Rename resource** dialog box that includes the following options:

- **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 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 2309)* or when the path to the moved or renamed resource contains entities. For these cases, Oxygen XML Developer displays a warning dialog box.

Figure 68. Problems Dialog Box



Batch Validation and Transformation

Oxygen XML Developer 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 1738)* 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.

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.



Configure Transformation Scenario(s)

Opens a dialog box 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 350)*
Quick Validation and Transformation for Main Files *(on page 354)*

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 Developer also allows you to store preferences, transformation scenarios, and validation scenarios at project level *(on page 2308)* 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 334)*, create a project or open an existing one.
3. Open the **Preferences** dialog box **(Options > Preferences)** *(on page 79)*.
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 2308)* 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 334)*, create a project or open an existing one.
2. When you create a new transformation scenario *(on page 1101)* or edit an existing one *(on page 1207)*, there is a **Storage** option. Switch the storage preference to **Project Options** *(on page 2308)* 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 334)*, create a project or open an existing one.
2. When you create a new validation scenario or edit an existing one, there is a **Storage** option. Switch the storage preference to **Project Options** *(on page 2308)* 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 Developer. 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 add-on.

Using Subversion (SVN) for Collaboration

Oxygen XML Developer also includes an embedded SVN (Subversion) Client. 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 334)*. 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 241)*
Sharing Transformation Scenarios *(on page 1216)*
Sharing Validation Scenarios *(on page 546)*

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 Developer 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 79)*, 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 2308)* 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 334)*.
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 Developer allows you to define *Main Files (on page 2306)* at project level. These *main files* are automatically used by Oxygen XML Developer 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 Developer maintains the hierarchy of the *main files*, helping you to determine the editing context.

Resources

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

Related information

Modular Contextual XML Editing Using 'Main Files' Support *(on page 567)*
Modular Contextual Schematron Editing Using 'Main Files' Support *(on page 962)*
Modular Contextual XSLT Editing Using 'Main Files' Support *(on page 627)*
Modular Contextual XML Schema Editing Using 'Main Files' Support *(on page 731)*

Modular Contextual Relax NG Schema Editing Using 'Main Files' Support *(on page 824)*
Modular Contextual Ant Build File Editing Using 'Main Files' Support *(on page 676)*

Main Files Benefits

Using the *Main Files* support in Oxygen XML Developer includes the following benefits:

- When the *main file* is validated, Oxygen XML Developer automatically identifies the modules included in the *main file* and validates all of them.
- When the *main file* is transformed, Oxygen XML Developer automatically identifies the modules included in the *main file* and transforms them accordingly.
- The *Content Completion Assistant (on page 2303)* presents all the components that are collected from the *main files* for the modules they include.
- The **Outline** view *(on page 465)* displays all the components that are defined in the *main files* hierarchy.
- The *main files* that are defined for the current module determines the scope of the search and refactoring actions *(on page 569)*. Oxygen XML Developer 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 Developer stores the *main files* in a folder located in the **Project** view *(on page 334)*, as the first child of the project root. The *Main Files Support* is disabled by default and Oxygen XML Developer 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 334)*.
- Select **Enable Main Files Support** from the contextual menu of the project root folder in the **Project** view *(on page 334)*. 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 334)*. 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. You 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 Developer hides it for all projects. You can make the window tip reappear by resetting Oxygen XML Developer to its default settings *(on page 242)*. However, doing so will result in you losing your customized options.

Related information

Detecting Main Files *(on page 352)*
Adding or Removing Files/Folders in the Main Files Directory *(on page 353)*

Detecting Main Files

Oxygen XML Developer 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 Developer 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 2304)* 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 Developer 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 351)*
Adding or Removing Files/Folders in the Main Files Directory *(on page 353)*

Adding or Removing Files/Folders in the Main Files Directory

Adding Files/Folders to the Main File Directory

The `Main Files` directory can contain logical folders, linked folders, or 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 **Referenced/Dependent Resources** view *(on page 570)*, use the **Add to Main Files** action.

To add folders in the `Main Files` directory, use one of the following methods:

- Right-click `Main Files` directory and select **Add Folder** from the contextual menu.
- Drag and drop folders into the `Main Files` directory.

You can view the *main files* for the current resource by selecting **Properties** from the contextual menu *(on page 343)* of the **Project** view *(on page 334)* and the *main files* for the current editor in the **Properties** *(on page 328)* and **Information** *(on page 437)* views.

Removing Files/Folders 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 or folder, remove it from the `Main Files` folder (for example, right-click and select **Remove from Main Files**). Removing files or folders from the `Main Files` folder does NOT delete the files from disk. It just removes the logical files from that logical folder in the project.

Related information

Enabling the Main Files Support *(on page 351)*

Detecting Main Files *(on page 352)*

Quick Validation and Transformation for Main Files

If *Main Files Support* is enabled *(on page 351)*, you can hover the cursor over the Main Files directory, or a node within the directory, and Oxygen XML Developer 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

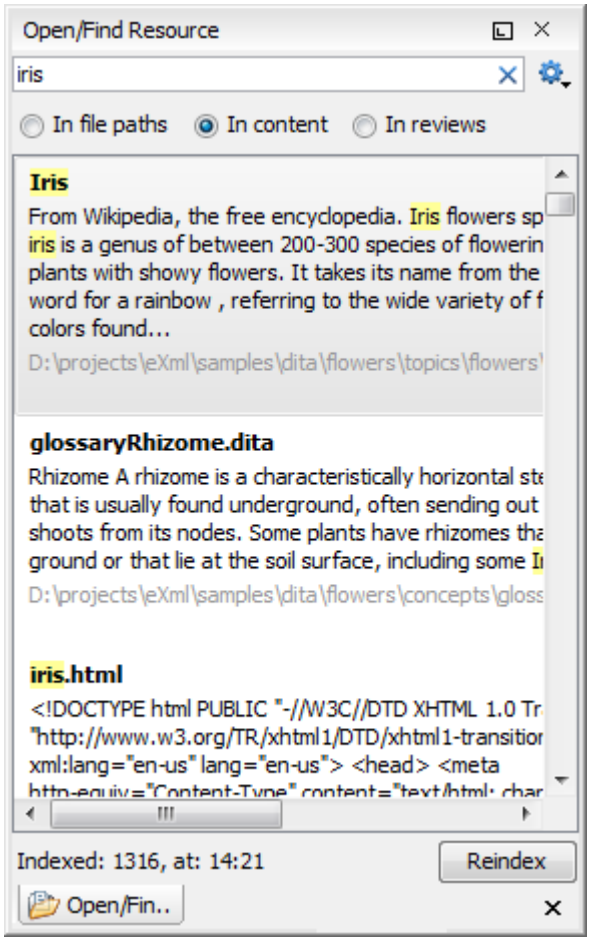
Oxygen XML Developer includes advanced search capabilities to help you locate documents and resources. The search features are powered by Apache Lucene. Apache Lucene is a free open source information retrieval software library. You can perform simple text searches or more complex searches using the Apache Lucene - Query Parser Syntax.

Note:
When Oxygen XML Developer 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.developer\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 Developer layout, next to the **Project** view *(on page 334)* . If the view is not displayed, it can be opened by selecting it from the **Window > Show View** menu.

Figure 69. Open/Find Resource View



You can use this view to find a file in the current Oxygen XML Developer project 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 356)*.

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 474)* view 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 227)* 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 363)* - Select this option to search for resources by their name or by its path (or a fragment of its path).
- **In content** *(on page 360)* - Select this option to search through the content of your resources.
- **In reviews** *(on page 363)* - 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 Developer 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.
- **Insert as Link** - Inserts a relative link to the selected resource at the cursor location within the current document.

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 227)*.

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 Developer 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 226)*.

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.

Resources

For more information about the **Open/Find Resource** feature and its search capabilities, watch our video demonstration:

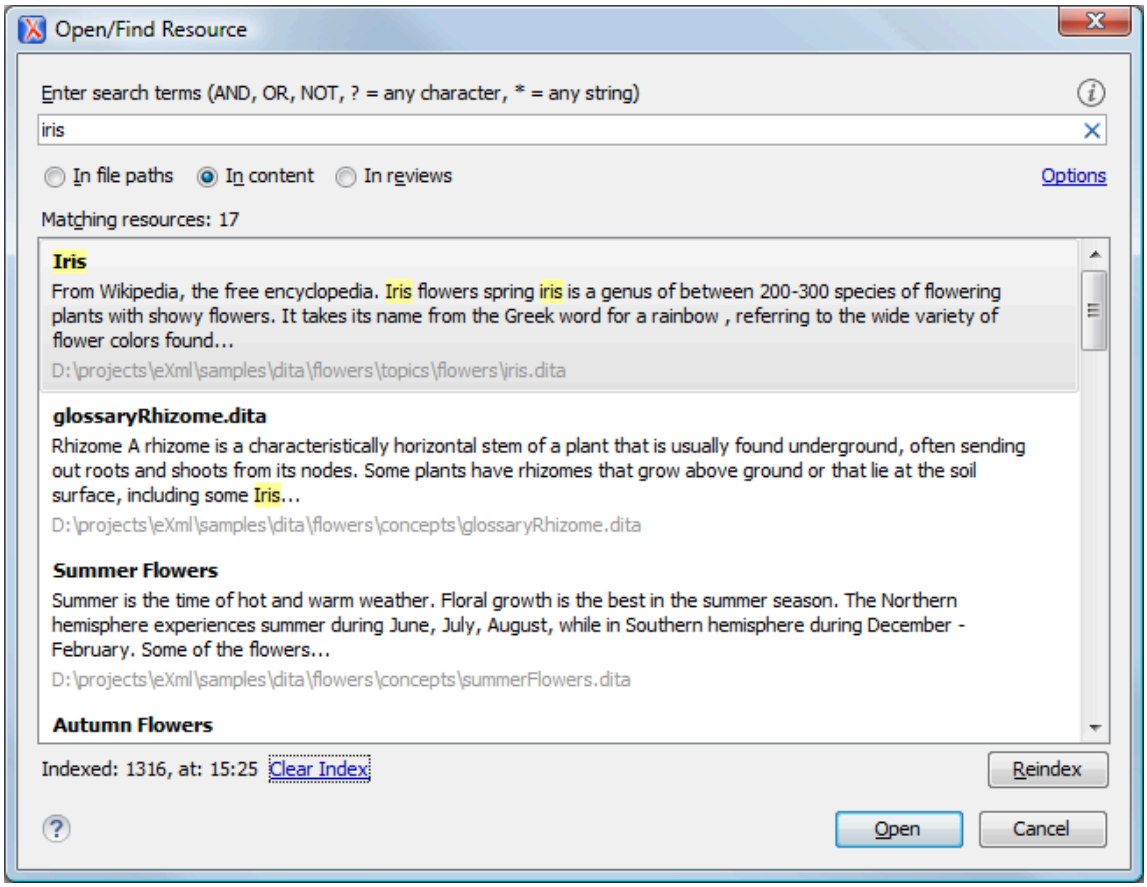
https://www.youtube.com/embed/PENoDNdaGao

Related information
Open/Find Resource Dialog Box *(on page 358)*

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.

Figure 70. Open/Find Resource Dialog Box



You can use this dialog box to find a file in the current Oxygen XML Developer project 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).

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 474)* view 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 363)* - Select this option to search for resources by their name or by its path (or a fragment of its path).
- **In content** *(on page 360)* - Select this option to search through the content of your resources.
- **In reviews** *(on page 363)* - 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 227)* 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 Developer 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.
- **Insert as Link** - Inserts a relative link to the selected resource at the cursor location within the current document.

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 227)*.

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 Developer 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 226)*.

Resources

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 354)*
Open/Find Resource Preferences Page *(on page 227)*

Searching in Content

To perform a search through the content of your resources, open the **Open/Find Resource** dialog box *(on page 358)* (from the **Find** menu or with **Ctrl + Shift + R (Command + Shift + R on macOS)**) or the **Open/Find Resource** view *(on page 354)* (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:

  ```
  title:(+Flowers +"Garden Preparation")
  ```

- **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 358)* (from the **Find** menu or with **Ctrl + Shift + R (Command + Shift + R on macOS)**) or the **Open/Find Resource** view *(on page 354)* (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 358)* (from the **Find** menu or with **Ctrl + Shift + R (Command + Shift + R on macOS)**) or the **Open/Find Resource** view *(on page 354)* (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 474)* view 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.

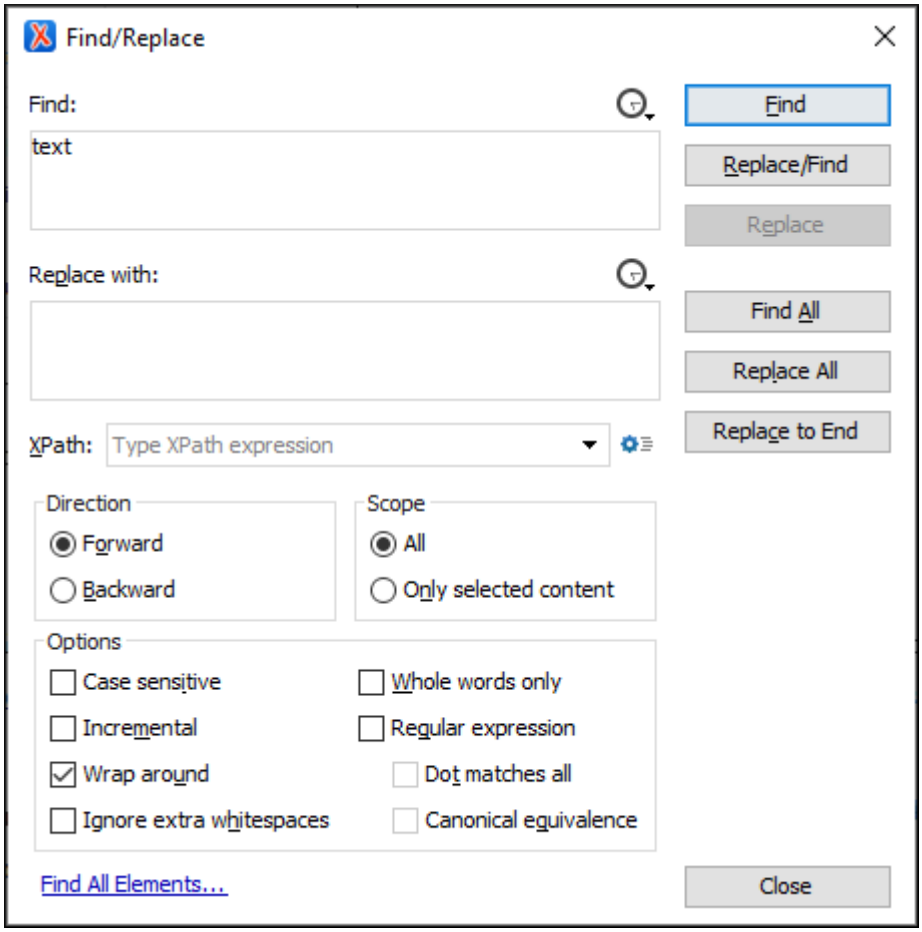
You can use Perl-like regular expressions syntax *(on page 379)* to define patterns. A content completion assistance window is available in the **Find** and **Replace with** areas to help you edit regular expressions. It is activated every time you type ****(backslash key) or on-demand if you press **Ctrl + Space** 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

Figure 71. The 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 367)*. 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 367)* 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

> **Note:**
> This option is disabled if the XML document is not well-formed or if the XML document is large and opened using the large or huge file loading optimization feature *(on page 397)*.

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 2303)* to help you input XPath expressions that are valid in the current context. See Working with XPath Expressions *(on page 1726)* 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 379)* 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 2303)* decompositions match. For example, the **ã** 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 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 474)* view.

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

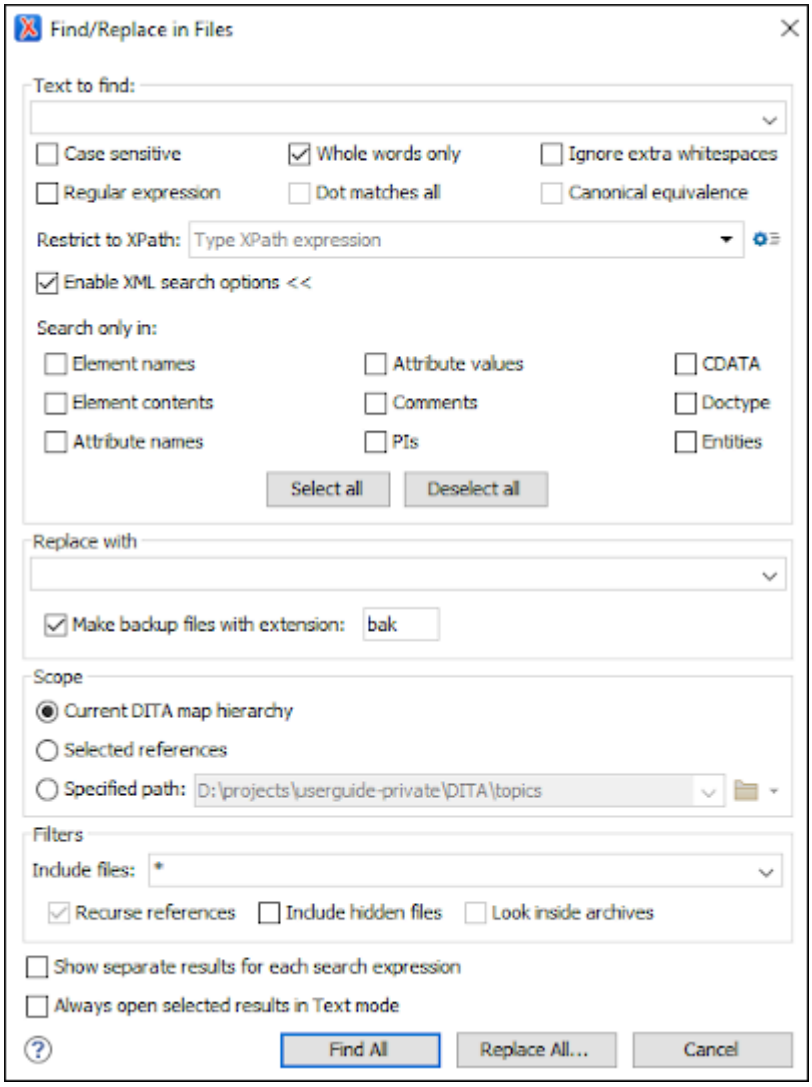
The **Find/Replace in Files** feature enables you to define *Search for* or *Search for and Replace* operations across multiple files. To open the **Find/Replace in Files** dialog box, use the  **Find/Replace in Files** action that is available in the following locations:

- The **Find** menu.
- The  **Find/Replace in Files** button on the main toolbar.
- The contextual menu of the **Project** view *(on page 334)*.
- The contextual menu of the **Data Source Explorer** view *(on page 1742)* for most types of database connections.

The operation works on both local and remote files from an SFTP, WebDAV, or CMS server.

Find/Replace in Files Dialog Box

Figure 72. 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 379)* 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 2303)* decompositions match. For example, the **ã** 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

> **Note:**
> This option only works on files that have an association to an XML-based editor (XML, XSL, XSD) in the Document Type Association Preferences *(on page 93)*.

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.

Example: Use the XPath filter expression `//*[not(local-name() = 'uicontrol')]` to skip over the contents of any `<uicontrol>` element.

> **Tip:**
> You can use the *Content Completion Assistant (on page 2303)* to help you input XPath expressions that are valid in the current context. See Working with XPath Expressions *(on page 1726)* 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

> **Note:**
> This option is only available when editing in **Text** mode and it only works on files that have an association to an XML-based editor (XML, XSL, XSD) in the Document Type Association Preferences *(on page 93)*.

This option 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 582)*. You can even customize your own refactoring operations *(on page 594)*.

Make backup files with extension

In the replace process Oxygen XML Developer 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 Developer. 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.

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 1735)* 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**.

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 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 Developer 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 379)* 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 Developer 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 226)*. For the other files it uses the encoding configured for non-XML files *(on page 123)*.
- You can cancel a long operation at any time by pressing the **Cancel** button of the progress dialog box or in the status bar, 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.

Related information

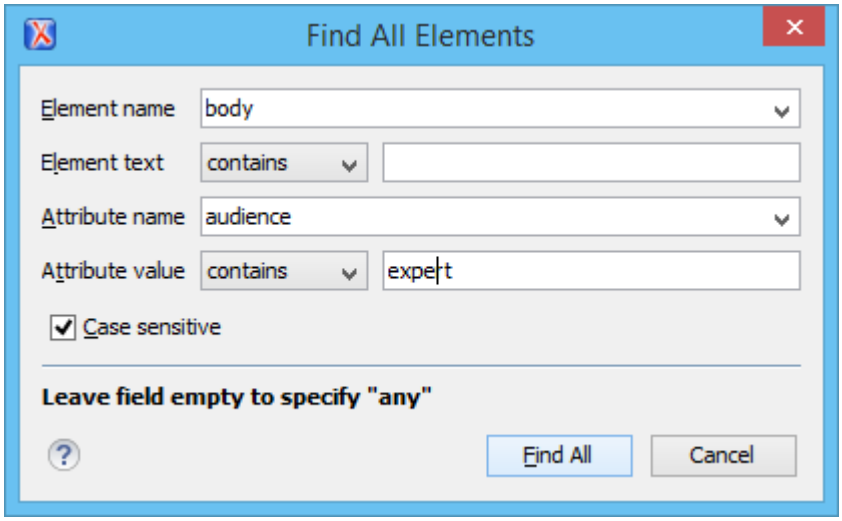
Built-in Refactoring Operations *(on page 582)*

Custom Refactoring Operations *(on page 594)*

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 364)*. It assists you in defining XML element / attribute search operations in the current document.

Figure 73. Find All Elements Dialog Box



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 (`<namespace prefix>:<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 (`<namespace prefix>:<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.

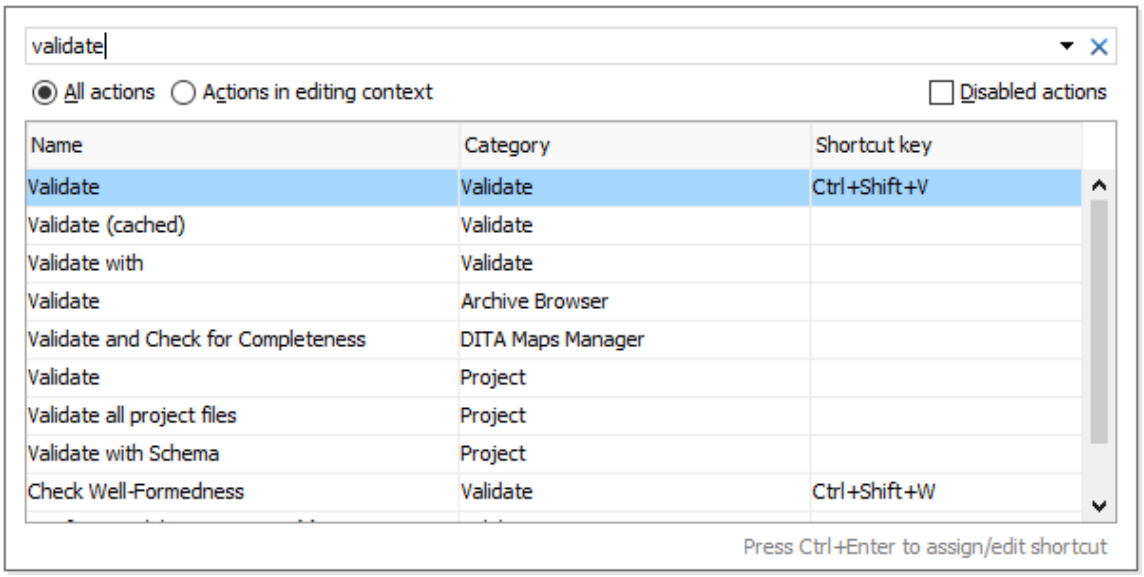
When you select **Find All**, Oxygen XML Developer tries to find the items that match all the search parameters. The results of the operation are presented as a list in the message panel.

Find and Invoke Actions

Oxygen XML Developer 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.

Figure 74. Find Action Pop-Up Window



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 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 364)* is available as a dockable toolbar *(on page 285)*. To display it, press the **Alt + Shift + F (Command + Option + 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 Developer 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 75. 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 364)*.
- **Find/Replace in Files** - Opens the **Find/Replace in Files** dialog box *(on page 368)*.
- **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 364)*, 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 Developer uses the Java regular expression syntax. It is **similar** to that used in Perl 5, with several exceptions. Thus, Oxygen XML Developer 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 Developer/Java in the text searches from the **Find/Replace** dialog box *(on page 364)* and **Find/Replace in Files** dialog box *(on page 368)*. 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 364)* and **Find/Replace in Files** dialog box *(on page 368)*:

- **/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, the `^` character matches at the beginning of the string, while in Java, it matches the beginning of a line.
- In Perl, the `$` character matches at the end of the string, while in Java, it matches the end of a line.
- 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 `*`.

Related information

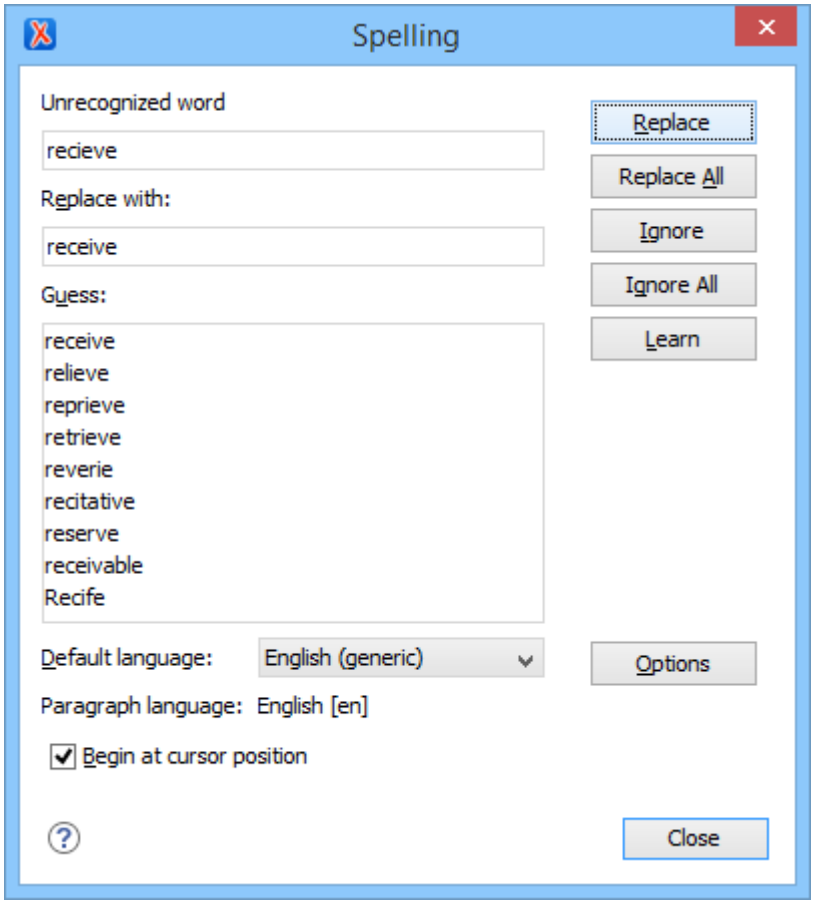
Comparison between the Java and Perl 5 regular expression syntax

Spell Checking

Oxygen XML Developer includes an automatic (as-you-type) spell checking feature *(on page 388)*, 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 **Check Spelling** action on the toolbar or from the **Edit** menu.

Figure 76. Check Spelling Dialog Box



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 Developer 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 Developer skips the content of the XML elements marked to be ignored *(on page 388)*.

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 163)* where you can configure various options regarding the feature.

Spell Check Dictionaries and Term Lists

Oxygen XML Developer 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 Developer includes the following built-in dictionaries for the spell checker:

- English (US) [`en_US`]
- English (UK) [`en_GB`]
- French [`fr`]
- German [`de_DE`]
- Spanish [`es_ES`]

Other Hunspell Dictionaries

You can also download Hunspell dictionaries for other languages and add them to the Oxygen XML Developer spell checker. For a website that includes numerous dictionary files, see: https://extensions.openoffice.org/.

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 Developer spell checker by following this procedure *(on page 383)*.

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 Developer provides a way of adding personalized *term lists (on page 385)* 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 Developer uses for storing *learned words (on page 387)*.

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 383)*
Adding Custom Spell Check Term Lists *(on page 385)*
Building and Testing Hunspell Dictionaries

Adding Custom Dictionaries and Term Lists

The Oxygen XML Developer 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 Developer spell checker.

You can add dictionaries *(on page 383)* and personalized term lists *(on page 385)* 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 166)* and the **Include dictionaries and term list from** option *(on page 166)* 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 Developer.

Related information

Replacing a Spell Check Dictionary *(on page 386)*

Editing the Spell Checking Dictionaries

Adding Custom Spell Check Dictionaries

There are three possible scenarios for adding Hunspell dictionaries to the Oxygen XML Developer 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. For a website that includes numerous dictionary files, see: https://extensions.openoffice.org/.

   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 79)* and go to **Editor > Spell Check > Dictionaries** *(on page 166)*.
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 166)*.
   b. Copy both files (`.dic` and `.aff`) to any other directory, select the **Include dictionaries and term list from** option *(on page 166)*, and select that directory. If you choose this option, make sure you read this important note *(on page 166)*.
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.

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
   asimptotic
   ```

   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 79)* and go to **Editor > Spell Check > Dictionaries** *(on page 166)*.
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 166)*.
   b. Save both files (`.dic` and `.aff`) to any other directory, select the **Include dictionaries and term list from** option *(on page 166)*, and select that directory. If you choose this option, make sure you read this important note *(on page 166)*.
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 79)* and go to **Editor > Spell Check > Dictionaries** *(on page 166)*.
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 166)*.
   b. Save both files (`.dic` and `.aff`) to any other directory, select the **Include dictionaries and term list from** option *(on page 166)*, and select that directory. If you choose this option, make sure you read this important note *(on page 166)*.
4. Restart the application for the spell checker to start using the new dictionary.

Related information
Adding Custom Spell Check Term Lists *(on page 385)*
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:

```
parabola

asimptotic

*hyperbola
```

> **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 79)* and go to **Editor > Spell Check > Dictionaries** *(on page 166)*.
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 166)*.
   b. Save the file (`.tdi`) to any other directory, select the **Include dictionaries and term list from** option *(on page 166)*, and select that directory. If you choose this option, make sure you read this important note *(on page 166)*.
5. Restart the application for the spell checker to start using the new term list.

Related information
Adding Custom Spell Check Dictionaries *(on page 383)*

Replacing a Spell Check Dictionary

There are several possible scenarios for replacing an existing Hunspell dictionary for the Oxygen XML Developer 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. For a website that includes numerous dictionary files, see: https://extensions.openoffice.org/.
2. Open the **Preferences** dialog box **(Options > Preferences)** *(on page 79)* and go to **Editor > Spell Check > Dictionaries** *(on page 166)*.
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 166)*. 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 166)*. If you choose this option, make sure you read this important note *(on page 166)*.

> **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 79)* and go to **Editor > Spell Check > Dictionaries** *(on page 166)*.
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 166)*. 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 166)*. If you choose this option, make sure you read this important note *(on page 166)*.
4. Restart the application for the spell checker to start using the new dictionary.

Related information
Adding Custom Dictionaries and Term Lists *(on page 382)*

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 380)* 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 166)*.

Related information
Adding Custom Spell Check Term Lists *(on page 385)*

Ignored Words (Elements)

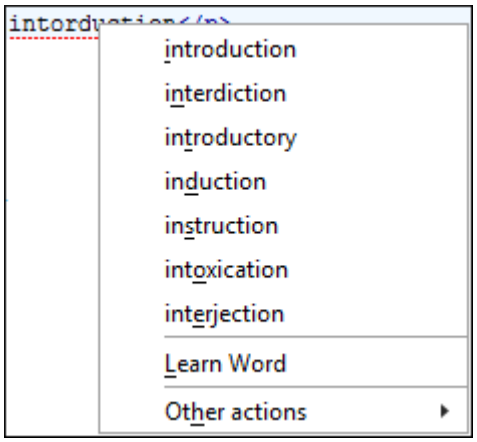
You may want the content of certain XML elements to always be skipped during the spell check process (for example, `<programlisting>`, `<codeblock>`, `<codeph>`, `<filepath>`, or `<screen>`). This can be done in one of several ways:

- You can skip through them manually, word by word, using the **Ignore** button in the **Spelling** dialog box *(on page 380)* 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 165)* in the **Spell Check** preferences page.

Automatic Spell Check

Oxygen XML Developer includes an option to automatically check the spelling as you type. This feature is disabled by default, but it can be enabled and configured in the **Spell Check** preferences page *(on page 163)*. When the **Automatic Spell Check** option *(on page 163)* 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.

Figure 77. Automatic Spell Checking in Text Mode



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 387)*.

Other actions

This submenu give you access to all the usual contextual menu actions.

Related information

Learned Words *(on page 387)*

Spell Check Multiple Files

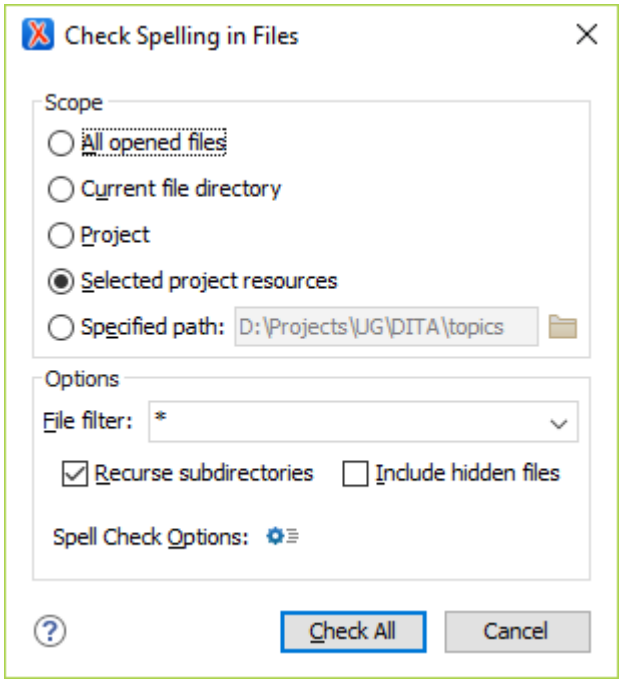
The **Check Spelling in Files** action allows you to check the spelling on multiple local or remote documents. This action is available in the following locations:

- The **Edit** menu.
- The contextual menu of the **Project** view *(on page 334)*.

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 474)* view 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 78. 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.
- **Entire 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 163)*.

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 Developer 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 andOxygen XML Developer provides support for all Unicode characters *(on page 391)*. There are various encoding options and features to help determine how to handle documents with unsupported characters *(on page 392)*.

Fonts

Oxygen XML Developer provides the ability to choose the fonts to be used in the various editing modes *(on page 88)*. 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 Developer tries to find that symbol in a *fallback* font *(on page 393)*.



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 Developer, open the **Preferences** dialog box **(Options > Preferences)** *(on page 79)*, go to **Appearance > Fonts**. You can select a font for each editing mode in this preferences page.

Navigation and Layout

Oxygen XML Developer 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 489)* for details about which option to choose.

Editing

Oxygen XML Developer includes a contextual menu action that converts a sequence of hexadecimal characters to the corresponding Unicode character *(on page 494)*.

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 394)*.

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 Developer 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 Developer uses 16-bit characters covering the Unicode Character set.

Note:
Oxygen XML Developer 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**.

Related information
Unicode Fallback Font Support *(on page 393)*
Inserting Special Characters with the Character Map *(on page 394)*

Opening and Saving Documents with Unsupported Characters

When loading documents, Oxygen XML Developer 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 Developer 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 Developer, 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 124)*. 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 Developer displays a notification that you need to resolve the conflict before saving the document.

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 Developer from the **Encoding** preferences page *(on page 123)*.

Unicode Fallback Font Support

Oxygen XML Developer provides fonts for most common Unicode ranges. However, if you use special symbols or characters *(on page 394)* 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 391)* as possible. When a display system encounters a character that is not part of the range of any of the available fonts, Oxygen XML Developer 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 Developer 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 Developer. To allow Oxygen XML Developer 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 Developer 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 Developer for the changes to take full effect.

Result: Whenever Oxygen XML Developer 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 Developer, open the **Preferences** dialog box **(Options > Preferences)** *(on page 79)*, 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 Developer for the font changes to take full effect.

Related information

Unicode Support *(on page 391)*
Inserting Special Characters with the Character Map *(on page 394)*

Inserting Special Characters with the Character Map

Oxygen XML Developer 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 395)* by selecting **More symbols** from the Ω **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 291)*).
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 Ω **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

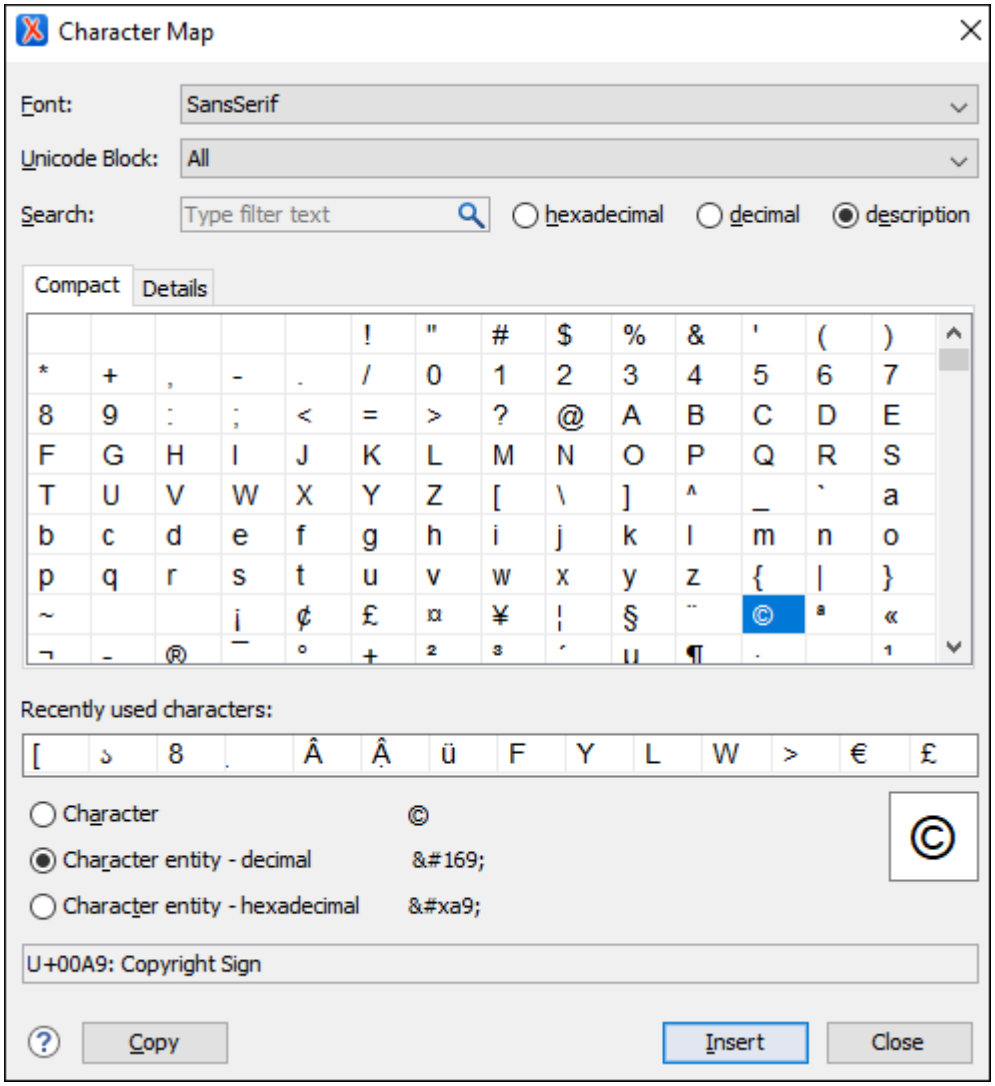
You can see the hexadecimal value for any character that is already inserted in your document by placing the cursor right after the character and you can see its value in the status bar at the bottom of the application.

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 395)* by selecting **More symbols** from the Ω **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 291)*).
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

Figure 79. 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 396)*. 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 your document. 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 your document next to the selections in this section. 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 88)* 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 279)*. 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 390)*

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 334)*, double-click the image name.
- In the **Project** view *(on page 334)*, right-click an image and select **Preview**.
- 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 1006)*, the **Image Preview** pane also includes the following other contextual menu actions: **Zoom in**, **Zoom out**, **Rotate**, and **Refresh**.

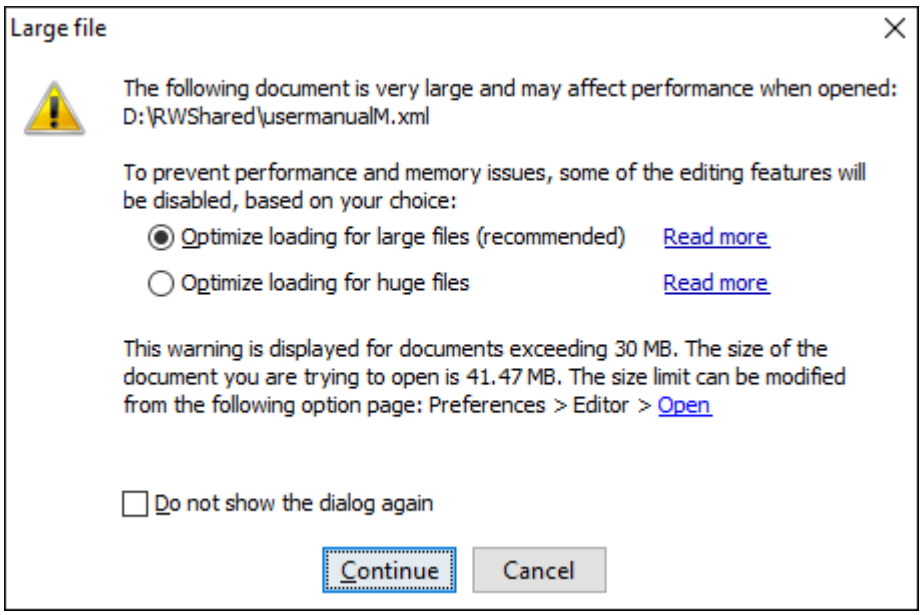
While the **Image Preview** view is visible, selecting an image in the **Project** view *(on page 334)* will automatically display the resource in the view. Also, as long as the **Link with Editor** toggle action (located on the **Project** view's toolbar) is enabled, focusing on the image in the **Image Preview** view will select the image file in the **Project** 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 134)*, Oxygen XML Developer prompts you to choose whether you want to optimize the loading of the document for large files or for huge files.

Figure 80. Large File Prompt Dialog Box



If your file has a size smaller than 300 MB, the recommended approach is **Optimize loading for large files** *(on page 398)*. For documents that exceed 300 MB, the recommended approach is **Optimize loading for huge files** *(on page 399)*.

Optimize Loading for Large Files

If you open a document that exceeds the limit configured in **Open** preferences *(on page 134)* (the default limit is 30 MB), a dialog box will be displayed *(on page 397)* 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 is not available.
- The XPath filter is disabled in the Find/Replace dialog box *(on page 364)*.
- The bidirectional Unicode support (right-to-left writing) is disabled.
- The **Format and indent the document on open** option *(on page 136)* 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 136)*.
- Localizations for the results of an XPath expression will be less precise.

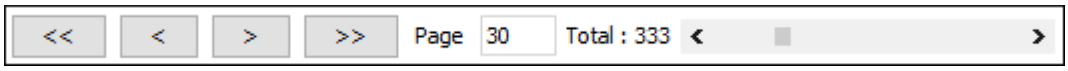
Related information

Optimize Loading for Huge Files *(on page 399)*

Optimize Loading for Huge Files

If you open a document that exceeds the limit configured in **Open** preferences *(on page 134)* (the default limit is 30 MB), a dialog box will be displayed *(on page 397)* 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 81. Huge File Editor Horizontal Slider



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 is disabled.
- The XPath filter is disabled in the **Find/Replace** dialog box *(on page 364)*.
- The bidirectional Unicode support (right-to-left writing) is disabled.
- The **Format and indent the document on open** option *(on page 136)* is automatically deselected for non-XML documents. For XML documents, the formatting uses less memory by ignoring the options set in the **Format** preferences page *(on page 136)*.
- The **Outline** view is not supported.
- The file content is soft wrapped by default.
- The **Find/Replace** dialog box *(on page 364)* only supports the **Find** action.
- Saving changes is only possible if the **Safe save** option *(on page 135)* (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 398)*

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 128)* 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 Developer **Text** mode. When a certain line length limit is reached (controlled from the Optimize loading for documents with lines longer than (Characters) *(on page 134)* option), Oxygen XML Developer 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 is disabled.
- Automatic spell checking *(on page 388)* is disabled.
- When editing XML documents, the **XPath** field is disabled in the **Find/Replace** dialog box *(on page 364)*.
- 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 Developer you can by default perform modifications to it. This behavior is controlled by the **Can edit read only files** option *(on page 125)*. 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 328)*. 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 Developer 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 Developer 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 421)*, 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 402)*, 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 405)*. If only two files are specified, the tool will start in the 2-way comparison mode *(on page 402)*. 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:

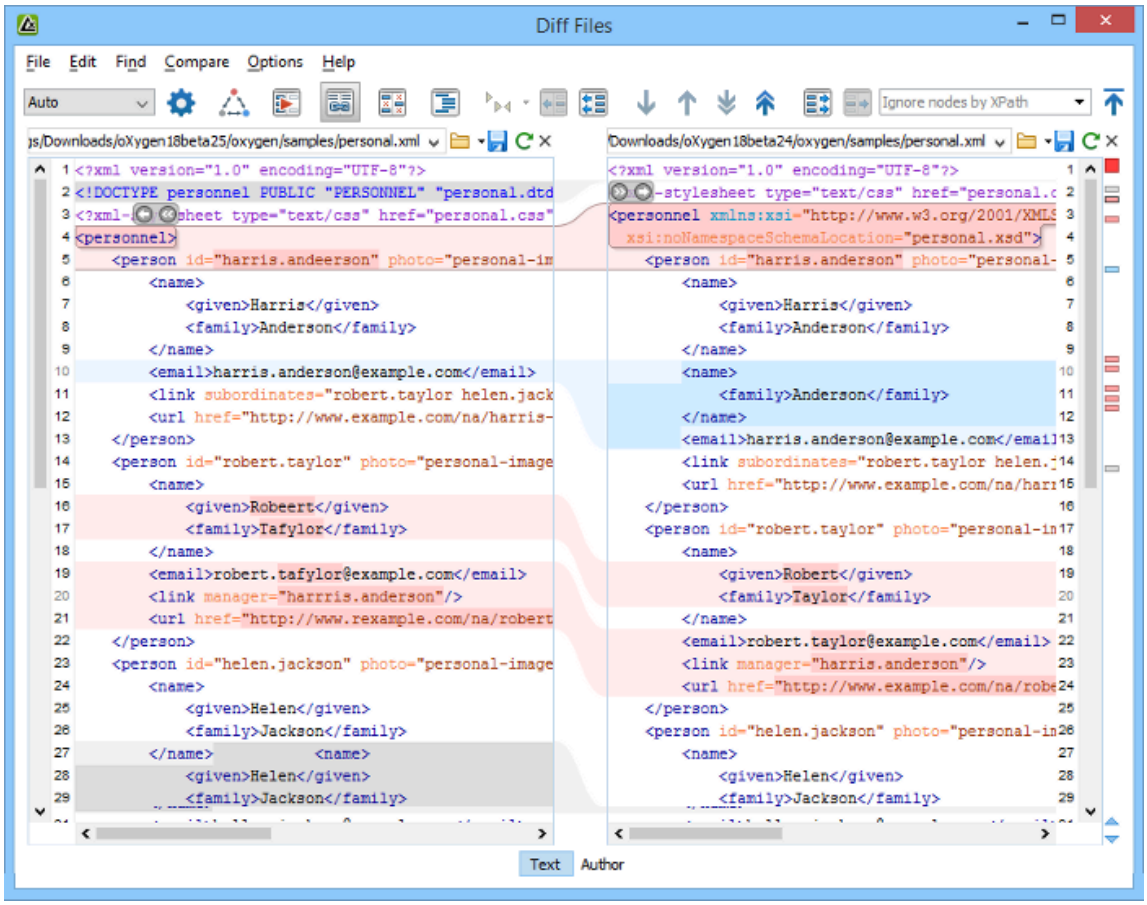
```
diffFiles.bat "c:\docs\file 1" "c:\docs\file 2" c:\docs\basefile
```

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 > Comparison Tools** menu or can be opened as a stand-alone application from the Oxygen XML Developer installation folder (`diffFiles.exe`).

Figure 82. 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:

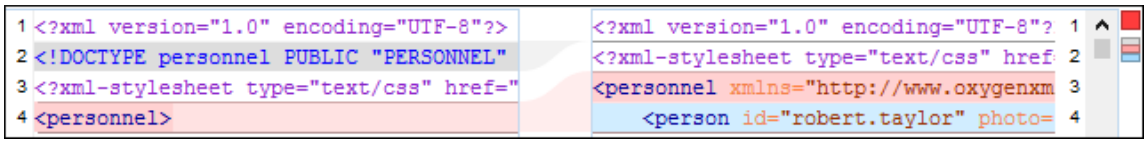
1. Open a file in the left panel and the file you want to compare it to in the right panel. You can specify the path by using the text field, the history drop-down, or the browsing actions in the **Browse** drop-down menu.

   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 404)* 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 83. 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 217)*, 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 413)* and in the various menus *(on page 416)* (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 Developer 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.

- **Words** - Computes the differences at word level, meaning that it compares two files or fragments looking for identical words.
- **Lines** - Computes the differences at line level, meaning that it compares two files or fragments looking for identical lines of text.
- **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).

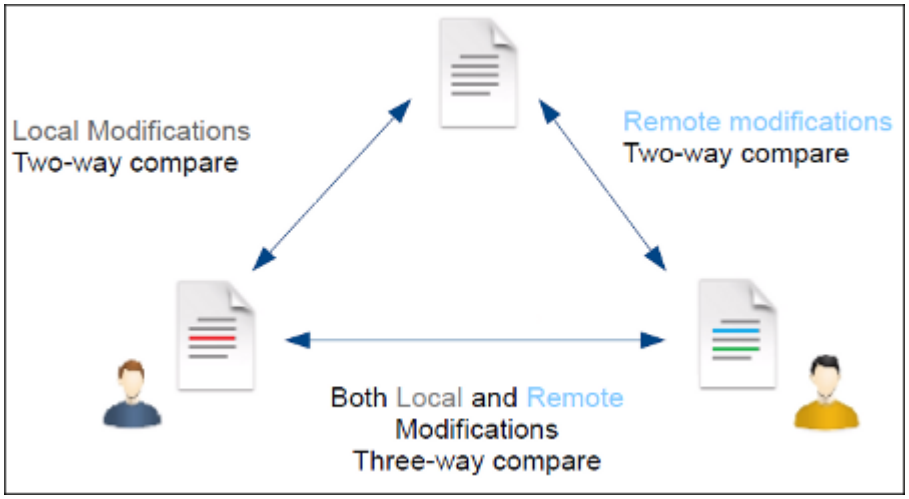
  For example:
  - When comparing XML files or fragments, a token can be one of the following:
    - The name of an XML tag
    - The **<** character
    - The **/>** sequence of characters
    - The name of an attribute inside an XML tag
    - The **=** sign
    - The **"** character
    - An attribute 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 Developer 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 84. Three-Way Comparison



Compare Files

To perform a three-way comparison, follow these steps:

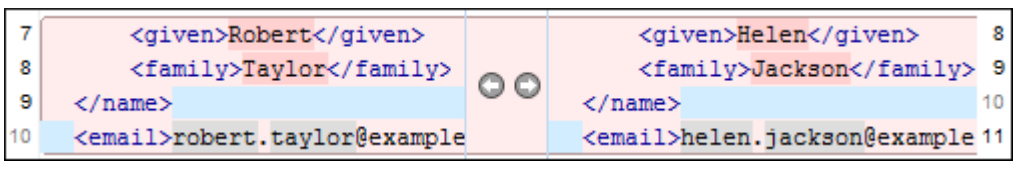
1. Open a file in the left panel and the file you want to compare it to in the right panel. You can specify the path by using the text field, the history drop-down, or the browsing actions in the **Browse** drop-down menu.

   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 404)* 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 85. 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 217)*, 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 413)* and in the various menus *(on page 416)* (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

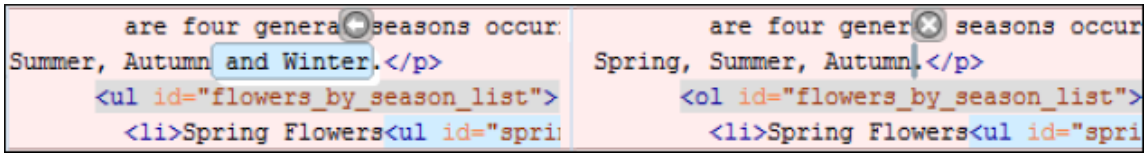
Oxygen XML Developer offers the following three-way diff algorithms to compare files:

- **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.
- **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 Developer 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 405)*, 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.

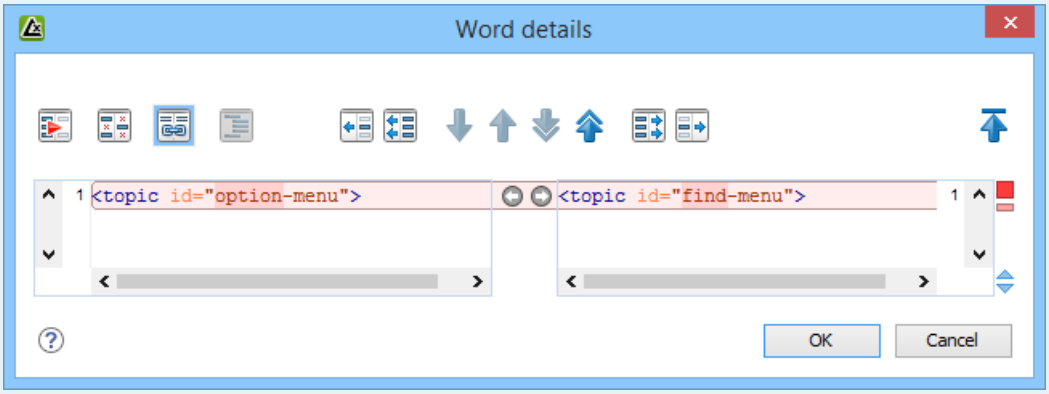
Figure 86. Second-Level Diff Comparison



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.

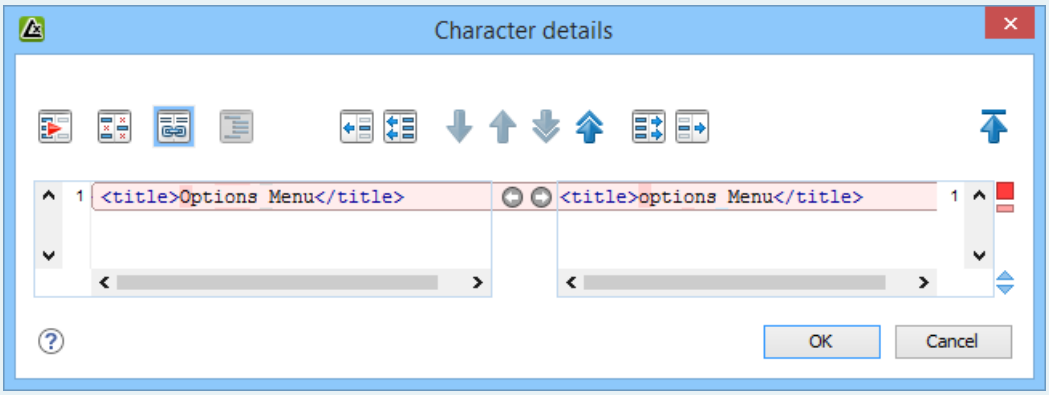
To do a word level comparison, select **Show word level details** from the contextual menu or **Compare** menu.

Figure 87. Word Level Comparison



To do a character level comparison, select **Show Character Level details** from the contextual menu or **Compare** menu.

Figure 88. Character Level Comparison



Related information

Files Comparison Preferences Page *(on page 215)*

Compare Directories Tool *(on page 421)*

Toolbar and Contextual Menu Actions of the Compare Files Tool *(on page 413)*

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 405)*. If only two files are specified, the tool will start in the 2-way comparison mode *(on page 402)*. 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
-ext $LOCAL $REMOTE $BASE $MERGED'
git config --global mergetool.oxygendiff.trustExitCode true
```

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.

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 *Oxygen* **Compare Files** tool provides the following advantages when using it with Sourcetree:

- The files are presented side-by-side to make it much easier to determine real changes.
- XML comparison algorithms are available.

- Various options can be used to configure the comparison.
- You can navigate through changes, one by one.

To integrate the **Compare Files** tool with Sourcetree, use the following 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* `diffFiles.exe` file (for example: `c:\Programs\Oxygen XML Editor\diffFiles.exe`).
   - **Arguments** - Enter **-ext $REMOTE $LOCAL $LOCAL**.
   - **Merge Tool** - Select **Custom**.
   - **Diff Command** - Enter the path of the *Oxygen* `diffFiles.exe` file (for example: `c:\Programs\Oxygen XML Editor\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

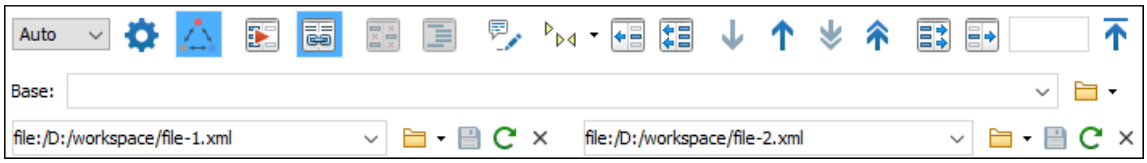
1. In Sourcetree, go to **Sourcetree > Preferences**.
2. Go to the **Diff** tab.
3. In the **External Diff/Merge** section, configure the settings as follows:
   - **External Diff Tool** - Select **Custom**.
   - **Diff Command** - Enter a command-line argument to launch the *Oxygen* `diffFiles.sh` file (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 89. 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.
- **Words** - Computes the differences at word level, meaning that it compares two files or fragments looking for identical words.
- **Lines** - Computes the differences at line level, meaning that it compares two files or fragments looking for identical lines of text.
- **Syntax Aware** - Computes differences for the file types or fragments known by Oxygen XML Developer, taking the syntax (the specific types of tokens) into consideration.
- **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 215)* 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.

 **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. If content is selected in the left panel, only that selection is affected, copying only the corresponding changes from the right to the selection on the left. The action name is also automatically updated to **Copy Changes in Selection from Right to Left** in this case.

 **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.

 **Copy All Changes from Left to Right**

Copies all changes from the file in the left panel to the file in the right panel. If content is selected in the right panel, only that selection is affected, copying only the corresponding changes from the left to the selection on the right. The action name is also automatically updated to **Copy Changes in Selection from Left to Right** in this case.

 **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 1726)* 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 217)* 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 405)*. 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 Developer 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 405)*.

Base > Open URL

Browses for a remote file that will be compared with both files in a three-way comparison *(on page 405)*.

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 405)*.

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 436)*.

Save Comparison as Document with Tracked Changes (Available for two-way comparison in Author mode only)

Allows you to merge two compared documents based on the differences detected and save the results as a specified file that includes the special change tracking marks. You can load the resulting file in **Oxygen's Author** mode to review the changes that resulted from the merge process and you can accept or reject them. Note that if the documents to be compared already contain tracked changes, they will be automatically accepted before generating the output file.

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. If content is selected in the left panel, only that selection is affected, copying only the corresponding changes from the right to the selection on the left. The action name is also automatically updated to **Copy Changes in Selection from Right to Left** in this case.

Copy All Changes from Left to Right

Copies all changes from the file in the left panel to the file in the right panel. If content is selected in the right panel, only that selection is affected, copying only the corresponding changes from the left to the selection on the right. The action name is also automatically updated to **Copy Changes in Selection from Left to Right** in this case.

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 Developer 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.

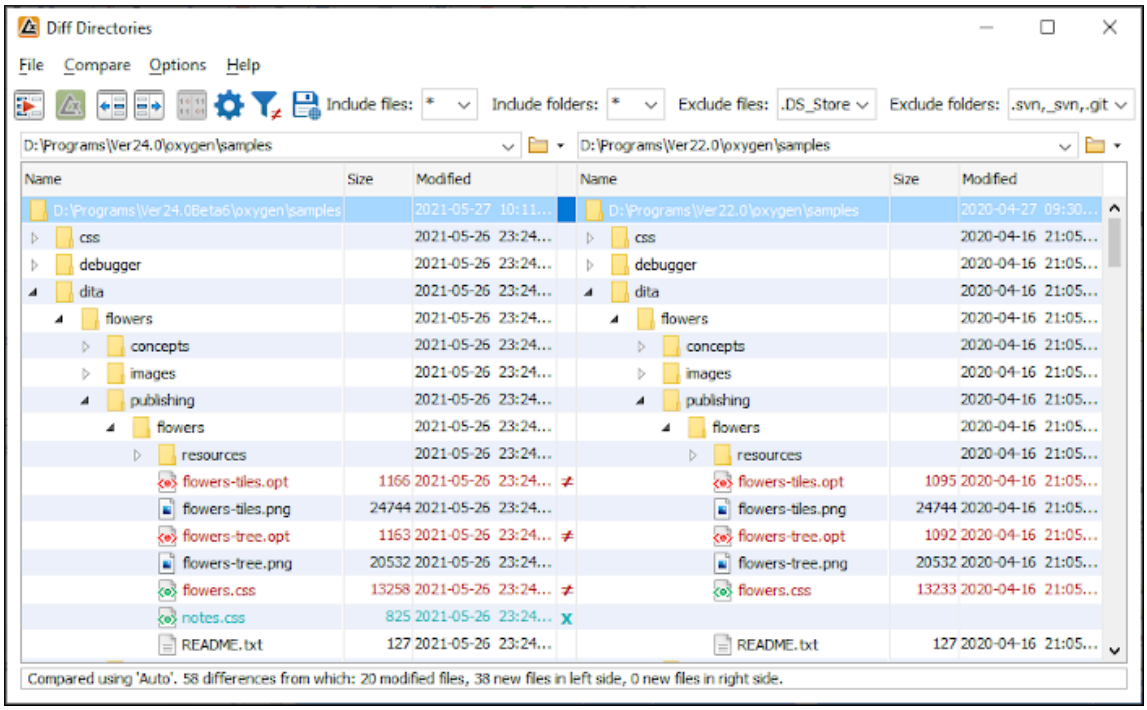
Support Center

Opens the Oxygen XML Developer 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 > Comparison Tools** menu or can be opened as a stand-alone application from the Oxygen XML Developer installation folder (`diffDirs.exe`).

Figure 90. Diff Directories Dialog Box



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 218)* 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 Developer treat supported archives as directories, select the **Look in archives** option *(on page 219)* 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 219)*. 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 Developer starts a file comparison *(on page 402)* between the two files, using the **Compare Files** tool.

Related information

Compare Files Tool *(on page 402)*

Compare Directories Script *(on page 2281)*

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 91. Compare toolbar



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 402)* 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 218)* 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 the following combo boxes: **Include files**, **Include folders**, **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. Each filter includes a drop-down menu with the latest 15 filters applied.

For example, to filter out all `JPEG` and `GIF` image files, edit the **Exclude files** filter box to read `*.jpeg`, `*.png`. To include only a specific sub-folder in the comparison (e.g. `topics/very-recent`), edit the **Include folders** filter to `topi*/ve*rece*`, or `topi*\ve*rece*`, choosing the appropriate **file separator** depending on your operating system.

> **Tip:**
> Instead of directly editing a filtering combo box, you can use **Ctrl+Shift+E (Meta+Shift+E on macOS)** keyboard shortcut when the focus is in the text field to display a specialized filter editor dialog box. You can edit any previously used filter (stored in the history) and create a new one. Furthermore, you can select multiple previous filters and simply combine them (as is) to create a new composite one, or you can edit the result of the multiple selection according to your needs. Also, previous filters can be removed from the history by selecting them and using **Delete (Fn+Backspace on macOS)** on your keyboard.

Contextual Menu Actions

Perform Files Differencing

Opens the **Compare Files** tool *(on page 402)* 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 Developer. 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 Developer.

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 Developer 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 436)*.

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 402)* 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 Developer 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 Developer 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.

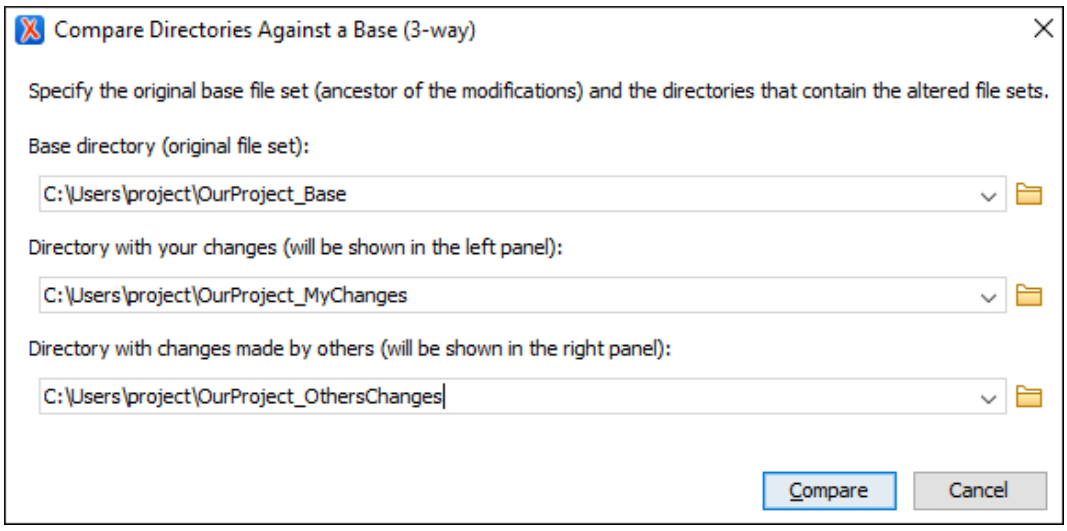
How to Perform 3-Way Directory Comparisons

To perform a 3-way directories comparison, follow these steps:

1. Select **Compare Directories Against a Base (3-way)** from the **Tools > Comparison 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 92. Compare Directories Against a Base File Set Chooser

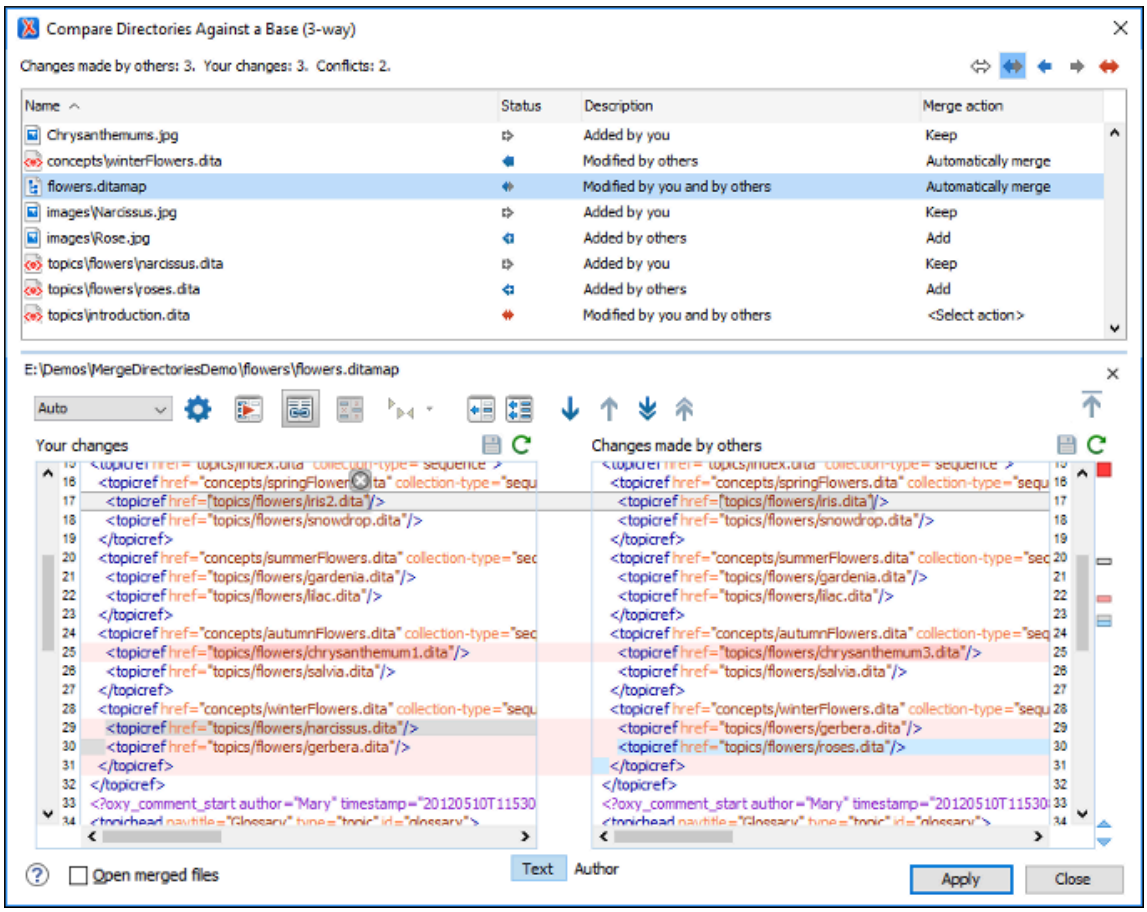
2. Select the file sets to be compared:

- **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 93. 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 Developer 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 Developer 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.
- **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 215)* 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.

 **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. If content is selected in the left panel, only that selection is affected, copying only the corresponding changes from the right to the selection on the left. The action name is also automatically updated to **Copy Changes in Selection from Right to Left** in this case.

 **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:

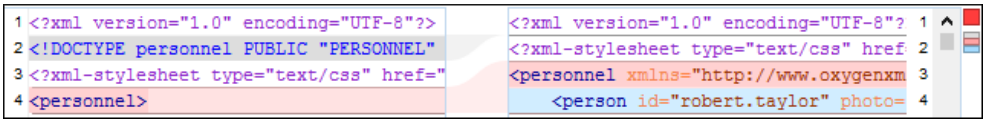
Reload

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 94. 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 217)*, 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.

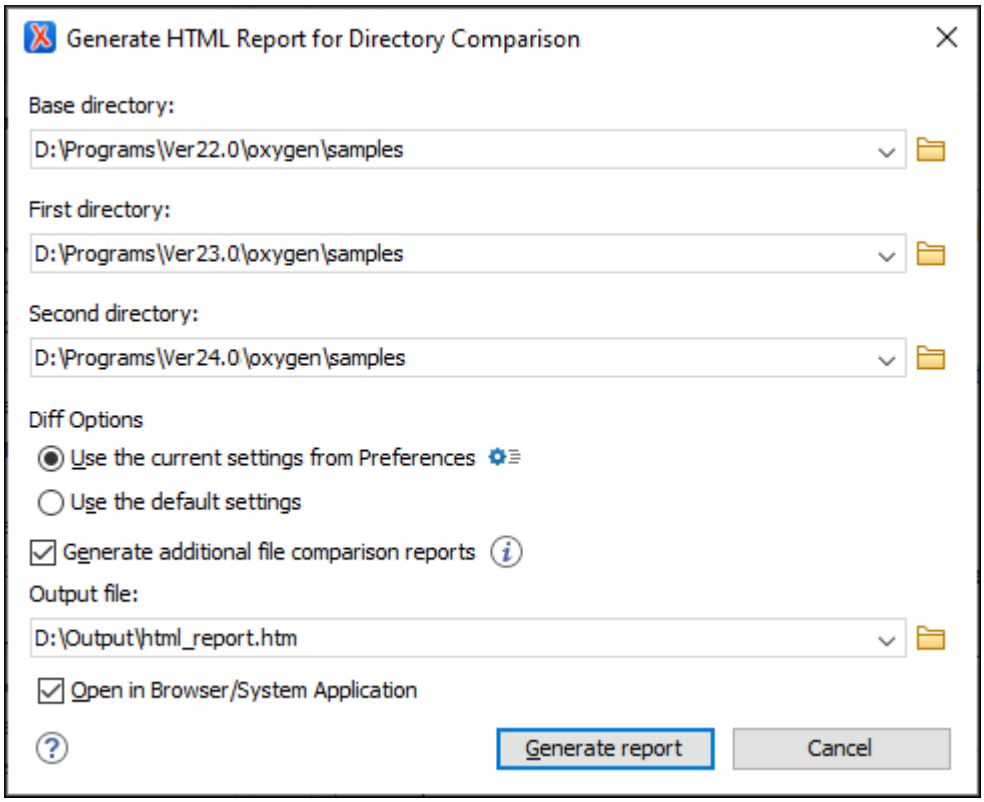
Related information

Compare Directories Tool *(on page 421)*
Compare Files Tool *(on page 402)*

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 > Comparison Tools** menu. It opens a dialog box where you can specify the directories to compare as well as some other options.

Figure 95. Generate HTML Report for Directory Comparison Dialog Box



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 218)* and the include/exclude filter options in the **Compare Directories** tool *(on page 424)* 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 436)*. 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.



Tip:
An XPath expression specified in the **Ignore nodes by XPath** text field within the **Files Comparison** preferences page *(on page 215)* is now taken into account if you enable the **Generate additional file comparison reports** option.

Output file

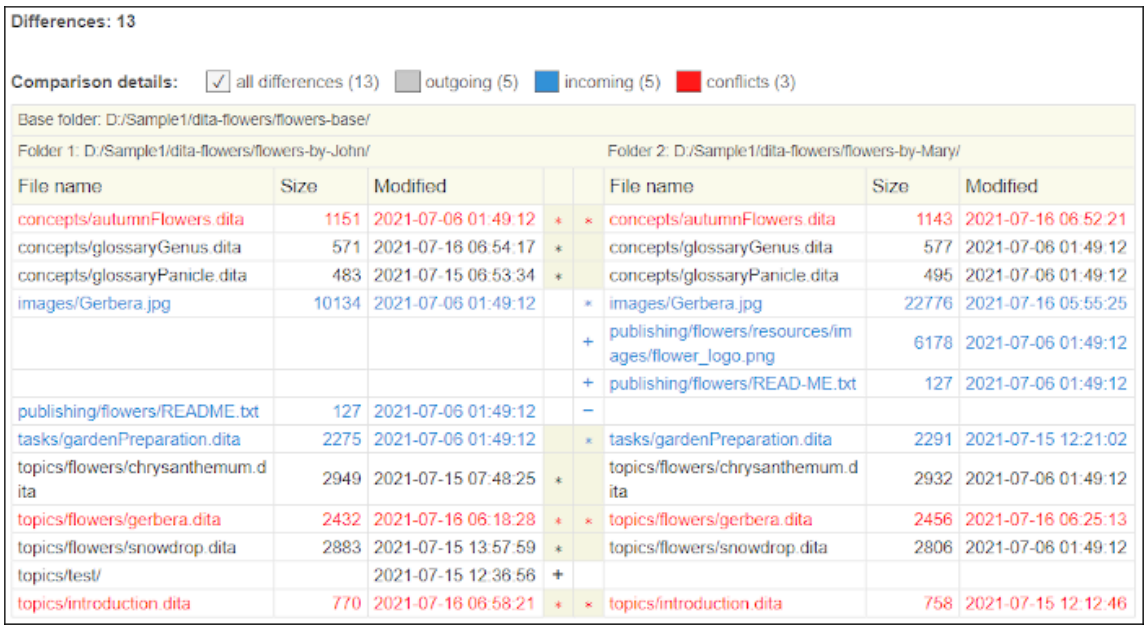
Specifies the path for an output file to save the comparison results file.

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 96. HTML Report for Directory Comparison



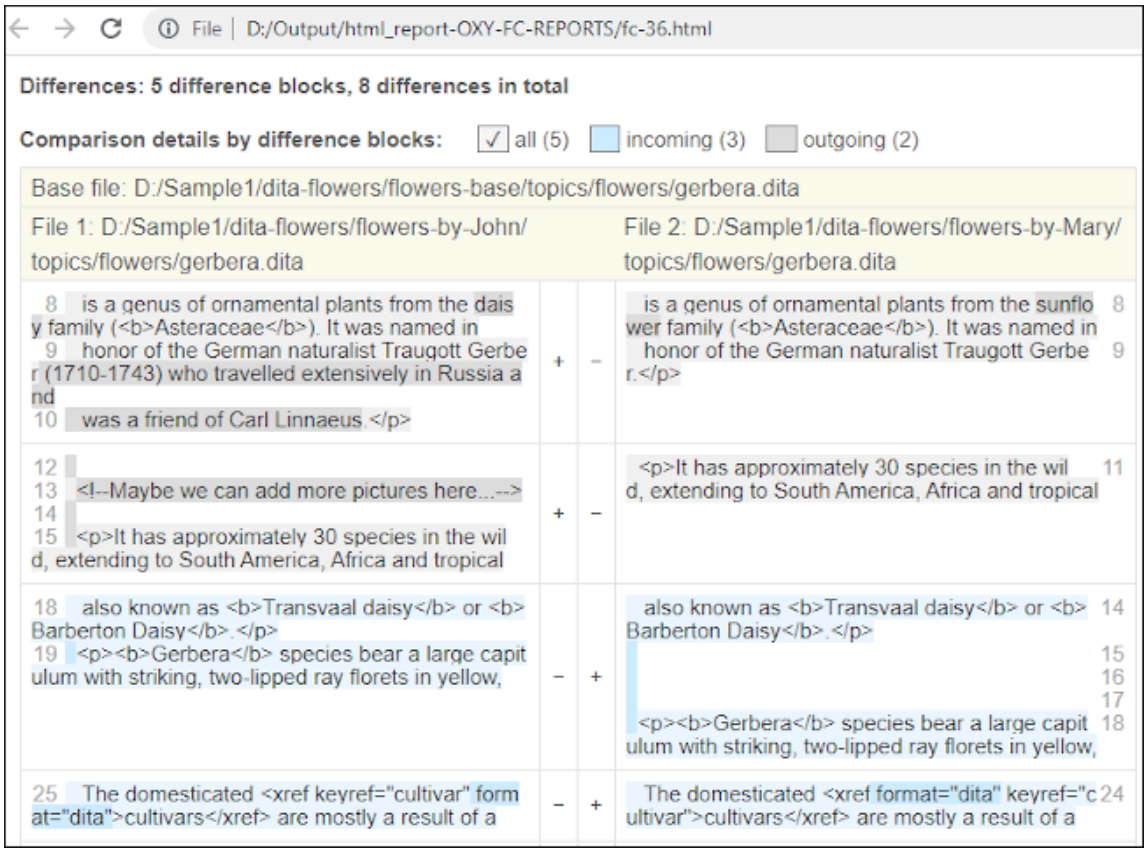
Differences: 13

Comparison details: ☑ all differences (13) ☐ outgoing (5) ☐ incoming (5) ☐ conflicts (3)

Base folder: D:/Sample1/dita-flowers/flowers-base/

Folder 1: D:/Sample1/dita-flowers/flowers-by-John/					Folder 2: D:/Sample1/dita-flowers/flowers-by-Mary/		
File name	Size	Modified			File name	Size	Modified
concepts/autumnFlowers.dita	1151	2021-07-06 01:49:12	*	*	concepts/autumnFlowers.dita	1143	2021-07-16 06:52:21
concepts/glossaryGenus.dita	571	2021-07-16 06:54:17	*		concepts/glossaryGenus.dita	577	2021-07-06 01:49:12
concepts/glossaryPanicle.dita	483	2021-07-15 06:53:34	*		concepts/glossaryPanicle.dita	495	2021-07-06 01:49:12
images/Gerbera.jpg	10134	2021-07-06 01:49:12		*	images/Gerbera.jpg	22776	2021-07-16 05:55:25
				+	publishing/flowers/resources/images/flower_logo.png	6178	2021-07-06 01:49:12
				+	publishing/flowers/READ-ME.txt	127	2021-07-06 01:49:12
publishing/flowers/README.txt	127	2021-07-06 01:49:12		−			
tasks/gardenPreparation.dita	2275	2021-07-06 01:49:12		*	tasks/gardenPreparation.dita	2291	2021-07-15 12:21:02
topics/flowers/chrysanthemum.dita	2949	2021-07-15 07:48:25	*		topics/flowers/chrysanthemum.dita	2932	2021-07-06 01:49:12
topics/flowers/gerbera.dita	2432	2021-07-16 06:18:28	*	*	topics/flowers/gerbera.dita	2456	2021-07-16 06:25:13
topics/flowers/snowdrop.dita	2883	2021-07-15 13:57:59	*		topics/flowers/snowdrop.dita	2806	2021-07-06 01:49:12
topics/test/		2021-07-15 12:36:56	+				
topics/introduction.dita	770	2021-07-16 06:58:21	*	*	topics/introduction.dita	758	2021-07-15 12:12:46

Figure 97. Example of an Additional File Comparison Report



File | D:/Output/html_report-OXY-FC-REPORTS/fc-36.html

Differences: 5 difference blocks, 8 differences in total

Comparison details by difference blocks: ☑ all (5) ☐ incoming (3) ☐ outgoing (2)

Base file: D:/Sample1/dita-flowers/flowers-base/topics/flowers/gerbera.dita

File 1: D:/Sample1/dita-flowers/flowers-by-John/topics/flowers/gerbera.dita			File 2: D:/Sample1/dita-flowers/flowers-by-Mary/topics/flowers/gerbera.dita
8 is a genus of ornamental plants from the daisy family (Asteraceae). It was named in	+	−	is a genus of ornamental plants from the sunflower family (Asteraceae). It was named in 8
9 honor of the German naturalist Traugott Gerber (1710-1743) who travelled extensively in Russia and			honor of the German naturalist Traugott Gerber. 9
10 was a friend of Carl Linnaeus .			
12	+	−	It has approximately 30 species in the wild, extending to South America, Africa and tropical 11
13 <!--Maybe we can add more pictures here...-->			
14			
15 It has approximately 30 species in the wild, extending to South America, Africa and tropical			
18 also known as Transvaal daisy or Barberton Daisy.	−	+	also known as Transvaal daisy or 14
19 Gerbera species bear a large capitulum with striking, two-lipped ray florets in yellow,			Barberton Daisy. 15 16 17
			Gerbera species bear a large capitulum with striking, two-lipped ray florets in yellow, 18
25 The domesticated <xref keyref="cultivar" format="dita">cultivars</xref> are mostly a result of a	−	+	The domesticated <xref format="dita" keyref="cultivar">cultivars</xref> are mostly a result of a 24

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 421)*

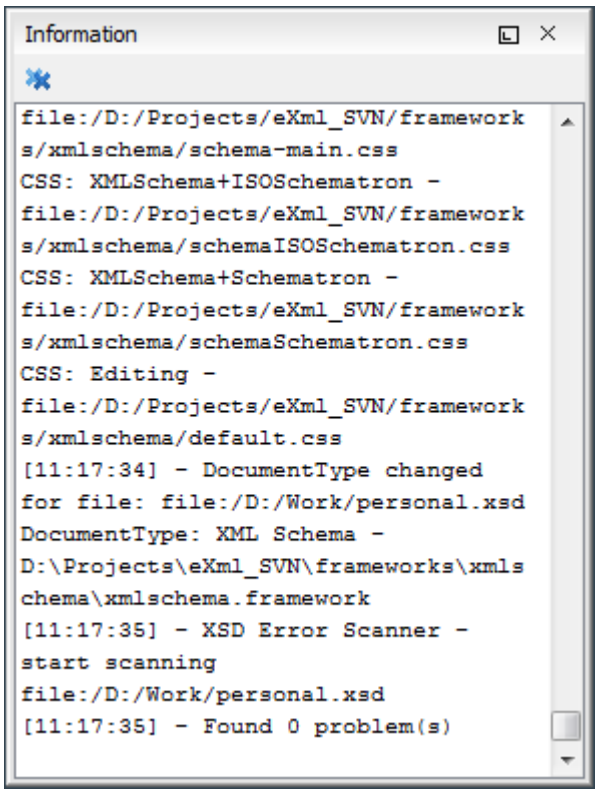
Compare Directories Against a Base (3-Way) Tool *(on page 427)*

Compare Files Tool *(on page 402)*

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 98. Information view messages



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 235)* 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 when the following actions generate results:

- **Find/Replace in Files** *(on page 368)*
- **Find/Replace** *(on page 364)*
- **Open/Find Resource** *(on page 358)*
- **Find All**
- **Find All Elements** *(on page 374)*
- **XPath in Files** *(on page 341)*
- **Search References** *(on page 502)*
- **Search Declarations** *(on page 503)*

By default, Oxygen XML Developer 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 124)*. 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 474)* view 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** view *(on page 474)* view 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.

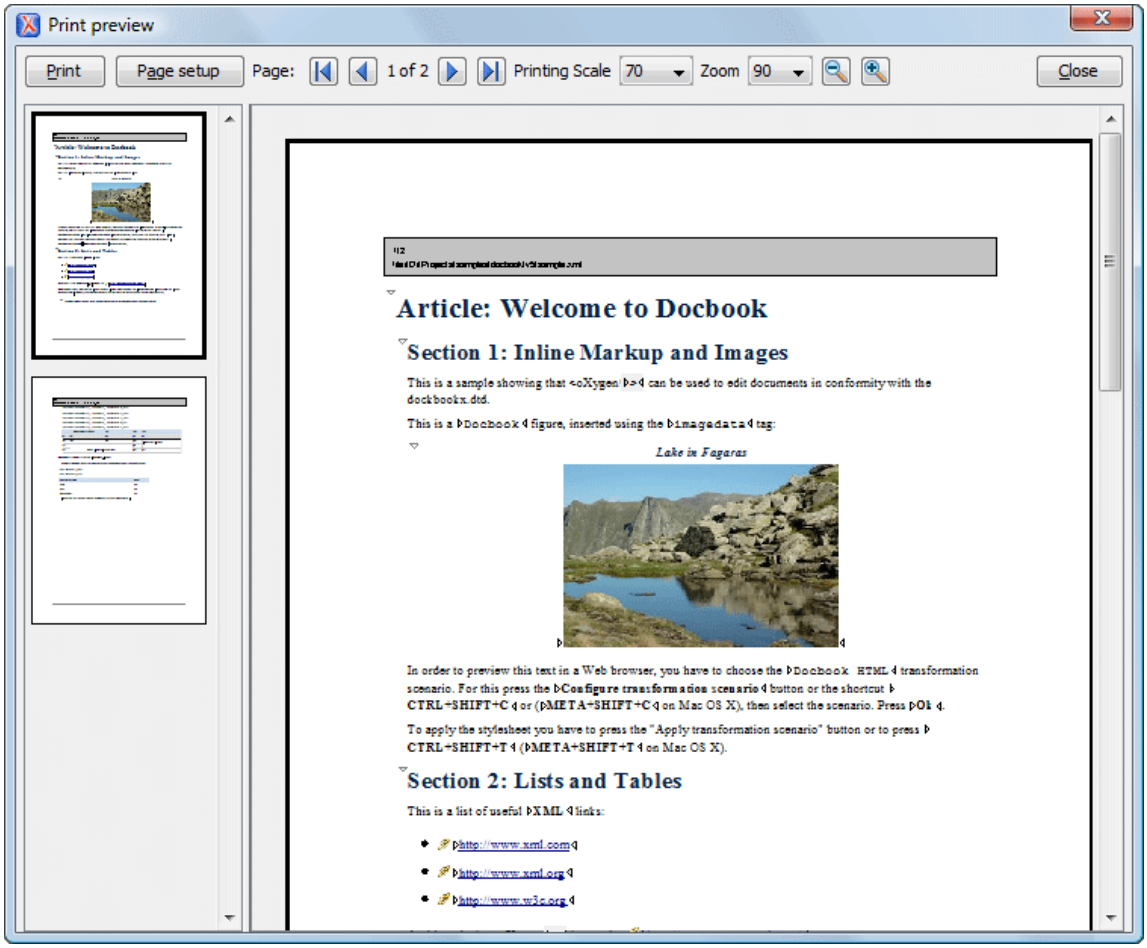
Printing a Document

Printing is supported in **Text** 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 99. 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 **Text** mode and line numbers are displayed (the **Show line numbers** option *(on page 127)* 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 is activated (the **Enable folding** option *(on page 128)* 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 Supported Document Types

Oxygen XML Developer includes built-in *frameworks* for the most popular XML document types (DITA, DocBook, TEI, XHTML, JATS) *(on page 1042)* 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 Developer 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 Developer, including general information about editing XML documents in **Text** mode *(on page 442)*, and **Grid** mode *(on page 505)*.

List of Supported 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
-  - NVDL (Namespace-based Validation Dispatching Language) schemas
-  - XSL:FO documents
-  - XQuery documents
-  - WSDL documents (Deprecated)
-  - Schematron documents
-  - JavaScript documents
-  - Python documents
-  - CSS documents
-  - LESS documents
-  - XProc scripts
-  - SQL documents
-  - JSON documents
-  - JSON Schema documents
-  - YAML documents

-  - Ant build scripts
-  - Markdown documents
- Additional supported document types include: Text, Java, Properties, Batch, Shell, PowerShell, Dockerfile, and PHP.

Related information
Built-in Frameworks (Document Types) *(on page 1042)*

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. For more information about schema association, see Associating a Schema to XML Documents *(on page 554)*.

Oxygen XML Developer includes fully supported built-in *frameworks (on page 2305)* for the most popular XML document types (DITA, DocBook, TEI, XHTML, JATS) *(on page 1042)* 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.

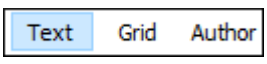
This section includes information about the user interface components and actions that are available in the various editing modes and numerous features to help you edit XML documents in any mode.

Related information
Text Editing Mode *(on page 279)*
Grid Editing Mode *(on page 279)*
Built-in Frameworks (Document Types) *(on page 1042)*

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 Developer **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 Developer also includes numerous specialized editing actions, a powerful *Content Completion Assistant (on page 457)*, a helpful **Outline** view *(on page 465)*, 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 Developer 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 Developer 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

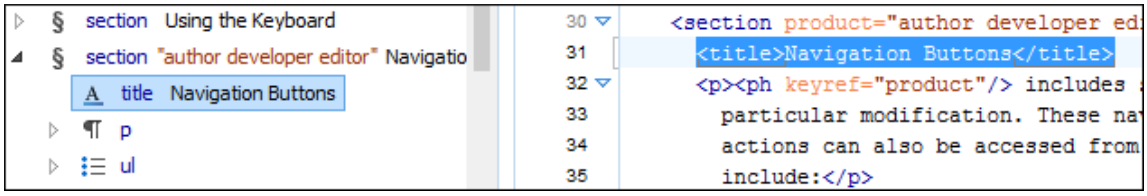
Available after you use the **Back** button at least once, and it navigates in the opposite direction as the **Back** button.

Navigating with the Outline View

Oxygen XML Developer includes an **Outline** view *(on page 465)* 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.

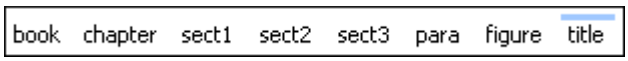
Figure 100. Outline View Navigation in Text Mode



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 101. Breadcrumb in Text Mode

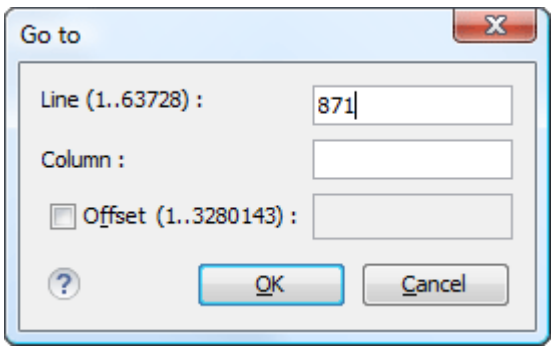


The last element listed in the *breadcrumb* is the element at the current cursor position. The current element is also highlighted by a thin light blue bar for easy identification. Clicking an element from the *breadcrumb* selects the entire element and navigates to it in the editor area.

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** .

Figure 102. Go to Dialog Box



The dialog box includes the following fields for specifying a specific navigation location:

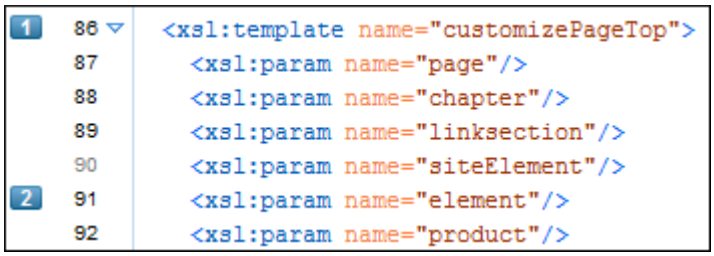
- **Line** - Destination line in the current document.
- **Column** - Destination column in the current document.
- **Offset** - Destination offset relative to the beginning of document.

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 223)* menu.

Figure 103. Editor Bookmarks



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 Developer 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 Developer will mirror-edit the name of the matching end tag. This feature can be controlled from the **Content Completion** option page *(on page 144)*.

- **Auto-breaking the edited line** - The **Hard line wrap** option *(on page 138)* automatically breaks the edited line when its length exceeds the maximum line length defined for the format and indent operation *(on page 138)*.
- **Indent on Enter** - The **Indent on Enter** option *(on page 137)* indents the new line inserted when you press **Enter**.
- **Smart Enter** - The **Smart Enter** option *(on page 137)* 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 Developer 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 126)* 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 126)* 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 88)*.

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 125)* 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.

The clipboard contains the precise node structure that is preserved when pasting to another location within the same document or to another document opened in the **Author** mode. The copied content contains an HTML flavor that attempts to convert the copied structure to plain HTML, which can be useful for preserving the styles when pasting into editors such as MS Word.

Copy as XML

Copies the selection as a string that contains the structured XML content.

Copy as Markdown

Copies the selection as a string that contains its Markdown-converted equivalent.

 **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 + Option + UpArrow on macOS)

Moves the node up one line.

Ctrl + Alt + DownArrow (Command + Option + 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 Developer 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 293)*

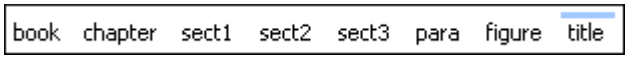
Editing XML Markup in Text Mode

Oxygen XML Developer 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 104. 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 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 a 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 (Option + UpArrow on macOS)

Moves the current node or selected nodes in front of the previous node.

Alt + DownArrow (Option + DownArrow on macOS)

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 Developer 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 145)* 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 145)* 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 + Option + 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 + Option + 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 + Option + 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.

Related information

Refactoring XML Documents *(on page 578)*
Contextual Menu Actions in Text Mode *(on page 492)*
Frequently Used Shortcut Keys *(on page 293)*

Folding XML Elements in Text Mode

When working with a large document, the *folding (on page 2305)* support in Oxygen XML Developer 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 2305)* feature is enabled in Oxygen XML Developer, but it can be disabled in the **Text** preferences page with the **Enable folding** option *(on page 128)*.

Figure 105. Folding of XML Elements in Text Mode



The fact that the folds are persistent is a unique feature of Oxygen XML Developer. 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

Folds all the elements except the current element.

Collapse Child Folds

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

Unfolds all elements in the current document.

Resources

For more information about the *folding* support in Oxygen XML Developer, 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 Developer 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 446)* 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 446)* 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 Developer 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 128)* is selected in the **Text** preferences page.

The following shortcuts can be used to create a rectangular selection:

Alt + Mouse Click + Mouse Movement (Option + Meta + Mouse Click + Mouse Movement on macOS)

Begins 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 457)*.

Shift + Alt + Left/Right Arrow Keys (Shift + Option + Meta + Left/Right Arrow Keys on macOS)

Begins 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 + Option + Meta + Up/Down Arrow Keys on macOS)

Begins 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 + Option + Meta + Left/Right Arrow Keys on macOS)

Begins 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 + Option + Meta + Home on macOS)

Begins a rectangular selection at the cursor position and extends it to the beginning of the current line.

Shift + Alt + End (Shift + Option + Meta + End on macOS)

Begins a rectangular selection at the cursor position and extends it to the end of the current line.

Shift + Alt + PageUp (Shift + Option + Meta + PageUp on macOS)

Begins a rectangular selection at the cursor position and extends it up one screen page.

Shift + Alt + PageDown (Shift + Option + Meta + PageDown on macOS)

Begins 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

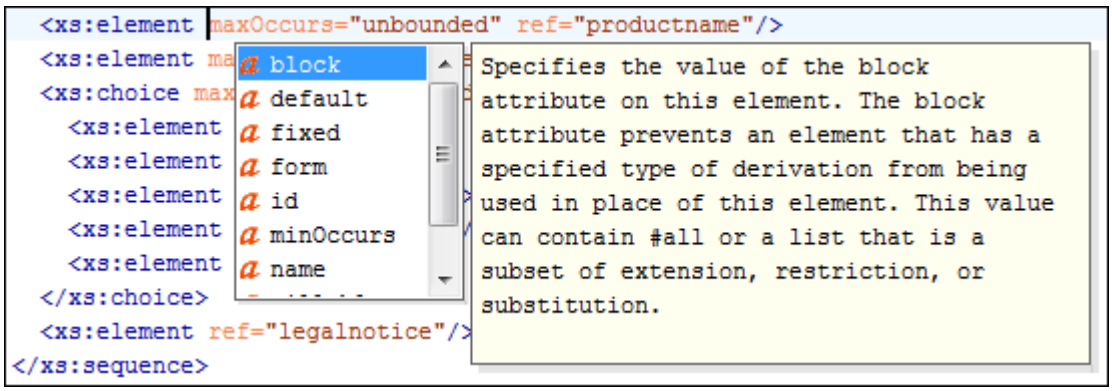
To edit the content of the rectangular selection, you can enter an in-place editing mode by releasing the **Alt** key (on macOS, release both **Alt** & **Meta**). Once you are in the editing mode, you can simply use your keyboard to edit the entire selection of content, or click anywhere inside the selection 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 Developer includes an intelligent *Content Completion Assistant (on page 2303)* 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)** *(on page 79)*, go to **Editor > Content Completion**, and deselect the **Enable content completion** option *(on page 144)*.

Figure 106. 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 (on page 2303)* 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 *(on page 554)* 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 *(on page 146)* from the **Content Completion** preferences page.
- After typing a partial element or attribute name, you can manually activate it by pressing **Ctrl + Space** or **Alt + ForwardSlash (Command + Option + 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 145)* (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 145)* and **Add first Choice particle** *(on page 145)* 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 Developer 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*.

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 2303)*.
- 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 150)* 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 150)* 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:

```
<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 Developer 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:

```
<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 151)* 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:

```
##  Information about a person.

element person { name, email*}
```

Collecting Annotation from DTDs

For DTD, Oxygen XML Developer defines a custom mechanism for annotations using comments enabled by the **Prefer DTD comments that start with "doc:" as annotations** *(on page 150)* 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 2304)* views, such as the **Model** view *(on page 470)*, **Attributes** view *(on page 468)*, **Elements** view *(on page 472)*, and **Entities** view *(on page 472)*. 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 465)*, 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 Developer includes a set of built-in code templates for CSS, 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** 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.

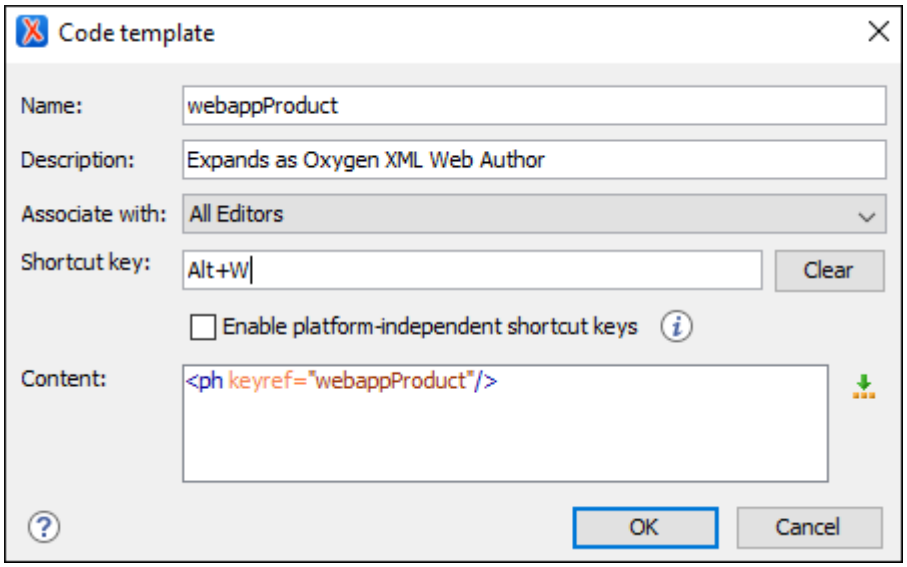
How to Create Code Templates

To create a code template, follow these steps:

1. Open the **Preferences** dialog box **(Options > Preferences)** *(on page 79)* 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 107. Code Template Configuration Dialog Box



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 2303)*. 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 252)* 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 2303)* (**Enter** in **Author** mode or **Ctrl + Space** 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 79)* 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 79)*, 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 79)* 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 , the code templates will be available in their content completion list.

Text Mode Views

There is a variety of *dockable (on page 2304)* 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.

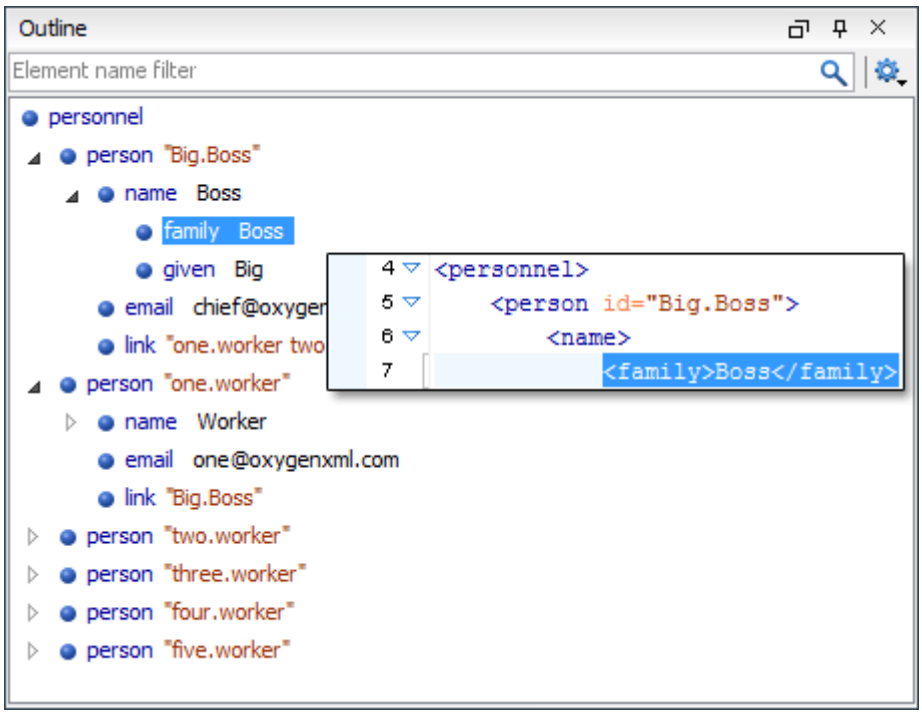
Drag and Drop Actions in the Outline View

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 108. 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 235)*.

Configure displayed attributes

Displays the **XML Structured Outline** preferences page *(on page 235)*.

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.

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.

Toggle Comment

Encloses the currently selected element in a comment, if the element is not already commented. If it is already commented, this action will remove the comment.

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:

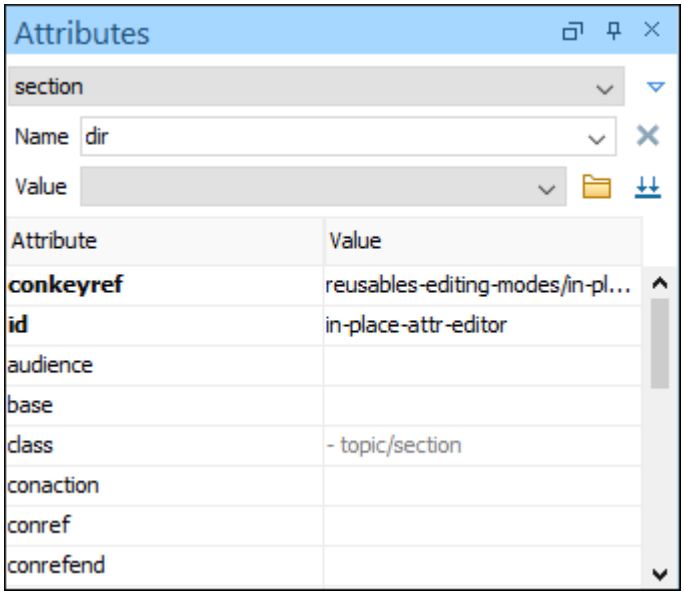
- The names of the attributes are rendered with a bold font, and their values with a plain font.
- Default values are rendered with a plain font, painted gray.
- Empty values display the text "[empty]", painted gray.
- Invalid attributes and values are painted red.

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 109. 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.

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

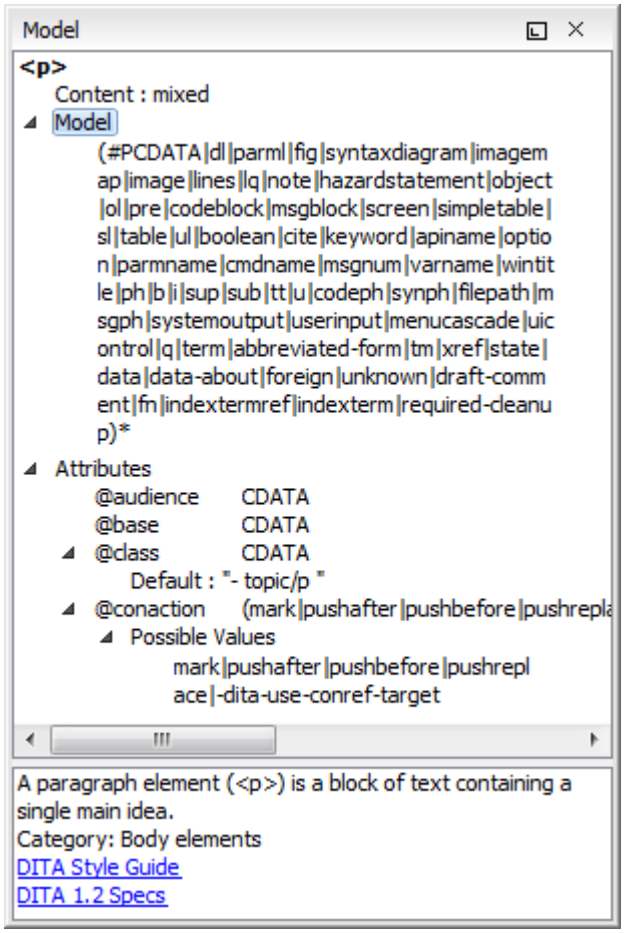
Depending on the content of the clipboard, the following cases are possible:

- 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 110. Model View

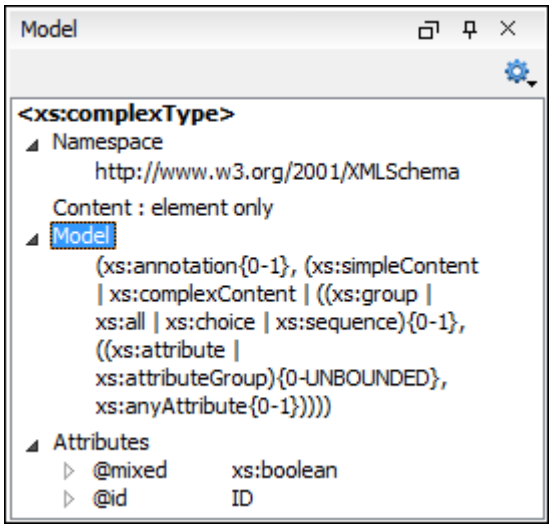


The **Model** view is comprised of two sections, an element structure panel and an annotations panel.

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.

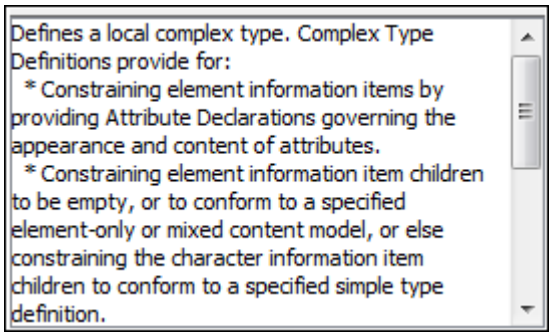
Figure 111. Element Structure Panel



Annotation Panel

The **Annotation** panel displays the annotation information for the currently selected element. This information is collected from the XML schema.

Figure 112. Annotation panel

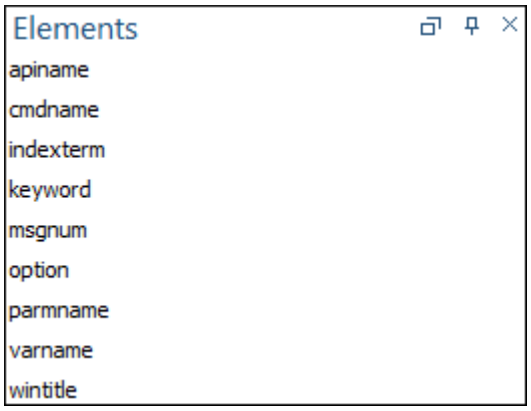


Elements View in Text Mode

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.

Figure 113. Elements View in Text 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:

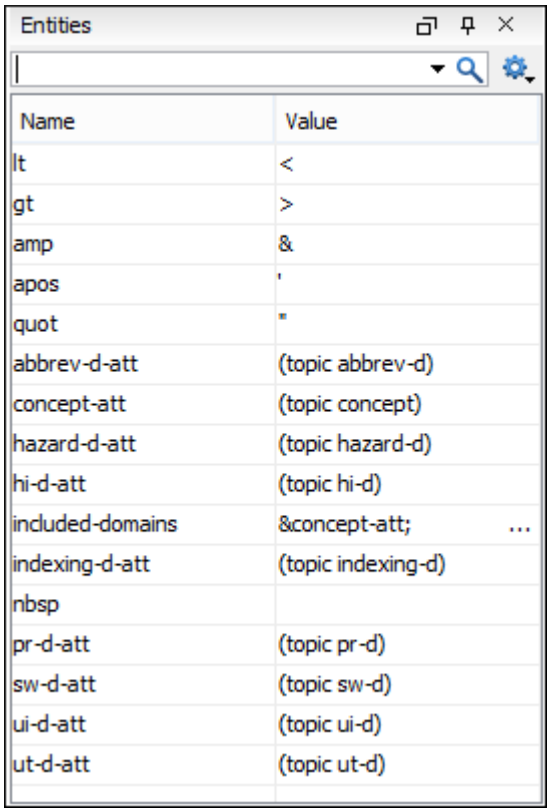
- **Predefined** - Entities that are part of the predefined XML markup (`<`, `>`, `&`, `'`, `"`).
- **Internal** - Defined in the DOCTYPE declaration header of the current XML.
- **External** - Defined in an external DTD module included in the DTD referenced in the XML DOCTYPE declaration.

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 114. Entities View



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

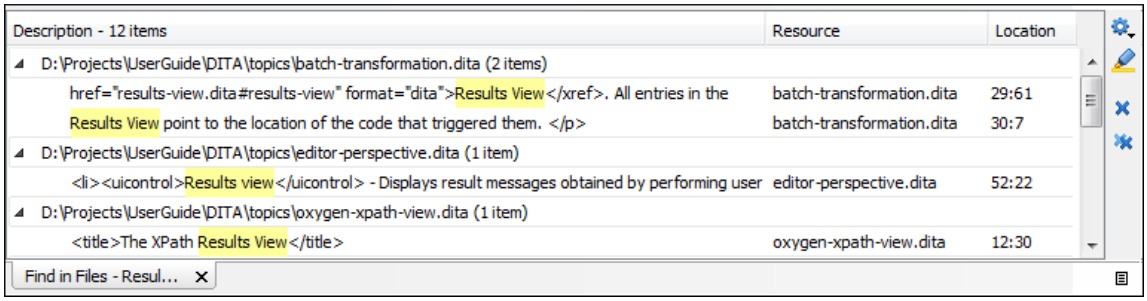
The **Results** view displays the messages generated as a result of user actions such as validations, transformations, search operations, and others. Each message is a link to the location related to the event that triggered the message. Double-clicking a message opens the file containing the location 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 2304)*, 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 223)* to this action.

The actions that contribute messages to this view include:

- **Validation** actions *(on page 517)*
- **Transformation** actions *(on page 1070)*
- **Check Spelling in Files** action *(on page 389)*
- **Find All** action from the **Find/Replace** dialog box *(on page 364)*
- **Find/Replace in Files** dialog box *(on page 368)*
- **Search References** action *(on page 651)*
- XPath expression results *(on page 1727)*
- SQL results *(on page 1794)*

Figure 115. 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 389)*.
- No grouping rule for the results of applying an XPath expression *(on page 1726)*.

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 549)*, 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 Developer 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 79)* 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.

Apply all default quick fix proposals

This action is available for the tabs in the **Results** view that displays the problems reported after a single file or batch validation operation. In the first phase, the quick fix proposals associated with the presented validation errors are automatically identified and mapped to the

corresponding document (if multiple quick fix proposals are available for the same validation error, only the first one is considered). Depending on the number of quick fixes to be applied, the next phase analyzes the impact of their application and may take some time (an operation progress tracker shows the status).

Once complete, a preview dialog box is presented that provides an overview of the content changes that will be made for each document. The comparison panel also informs you of any problems encountered. In addition, you can choose to exclude certain files from the set of files that the quick fixes will be applied to if you do not agree with the proposed changes to be made. If you do agree with the changes presented, click **Apply** to trigger the procedure of applying the quick fixes and updating the contents of the documents.

> **Important:**
> Oxygen XML Developer does not provide an automatic means for reverting changes made by this action so make sure you agree with the previewed changed for applying them.

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 2306)* (in the case of a validation scenario *(on page 525)*, 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 525)*.
- 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 225)* 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 477)*. 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 477)*. 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 477)*.

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 477)*.

Save Validation Report (Available for validation errors)

Unlike the **Save Results as HTML** action, where the validation results are formatted in a single table, the HTML report created by this action groups the detected errors by the system IDs of the validated documents and creates a table for each. For greater flexibility in analyzing the results, most of its columns are sortable. Also, links are provided for quickly viewing both the validated document and its associated schema, and links to additional information about the errors detected, when available.

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 389)*.
- No grouping rule for the results of applying an XPath expression *(on page 1726)*.

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 474)* displays the results from the following operations:

- Document validation *(on page 516)*
- Checking the form of documents *(on page 515)*
- XSLT or FO transformations *(on page 1070)*
- Finding all occurrences of a string in a file *(on page 364)*
- Finding all occurrences of a string in multiple files *(on page 368)*
- Applying an XPath expression to the current document *(on page 1729)*

To make a persistent copy of the **Results** view *(on page 474)*, 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:

```
<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>
            </start>
        </Location>
    </Incident>
</Report>
```

Related Information:
Results View *(on page 474)*

Syntax Highlighting in XML Documents

Oxygen XML Developer 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 79)*.
2. Go to **Editor > Syntax Highlight** *(on page 158)*.
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 Developer 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 159)*.

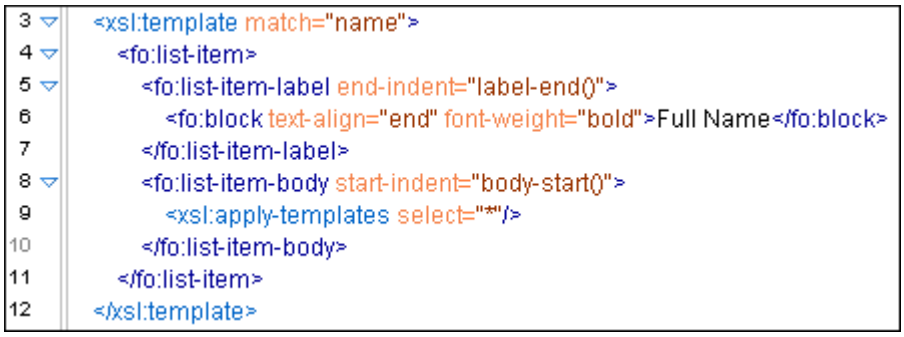
Related Information:
Customize Syntax Highlight colors *(on page 158)*

Syntax Highlight Depending on Namespace Prefix

The syntax highlight scheme of an XML file type *(on page 158)* 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 159)* allows easier identification of the tags.

Figure 116. Example of Coloring XML Tags by Prefix



Related Information:

Changing the colors displayed in the Text Mode Editor *(on page 158)*

Formatting and Indenting XML Documents

In **Text** mode *(on page 279)*, you can decide how the XML file is formatted and indented. In the other modes, and when you switch between modes, Oxygen XML Developer 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 Developer 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:

```
<p>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 <xref
href="http://en.wikipedia.org/wiki/Law_of_the_land" format="html"
scope="external">Law of the land</xref>.
We will sell to no man, we will not deny to any man either Justice or Right.</p>
```

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:

```
<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 Developer Determines When Whitespace is Significant

When Oxygen XML Developer 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 Developer 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 Developer 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:

```
<p>The file is located in <i>HOME</i>/<i>USER</i>/hello.

     This is a <strong>big</strong>


<emphasis>deal</emphasis>.

</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 Developer 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 Developer 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 Developer 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.

Schema-aware formatting

If a schema is available for the XML document, Oxygen XML Developer 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 the **Text** editing mode,open the **Preferences** dialog box **(Options > Preferences)** *(on page 79)*, go to **Editor > Format > XML**, and select/deselect the **Schema-aware format and indent** option *(on page 141)*.

Preserve space elements list

If an element is listed in the **Preserve space** tab of the **Element Spacing** list *(on page 140)* in the XML formatting preferences *(on page 139)*, 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 140)* in the XML formatting preferences *(on page 139)*, 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 140)* in the XML formatting preferences *(on page 139)*, 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.

How Oxygen XML Developer formats and indents XML

You can control how Oxygen XML Developer 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 136)*
- **XML Formatting** preferences page *(on page 139)*
- **Whitespaces** preferences page *(on page 142)*

When Oxygen XML Developer formats and indents XML

Oxygen XML Developer 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 **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 130)* is selected in the **Grid** preferences page.

Setting an Indent Size to Zero

Oxygen XML Developer will automatically format and indent *(on page 481)* 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 Developer 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 Developer to maintain a zero indent when editing or formatting those documents:

1. Open the **Preferences** dialog box **(Options > Preferences)** *(on page 79)* and go to **Editor > Format** *(on page 136)*.
2. Select **Detect indent on open**.
3. Select **Use zero-indent if detected**.

Oxygen XML Developer 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 Developer 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 79)* and go to **Editor > Format** *(on page 136)*.
2. Deselect **Detect indent on open**.
3. Set **Indent size** to `0`.

All documents will be formatted and indented with an indent of zero.



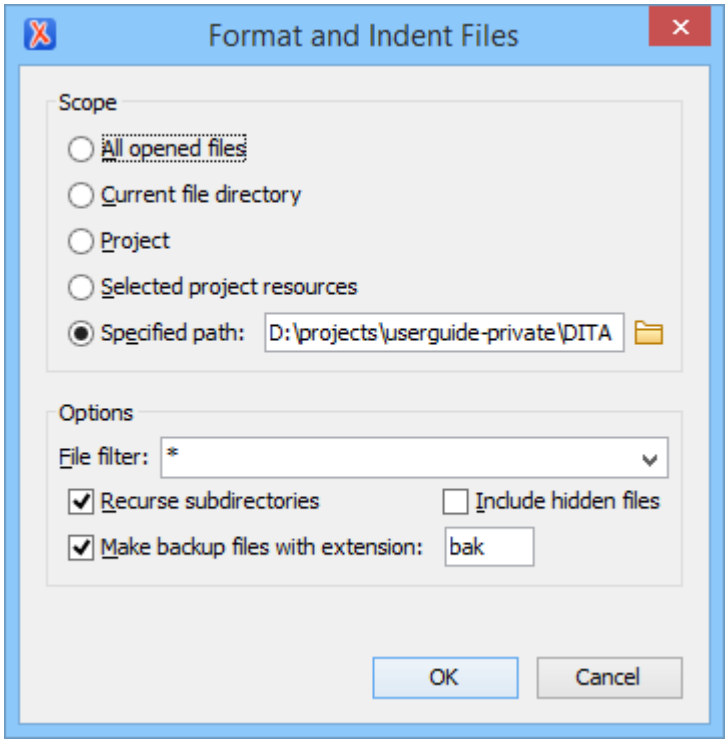
Warning:
Setting the indent size to zero can change the meaning of some file types, such as Python source files.

Format and Indent (Pretty-Print) Multiple Files

Oxygen XML Developer provides support for formatting and indenting (*pretty-print (on page 2307)*) 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 334)* or from the **Tools** menu. This opens the **Format and Indent Files** dialog box that allows you to configure options for the action.

Figure 117. Format and Indent Files Dialog Box



The **Scope** section allows you to choose from the following scopes:

- **All opened files** - The *pretty-print (on page 2307)* 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 2307)* 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 2307)* 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 2307)* is also performed in the hidden files.
- **Make backup files with extension** - When selected, Oxygen XML Developer 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 Developer 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** 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 364)*, 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.

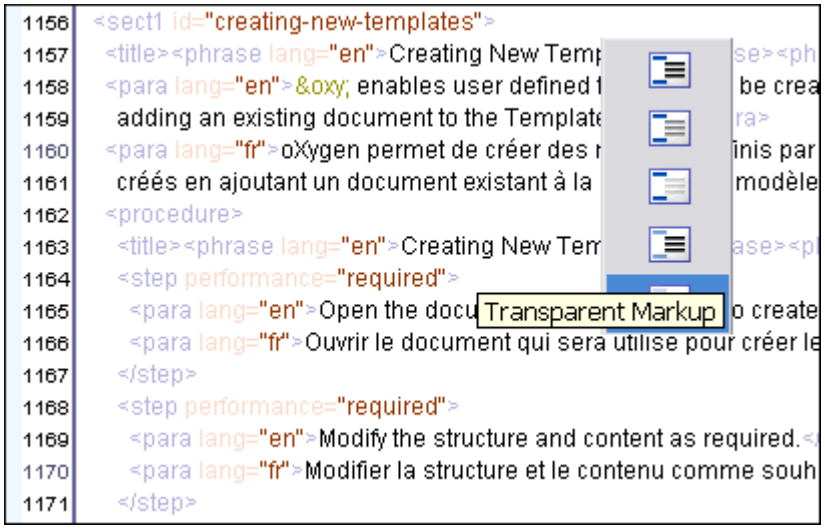
Adjusting the Transparency of XML Markup

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 Developer 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 291)*. 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 118. Tags Transparency Selector



Locking and Unlocking XML Markup

For documents with fixed markup, such as forms that do now 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 128)*.

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 Developer 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 133)*.

Enabled

If you choose to enable support for special characters and as long as you chose a font *(on page 88)* 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 Ậ 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 454)* 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 397)* since disabling the bidirectional editing support can enhance performance.

Example: The Ậ 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 Grid Mode *(on page 512)*
Inserting Special Characters with the Character Map *(on page 394)*

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 397)*.

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 402)*.

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 2308)* 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 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. 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** 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** dialog box, this scope will be used instead.

Change scope

Opens the **Select the scope for the Search and Refactor operations** *(on page 569)* 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 567)*

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 79)* and go to **Editor > Mark Occurrences** *(on page 159)*.

Related Information:
Modular Contextual XML Editing Using 'Main Files' Support *(on page 567)*

Contextual Menu Actions in Text Mode

When editing XML documents in **Text** mode, Oxygen XML Developer provides the following actions in the contextual menu (many of them also appear in the submenus of the **Document** menu):

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 136)*.

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 2307) 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 391)*. 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 394)*.

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 124)* 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 236)*.

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 124)* 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 236)*.

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 124)* 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 236)*.

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 124)* 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 236)*.

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 124)* 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 236)*.

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 124)* 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 236)*.

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 + Option + 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:

```
<!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 2303)* 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 Developer 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 145)* 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 145)* 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 + Option + X on macOS))

Deletes the start and end tag of the current element.

Split element (Alt + Shift + D (Ctrl + Option + 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 + Option + 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

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. 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** 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 569)*.

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** 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 569)*.

Search Occurrences in file

Searches for the declaration and references of the ID in the current document.

Quick Fix/Assist (Alt + 1 (Command + Option + 1 on macOS))

When the cursor is inside the value of an `@id` or `@idref` attribute, this action opens the *Quick Fix/Assist (on page 2308)* window with some search and refactoring actions available for the ID or IDREF. When the contextual menu is invoked on an error where Oxygen XML Developer can provide a Quick Fix *(on page 551)*, this action open the *Quick Fix/Assist (on page 2308)* window with a list of quick fix suggestions to choose from.

Apply all default quick fix proposals

This action is available when invoking the contextual menu (right-click) on a document that has one or more quick fix proposals detected for reported validation errors. If multiple quick fixes are available for the same validation error, the default quick fix defined in the Schematron validation schema using the `sqf:default-fix` attribute is automatically selected to be applied. If no default quick fix is explicitly specified, the first quick fix proposal in the list is considered. All quick fix proposals are then automatically executed in bulk, one after the other.

Important Notes to Consider:

- To maintain the accuracy of the initially calculated error validation ranges, the quick fix proposals are applied in the reverse order of their selection.
- If two or more quick fixes act on the same "area" within the document, only one is applied (no changes can be made to changes already made).

- Quick fixes that involve "user-entered values" that normally present a dialog box to facilitate data entry will not be executed (the automatic process of applying all selected quick fixes cannot be interrupted by the presence of the respective dialog boxes).

Once the analysis of the impact of applying the quick fixes on the content is complete, a preview dialog box is presented that provides an overview of the content changes that will be made, according to the quick fixes that will be applied. If you agree with the changes presented, click **Apply** to trigger the quick fixes and update the content.

Tip:
This action is also available in the contextual menu when right-clicking the status bar at the bottom of the editor (if the currently highlighted validation issue has at least one detected quick fix suggestion available). Specific quick fix suggestions that are defined for the particular validation issue are also available in this menu.

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 397)*.

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 402)*.

Show referenced resources

Opens the **Referenced/Dependent Resources** view *(on page 570)* that allows you to see the referenced resource hierarchy for an XML document.

Show dependent resources

Opens the **Referenced/Dependent Resources** view *(on page 570)* 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 Developer 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 (on page 2303)* is also available to help you edit or insert XML elements.

Resources

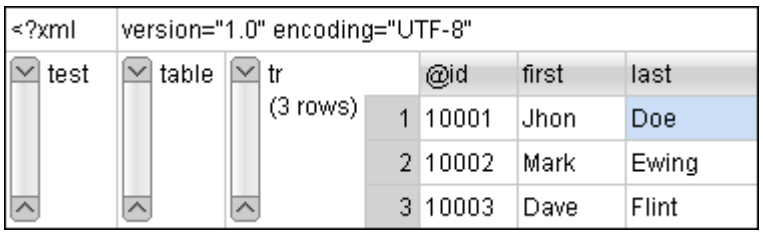
For more information about some of the features available in the **Grid** editor, watch our video demonstration:

https://www.youtube.com/embed/PoYm2VqisWk

Layouts: Grid and Tree

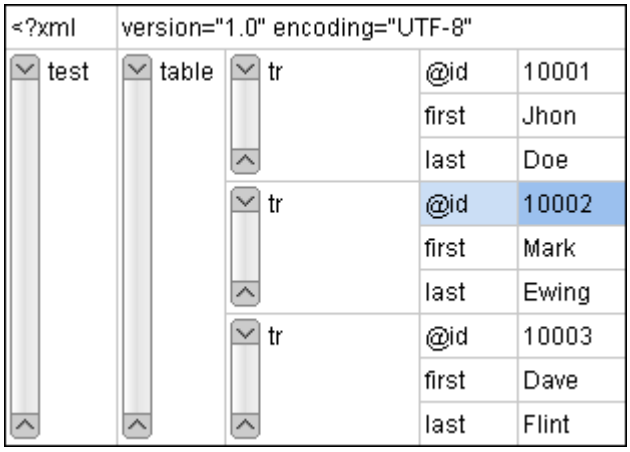
The **Grid** editor offers two layout modes. The default one is the grid layout. This smart layout detects the recurring elements in the XML document and creates tables having the children (including the attributes) of these elements as columns. This way, it is possible to have tables nested in other tables, reflecting the structure of your document.

Figure 119. Grid Layout



The other layout mode is tree-like. It does not create any tables and it only presents the structure of the document.

Figure 120. Tree Layout



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 5. Shortcuts in the Grid Mode

Key	Action
Tab	Moves the cursor to the next editable value in a table row.
Shift + Tab	Moves the cursor to the previous editable value in a table row.
Enter	Begins editing and lets you insert a new value. Also commits the changes after you finish editing.
UpArrow/PageUp	Navigates toward the beginning of the document.
DownArrow/PageDown	Navigates toward the end of the document.
Shift	Used in conjunction with the navigation keys to create a continuous selection area.
Ctrl (Command on macOS) key	Used in conjunction with the mouse cursor to create discontinuous selection areas.

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 507)*

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 512)* 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 Developer 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 512)*. 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 Developer 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.

Refresh Selected

Forces the layout to be recomputed.

Related Information:

Grid Mode Navigation *(on page 506)*
Copy and Paste in the Grid Editing Mode *(on page 510)*
Drag and Drop in the Grid Editing Mode *(on page 509)*
Content Completion Assistant in Grid Mode *(on page 512)*

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

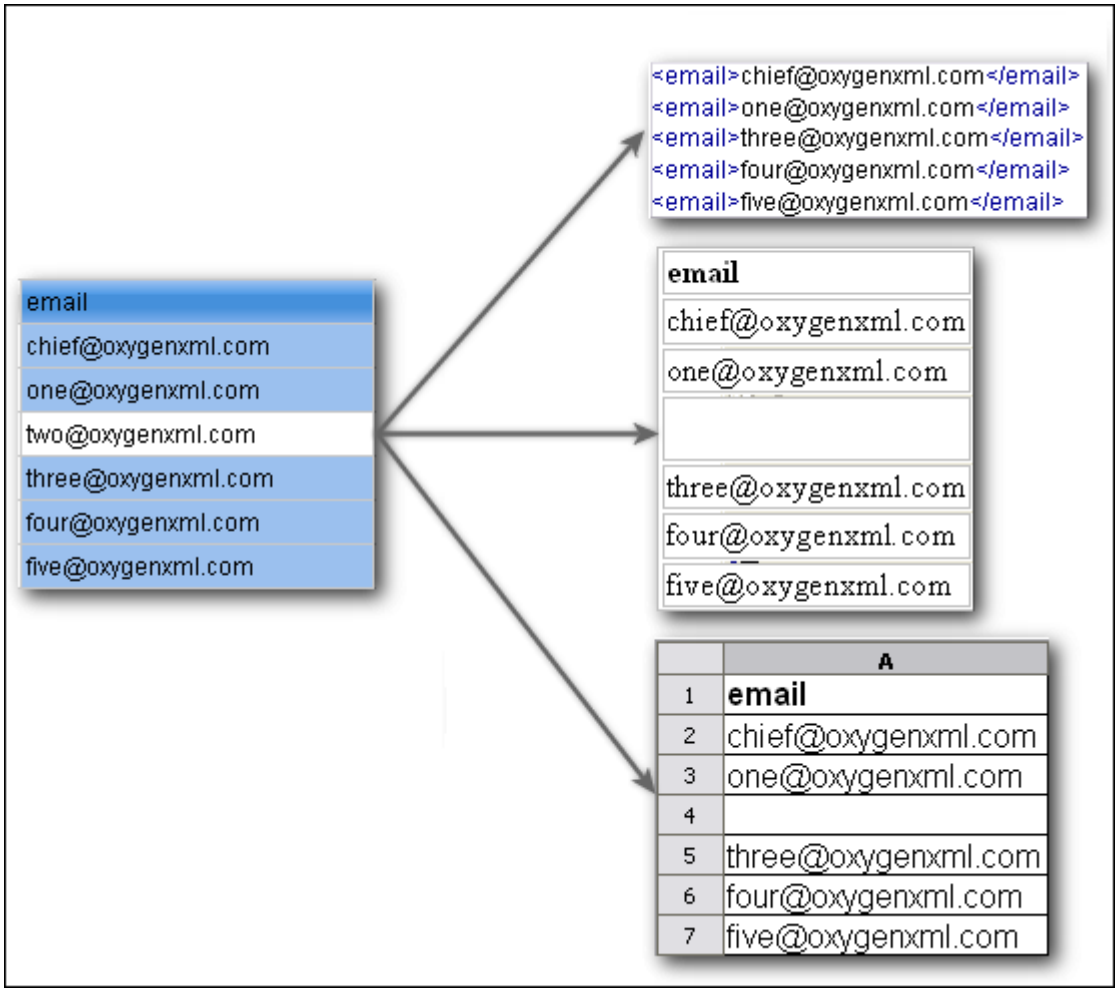
You can paste copied nodes relative to the currently selected cell using one of the following actions (available in the contextual menu):

- **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.

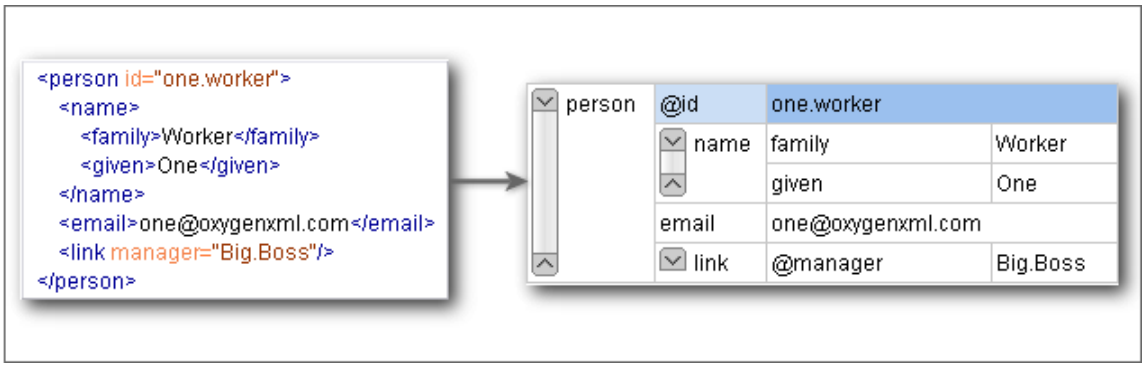
Figure 121. Copying from Grid to Other Editors



Pasting Content from Other Editors into Grid Mode

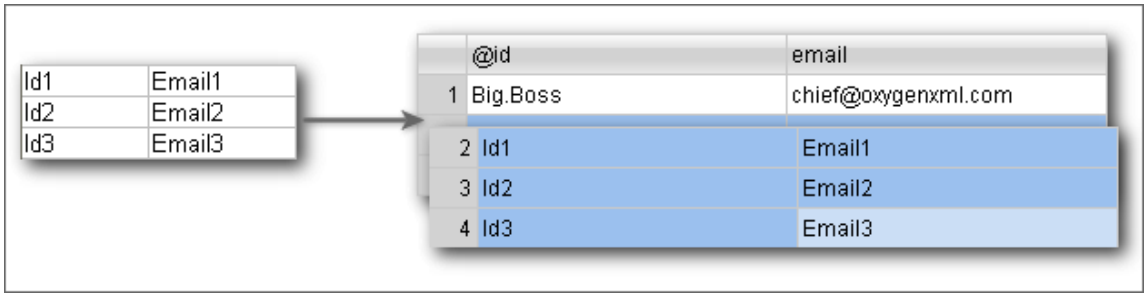
You can also paste well-formed XML content or tab-separated values from other editors into the **Grid** editor. If you paste XML content, the result will be the insertion of the nodes obtained by parsing this content.

Figure 122. 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.

Figure 123. Pasting Tab-Separated Values into Grid



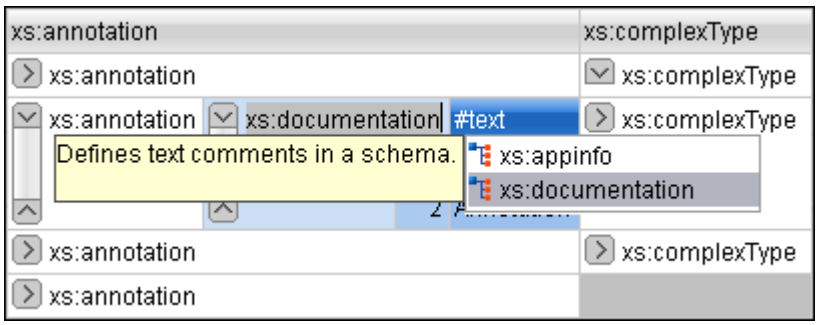
If you need to add copied content to your existing content (rather than overwriting existing cells), you need to first insert new cells by using the **Insert row** or **Insert column** actions from the contextual menu. This is useful, for example, when trying to transfer data from spreadsheet-like editors 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 2303)* 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).

Figure 124. Content Completion in Grid Editing Mode



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.

Figure 125. Default left to right text orientation

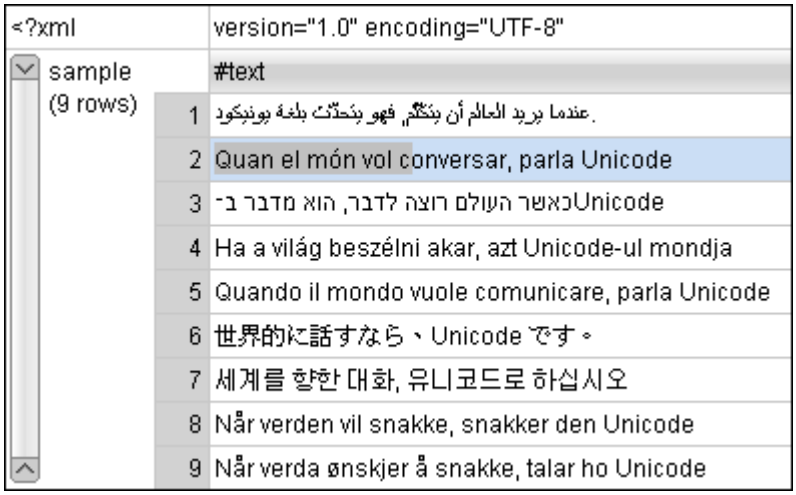
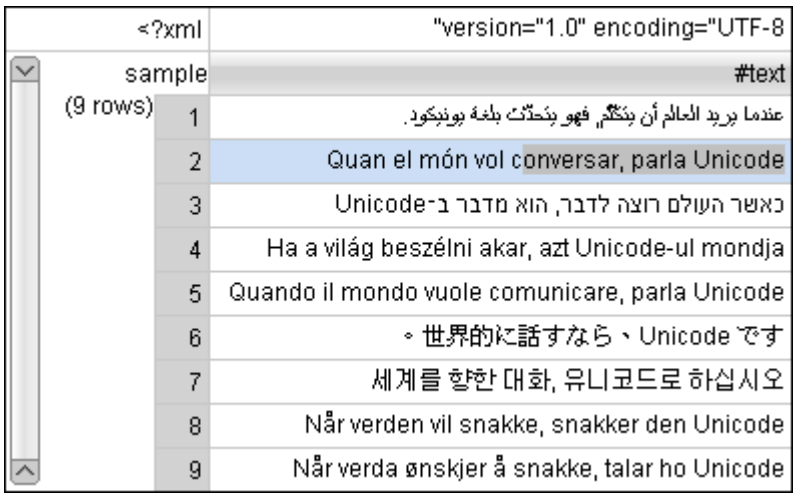


Figure 126. Right to left text orientation



Related Information:

Special Character Support in Text Mode *(on page 489)*
Inserting Special Characters with the Character Map *(on page 394)*

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 Developer and switch to **Grid** mode.
2. Expand the nodes *(on page 507)* 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 Developer 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 1817)*.

Resources

For more information about exchanging data between Oxygen XML Developer 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 1817)*
Pasting Content from Other Editors into Grid Mode *(on page 511)*

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 Developer 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 567)*

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 **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 **Check Well-Formedness** action from the **Validate** submenu when invoking the contextual menu in the **Project** view *(on page 334)*.

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

```
<root><tag></root>
```

When the **Check Well-Formedness** action is performed, the following error is displayed:

```
The element type "tag" must be terminated by the matching end-tag "</tag>"
```

To resolve the error, click the record in the resulting 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 </tag>.

Example: A non *Namespace-wellformed* Document

```
<prefix:elem></prefix:elem>
```

When the **Check Well-Formedness** action is performed, the following error is displayed:

```
The prefix "prefix" for element "prefix:elem" is not bound.
```

Example: A non *Namespace-valid* Document

```
<x:y></x:y>
```

When the **Check Well-Formedness** action is performed, the following error is displayed:

```
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 Developer, 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 554)* section.

Automatic Validation

By default, Oxygen XML Developer automatically checks for validation errors as you are editing a document. The **Enable automatic validation** option *(on page 160)* in the **Document Checking** preferences page *(on page 160)* 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 161)* 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 517)*
Presenting Validation Errors in Text Mode *(on page 519)*

Manual Validation Actions

You can choose to validate documents at any time by using the manual validation actions that are available in Oxygen XML Developer.

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

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 334)*.

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 2309)* 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 557)*.

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 1873)* developed independently and added to Oxygen XML Developer after installation.

Validate with Schema

Available from the **Validate** submenu when invoking contextual menu in the **Project** view *(on page 334)*.

This action opens a dialog box that allows you to specify a schema for validating all selected files *(on page 558)*.

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 171)* for the built-in validator in the Oxygen XML Developer preferences page.

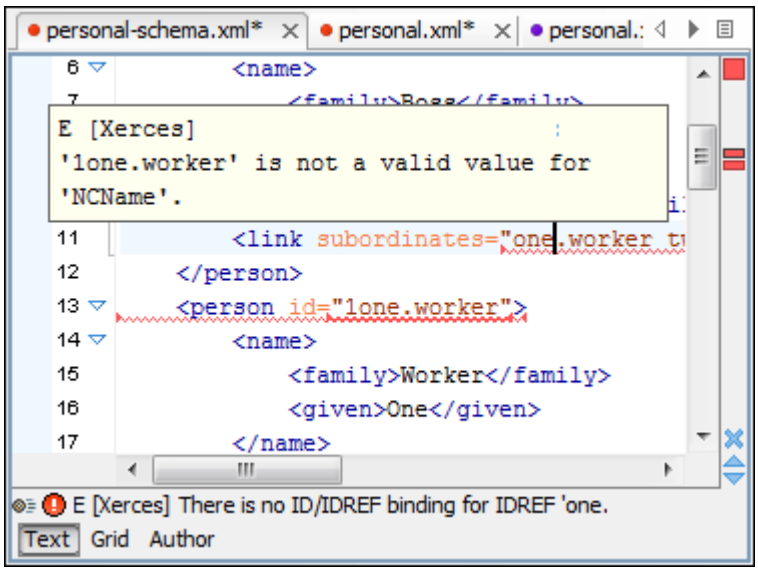
Tip:
If a large number of validation errors are detected and the validation process takes too long, you can limit the maximum number of reported errors in the **Document Checking** preferences page *(on page 160)*.

Related information
Automatic Validation *(on page 517)*
Presenting Validation Errors in Text Mode *(on page 519)*

Presenting Validation Errors in Text Mode

By default, Oxygen XML Developer automatically validates documents *(on page 517)* while editing in the **Text** mode, and actions are also available to manually validate documents *(on page 517)* on-request.

Figure 127. Presenting Validation Errors in Text Mode



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 468)*, 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 160)*.

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 79)*, go to **Editor > Document checking**, and specify the desired limit in the **Maximum number of validation highlights** option *(on page 160)*.

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 160)*.

Bottom Part of the Stripe

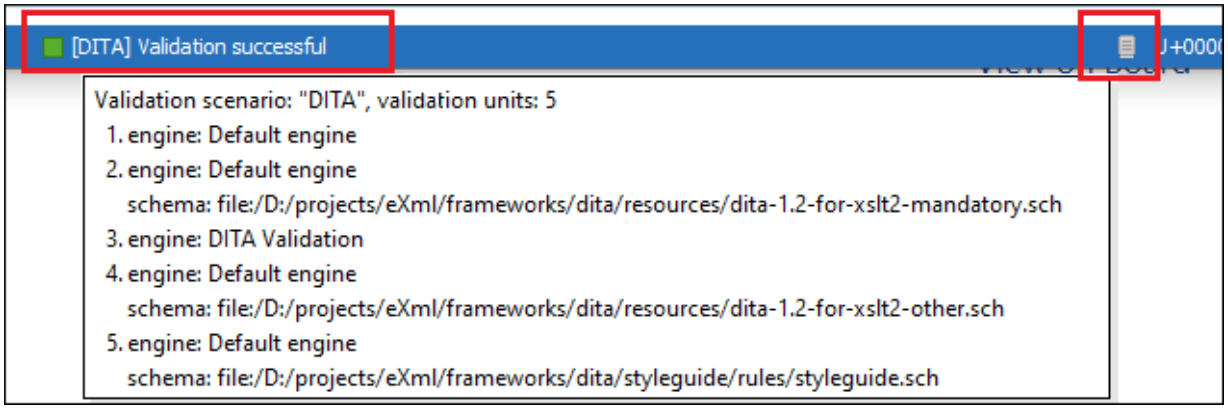
Two navigation arrows ( ) can be used to jump to the next or previous issue. The same actions can be triggered from **Document > Automatic validation > Next Error/Highlight** (**Ctrl + Period (Command + Period on macOS)**) and **Document > Automatic validation > Previous Error/ Highlight** (**Ctrl + 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 551)* (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 160)* 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 437)*, where even more details are displayed.



- If you want to see all the validation messages grouped in the **Results** view *(on page 474)* view, 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 516)*

Presenting Schematron Validation Issues *(on page 962)*

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.sf.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 161)*. After a custom validation engine is properly configured *(on page 161)*, 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 523)* of Oxygen XML Developer for linked messages:

- **Saxon-EE** - Included in Oxygen XML Developer. 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 172)*.
- **LIBXML** - Not included in Oxygen XML Developer and, depending on your operating system, the libraries need to be downloaded and installed separately from 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 2309)* (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 ${ds}` parameter manually to the DTD validation command line. `${ds}` represents the detected DTD declaration in the XML document.

  CAUTION:
  File paths containing spaces are not handled correctly in the LIBXML processor. For example, the built-in *XML Catalog (on page 2309)* files of the built-in document types (DocBook, TEI, DITA, etc.) are not handled by LIBXML if Oxygen XML Developer 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.

A custom validator cannot be applied on files loaded through an Oxygen XML Developer custom protocol plugin *(on page 1873)* developed independently and added to Oxygen XML Developer after installation.

Linked Output Messages of an External Engine

Validation engines display messages in an output view at the bottom of the Oxygen XML Developer 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.

  Note:
  The **Line**/**Column** pair works in conjunction with the **EndLine**/**EndColumn** pair. Thus, if both pairs are specified, then the highlight starts at **Line**/**Column** and ends at **EndLine**/**EndColumn**. If the **EndLine**/**EndColumn** pair is missing, the highlight starts from the beginning of the line identified by the **Line** parameter and ends at the column identified by the **Column** parameter.

- **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 Developer, 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:

```
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 Developer 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 99)*.
- In a validation scenario *(on page 526)*, 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 1104)* 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 1857)*.

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 Developer 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 12 or Saxon 6.5 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 Developer (Saxon 12) 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 2306)* 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 350)* or Modular Contextual XML Editing Using 'Main Files' Support *(on page 567)*.

All validation scenarios that are defined in the *framework* associated with the current document are shown in the **Validation** toolbar drop-down menu. This makes it easy for the user to turn validation scenarios on or off for a document.

> **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 516)*

Presenting Validation Errors in Text Mode *(on page 519)*

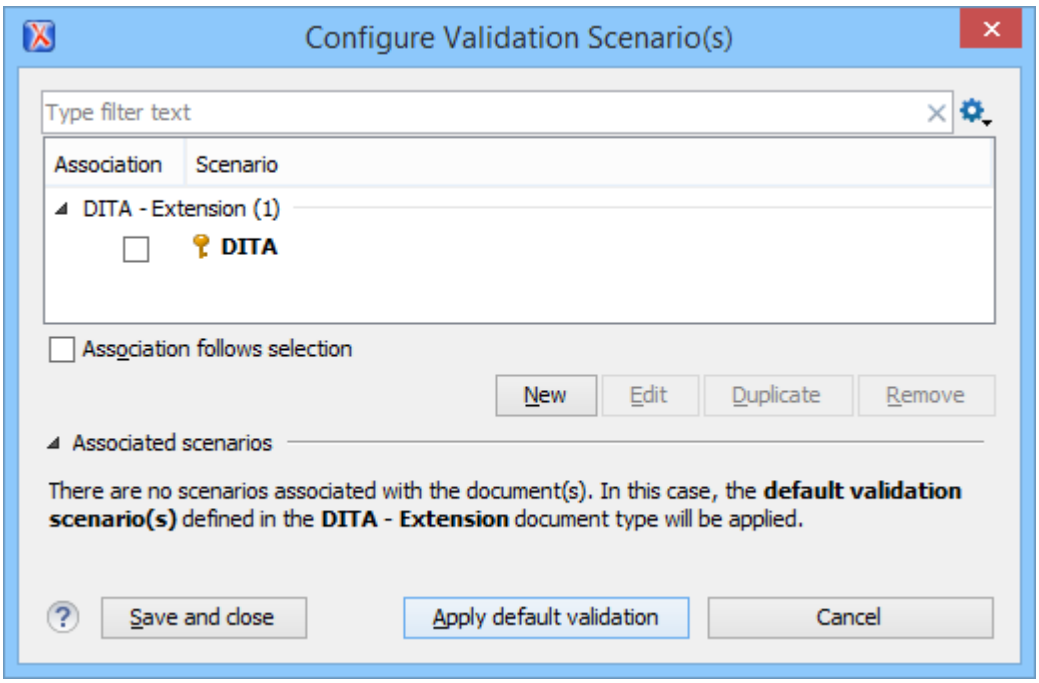
Creating a New Validation Scenario

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 334)*).
   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 2305)* 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 350)* or Modular Contextual XML Editing Using 'Main Files' Support *(on page 567)*.

Figure 128. 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 Developer 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 Developer 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.

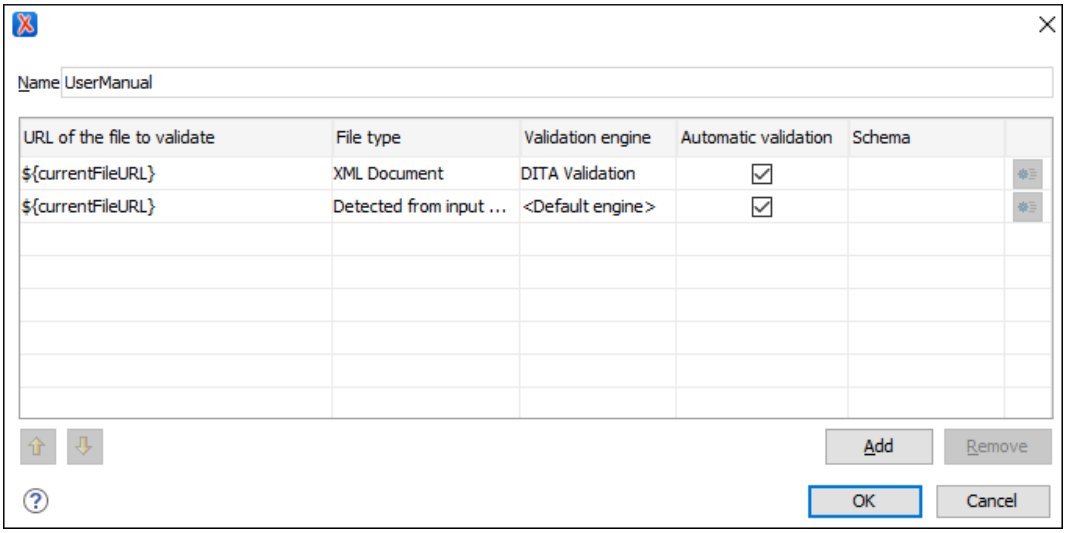
> **Note:**
> When you keep both scenarios, Oxygen XML Developer adds `imported` to the name of the imported scenario.

Export selected scenarios

Use this option to export selected scenarios individually. Oxygen XML Developer 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 129. 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

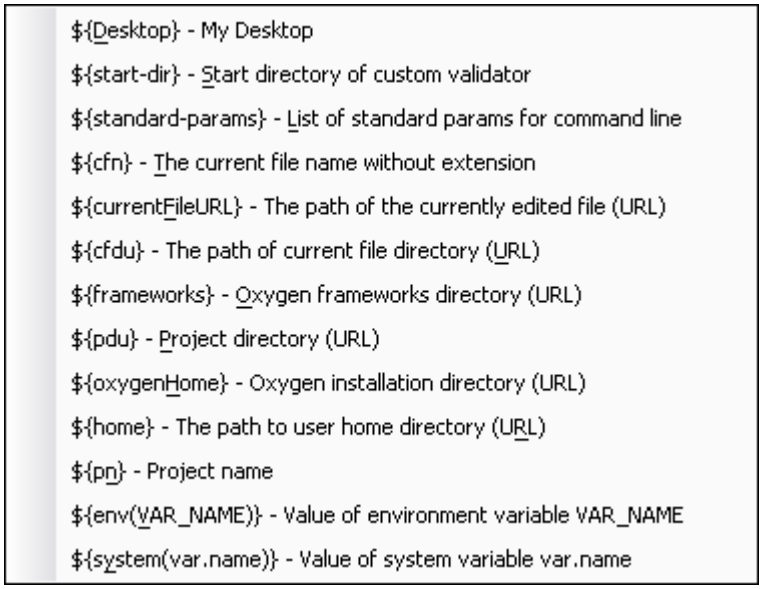
You can choose between storing the scenario in the **Project Options** *(on page 2308)* or **Global Options** *(on page 2305)*.

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 252)* or a custom editor variable *(on page 261)*.

Figure 130. Insert an Editor Variable



File type

The type of the document that is validated in the current validation unit. Oxygen XML Developer 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 Developer 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 180)*, XQuery preferences page *(on page 185)*, XML Schema preferences page *(on page 172)*).
- **DITA Validation** engine - Performs DITA-specific checks in the context of the specifications.
- **DITA Map Validation and Completeness Check** engine - Performs a validation process that checks the DITA map document and all referenced topics and maps.
- **DITA-OT Project Validation and Completeness Check** engine - Performs a validation process that checks each context from the provided DITA-OT project file.
- **Table Layout Validation** engine - Looks for table layout problems.

Automatic validation

If this option is selected, the validation operation defined by this row is also applied by the automatic validation feature *(on page 517)*. If the **Automatic validation** feature is disabled in the Document Checking preferences page *(on page 160)*, 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.

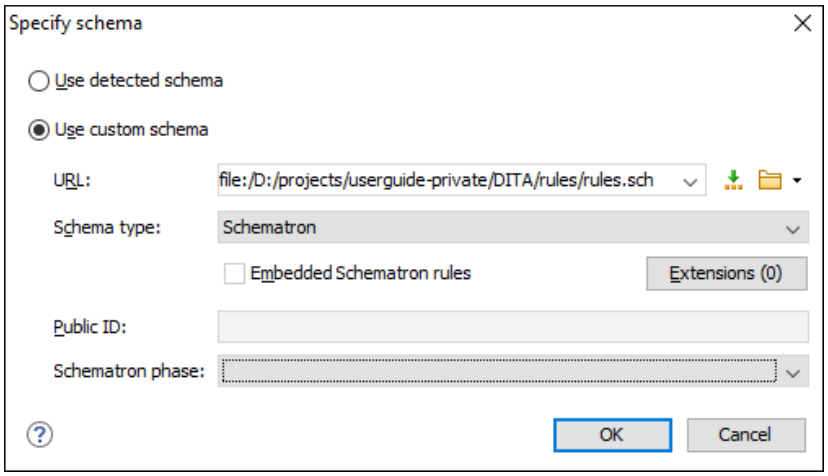
Settings

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 131. 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 554)*.

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) or SFTP (or a custom protocol *(on page 1890)*). You can specify the URL by using the text field, the history drop-down, the **Insert Editor Variables** *(on page 252)* 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 2306)* 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.

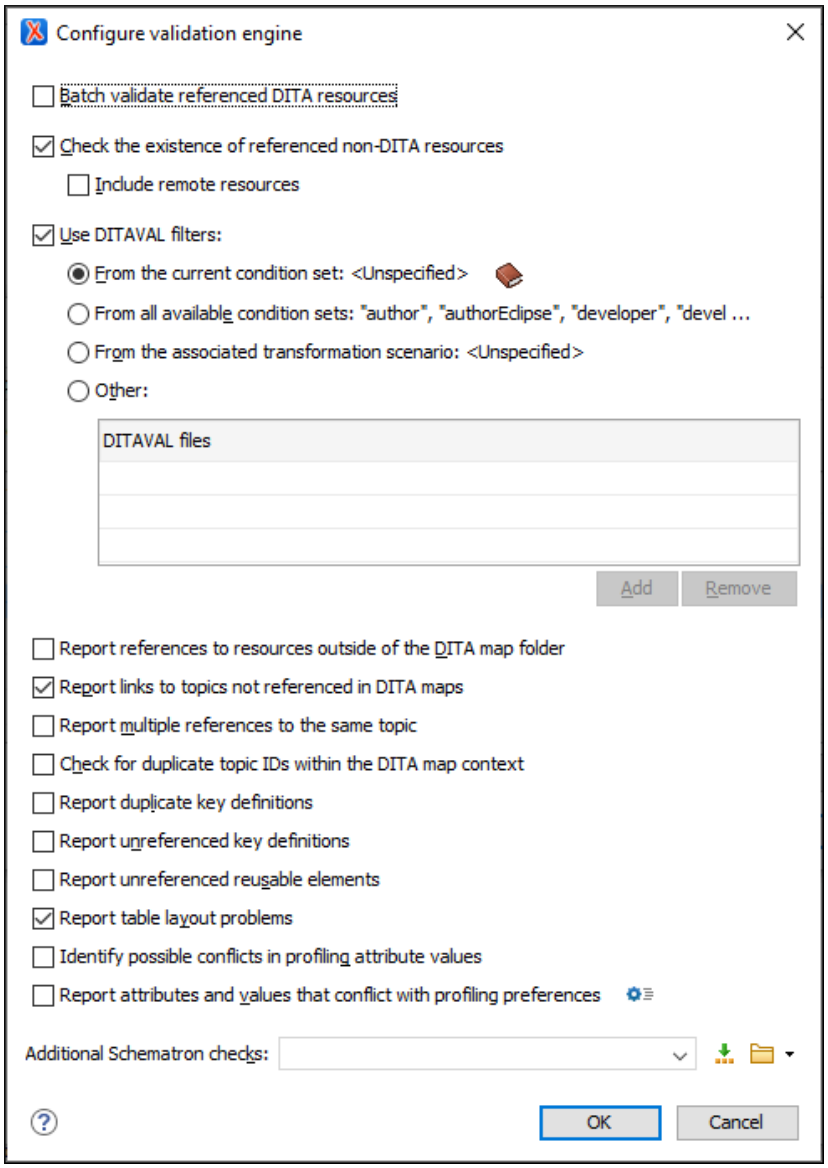
◦ **Configure Validation Engine Dialog Box**

This dialog box allows you to configure options for checking the DITA map document and all referenced topics and maps.

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 132. 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 525)*.

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. Clicking the **Details** icon opens a topic in the Oxygen XML Developer 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 252)* 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 2309)* 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*. Also reports related links defined in relationship tables whose target topics are not referenced 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.

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. 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 Developer 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 252)* button, or the browsing actions in the **Browse** drop-down list.

> **Advanced Tip:**
> Some APIs are available that retrieve information about DITA keys that are referenced within a topic. The APIs can be called from XSLT Stylesheets (including XML Refactoring operations) or Schematron schemas. For details, see API Documentation: DITAXSLTExtensionFunctionUtil.

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.

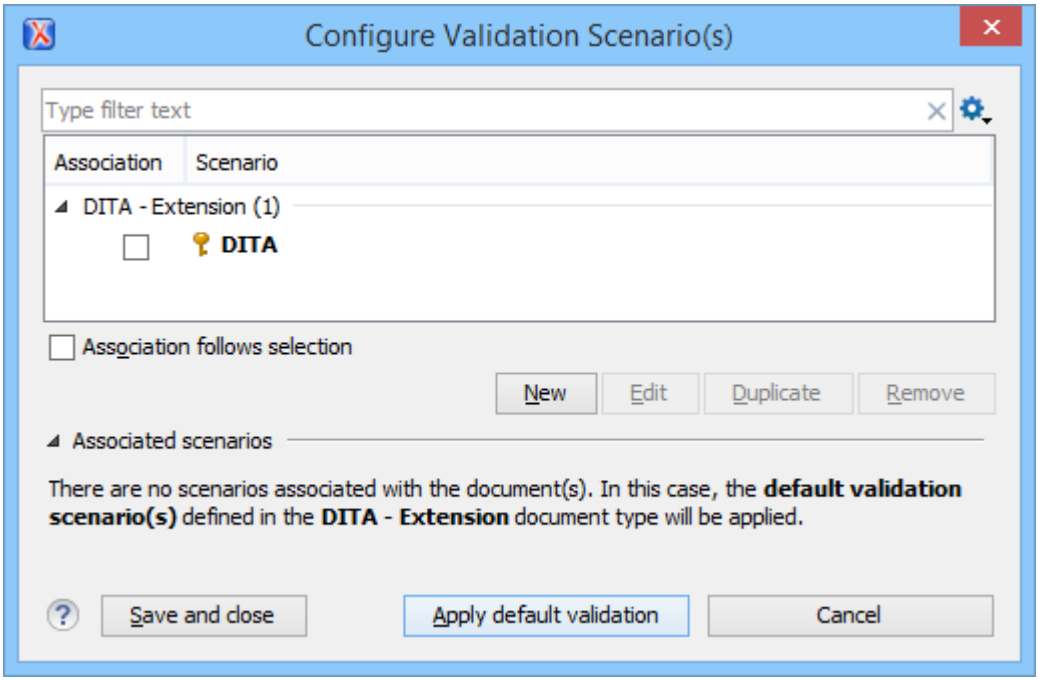
Editing a Validation Scenario

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 334)*).
   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 2305)* 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 350)* or Modular Contextual XML Editing Using 'Main Files' Support *(on page 567)*.

Figure 133. 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 Developer 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 Developer 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.

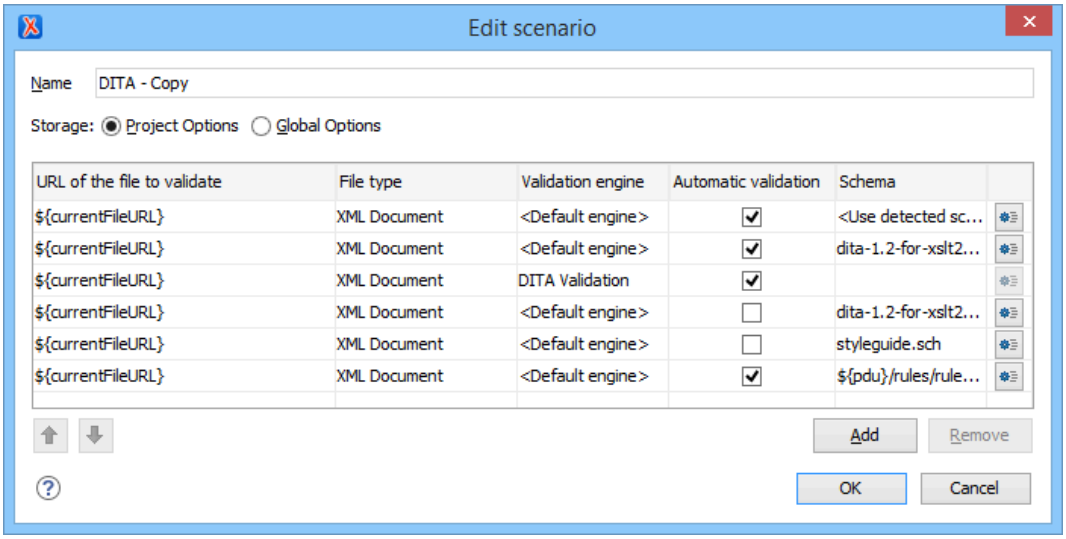
> **Note:**
> When you keep both scenarios, Oxygen XML Developer adds `imported` to the name of the imported scenario.

Export selected scenarios

Use this option to export selected scenarios individually. Oxygen XML Developer 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 Developer 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 134. 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

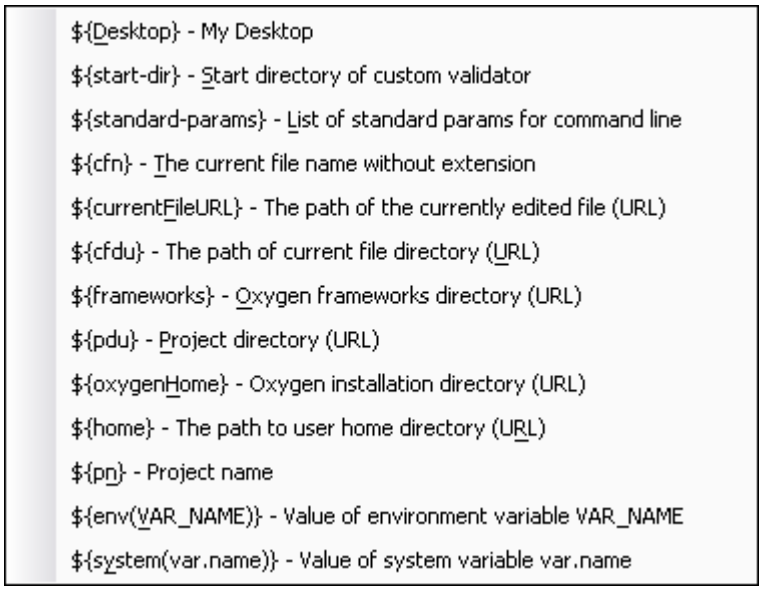
You can choose between storing the scenario in the **Project Options** *(on page 2308)* or **Global Options** *(on page 2305)*.

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 252)* or a custom editor variable *(on page 261)*.

Figure 135. Insert an Editor Variable



File type

The type of the document that is validated in the current validation unit. Oxygen XML Developer 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 Developer 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 180)*, XQuery preferences page *(on page 185)*, XML Schema preferences page *(on page 172)*).
- **DITA Validation** engine - Performs DITA-specific checks in the context of the specifications.
- **DITA Map Validation and Completeness Check** engine - Performs a validation process that checks the DITA map document and all referenced topics and maps.
- **DITA-OT Project Validation and Completeness Check** engine - Performs a validation process that checks each context from the provided DITA-OT project file.
- **Table Layout Validation** engine - Looks for table layout problems.

Automatic validation

If this option is selected, the validation operation defined by this row is also applied by the automatic validation feature *(on page 517)*. If the **Automatic validation** feature is disabled in the Document Checking preferences page *(on page 160)*, 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.

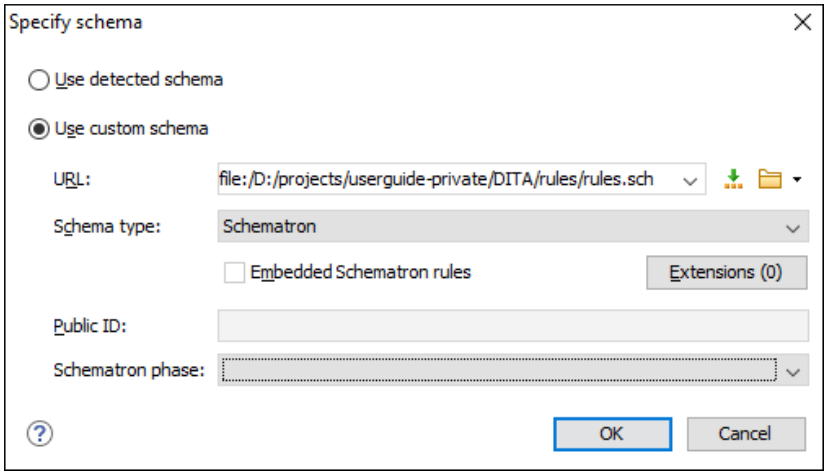
Settings

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 136. 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 554)*.

  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) or SFTP (or a custom protocol *(on page 1890)*). You can specify the URL by using the text field, the history drop-down, the **Insert Editor**

Variables *(on page 252)* 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 2306)* 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.

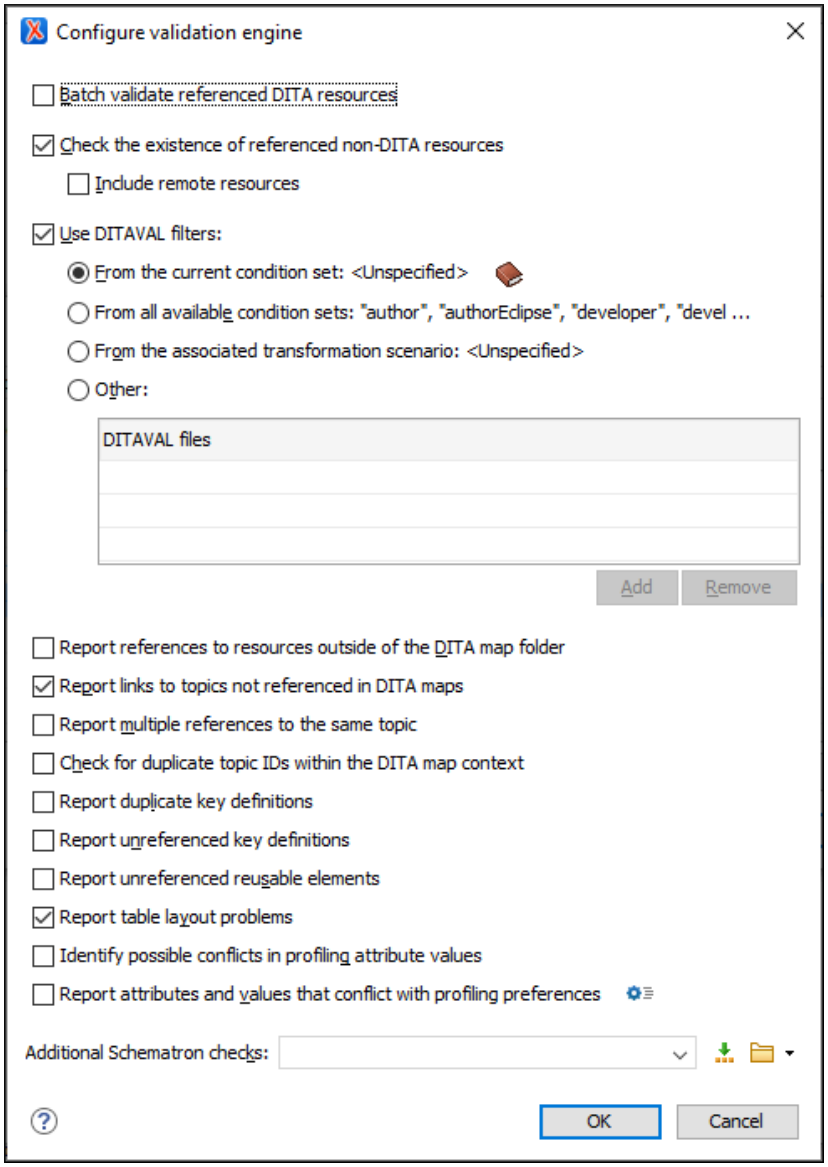
◦ **Configure Validation Engine Dialog Box**

This dialog box allows you to configure options for checking the DITA map document and all referenced topics and maps.



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 137. 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 525)*.

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. Clicking the **Details** icon opens a topic in the Oxygen XML Developer 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 252)* 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 2309)* 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*. Also reports related links defined in relationship tables whose target topics are not referenced 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.

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. 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 Developer 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 252)* button, or the browsing actions in the **Browse** drop-down list.

> **Advanced Tip:**
> Some APIs are available that retrieve information about DITA keys that are referenced within a topic. The APIs can be called from XSLT Stylesheets (including XML Refactoring operations) or Schematron schemas. For details, see API Documentation: DITAXSLTExtensionFunctionUtil.

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

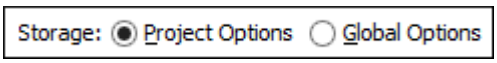
Removes an existing validation unit from the list.

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 2308)* or by exporting them to a specialized scenarios file *(on page 251)* 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 2308)* or **Global Options** *(on page 2305)*.



Selecting **Project Options** *(on page 2308)* 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 2305)* 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.

Related Information:
Sharing Application Settings *(on page 241)*

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 2309)*.

For example, if the XML document contains a reference to a remote schema `docbook.rng` like this:

```
<?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:

```
<uri name="http://www.oasis-open.org/docbook/xml/5.0/rng/docbook.rng"
    uri="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:

```
<topic xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:noNamespaceSchemaLocation="urn:oasis:names:tc:dita:xsd:topic.xsd:1.1">
```

the URN can be resolved to a local schema file with a catalog entry like this:

```
<uri name="urn:oasis:names:tc:dita:xsd:topic.xsd:1.1"
    uri="topic.xsd"/>
```

Related Information:

Working with XML Catalogs *(on page 564)*

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 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>
        <listitem>
          <link>a link here</link>
        </listitem>
      </itemizedlist>
    </sect1>
</article>
```

The  **Validate Document** action will return the following error:

```
Unexpected element "link". The content of the parent element type must match
"(calloutlist|glosslist|bibliolist|itemizedlist|orderedlist|segmentedlist|simplelist
|variablelist|caution|important|note|tip|warning|literallayout|programlisting
|programlistingco|screen|screenco|screenshot|synopsis|cmdsynopsis|funcsynopsis
```

```
|classsynopsis|fieldsynopsis|constructorsynopsis|destructorsynopsis|methodsynopsis
|formalpara|para|simpara|address|blockquote|graphic|graphicco|mediaobject|mediaobjectco
|informalequation|informalexample|informalfigure|informaltable|equation|example|figure
|table|msgset|procedure|sidebar|qandaset|task|anchor|bridgehead|remark|highlights
|abstract|authorblurb|epigraph|indexterm|beginpage)+".
```

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 Developer 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-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 Developer 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

```
<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

```
<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>
    <start>
     ..............
    </start>
</grammar>
```

Related Information:
Embedding Schematron Quick Fixes in Relax NG or XML Schema *(on page 1005)*

Ignoring/Unignoring Validation Problems

If the **Enable support for ignoring validation problems** option *(on page 163)* is selected in the **Ignored Validation Problems** preferences page *(on page 163)*, 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 474)*, 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.

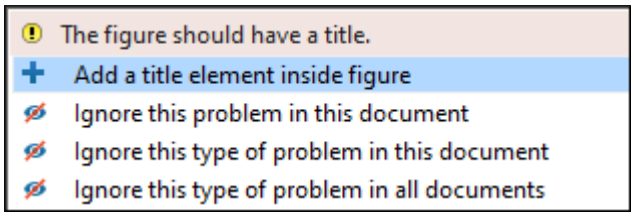
Ignore this type of problem in this document

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 138. Schematron Ignored Problems Quick Fix Proposals



When a validation problem is ignored:

- It is added to the *Ignored Problems Table (on page 163)* in the **Ignored Validation Problems** preferences page *(on page 163)*. 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 474)*, 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 163)*) 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 474)*.

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 Developer *Quick Fix* support *(on page 2308)* 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

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:

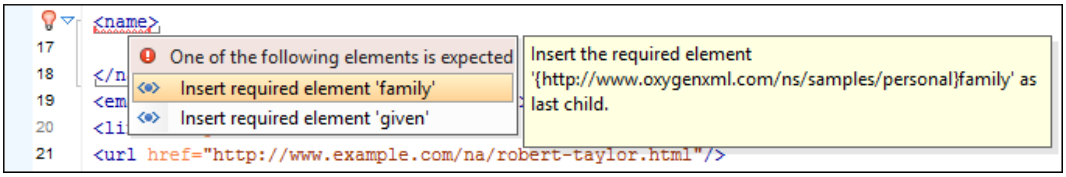
- When hovering over the error or warning, the proposals may be presented in a tooltip pop-up window and the available quick *Quick Fixes* include a link that can be used to perform the fix.

Figure 139. Quick Fix Presented in a Tooltip in Text Mode



- 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 Developer displays the list of available fixes.

Figure 140. Quick Fix Menu Invoked by Clicking on the Icon



- 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 + Option + 1 on macOS)** on your keyboard.

Tip:
If the **Oxygen AI Positron** add-on (version 7.0 or later) is installed, an **AI Positron Fix** action is also available in the **Quick Fix** menu and in the tooltip when hovering over a validation issue. You can use this action to fix the problem with assistance from the **AI Positron** platform. It will propose content in the chat pane within the **AI Positron** view that can be used to solve the problem.

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 144)* are configured.

Quick Fixes for DTD, XSD, and Relax NG Errors

Oxygen XML Developer offers *Quick Fixes (on page 2308)* 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 .

Oxygen XML Developer provides *Quick Fixes* for numerous types of problems, including the following:

Problem Type	Available Quick Fixes
A specific element is required in the current context	Insert the required element
An element is invalid in the current context	Remove the invalid element
The content of the element should be empty	Remove the element content
An element is not allowed to have child elements	Remove all child elements
Text is not allowed in the current element	Remove the text content
A required attribute is missing	Insert the required attribute
An attribute is not allowed to be set for the current element	Remove the attribute
The attribute value is invalid	Propose the correct attribute values

Problem Type	Available Quick Fixes
ID value is already defined	Generate a unique ID value
References to an invalid ID	Change the reference to an already defined ID

Related Information:

Schematron Quick Fixes (SQF) *(on page 553)*
Ignoring/Unignoring Validation Problems *(on page 549)*

Schematron Quick Fixes (SQF)

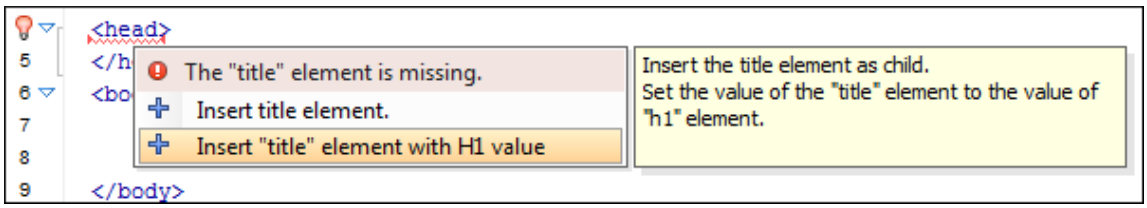
Oxygen XML Developer provides support for Schematron *Quick Fixes (on page 2308)* (SQF). They help you resolve issues that appear in XML documents that are validated against Schematron schemas by offering you solution proposals. The Schematron *Quick Fixes* are an extension of the Schematron language 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.

Figure 141. Example of a Schematron Quick Fix



Related Information:

Editing Schematron Quick Fixes *(on page 978)*
Schematron Quick Fix Specifications
Presenting Schematron Validation Issues *(on page 962)*
Examples of Schematron Rules and Quick Fixes *(on page 950)*
Ignoring/Unignoring Validation Problems *(on page 549)*

Associating a Schema to XML Documents

To provide as-you-type validation and to compute valid proposals for the *Content Completion Assistant (on page 2303)*, Oxygen XML Developer 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 Developer 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 555)* 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 559)*.
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 561)*.

   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 563)*.

Detecting a Schema for Content Completion

For content completion, Oxygen XML Developer 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 555)*.
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 559)*.
3. Oxygen XML Developer determines the most appropriate or highest ranking schema that is associated directly in the XML document *(on page 561)* 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 563)*.

Related Information:
Modular Contextual XML Editing Using 'Main Files' Support *(on page 567)*
W3C: Associating Schemas with XML Documents

Associating a Schema Through a Validation Scenario

Oxygen XML Developer 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 525)* and there are several methods that can be used to do so.

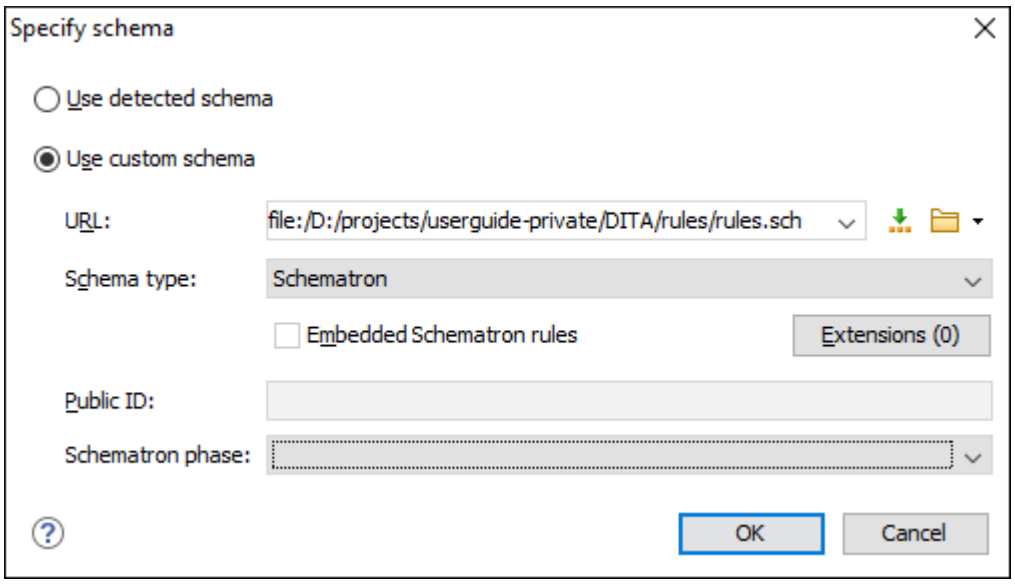
Configure a Validation Scenario and Specify the Schema

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 334)*).
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 538)* according to your needs and click the **Specify Schema** button.

   Step Result: The **Specify Schema** dialog box is displayed:

Figure 142. 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 554)*.

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) or SFTPS (or a custom protocol *(on page 1890)*). You can specify the URL by using the text field, the history drop-down, the **Insert Editor Variables** *(on page 252)* 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 2306)* 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 143. 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) or SFTPS (or a custom protocol *(on page 1890)*). You can specify the URL by using the text field, the history drop-down, the **Insert Editor Variables** *(on page 252)* 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 2306)* 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:

   Figure 144. 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) or SFTPS (or a custom protocol *(on page 1890)*). You can specify the URL by using the text field, the history drop-down, the **Insert Editor Variables** *(on page 252)* 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 2306)* 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 Developer uses rules defined in the schema that is detected in the validation scenarios that are associated to each particular document type.

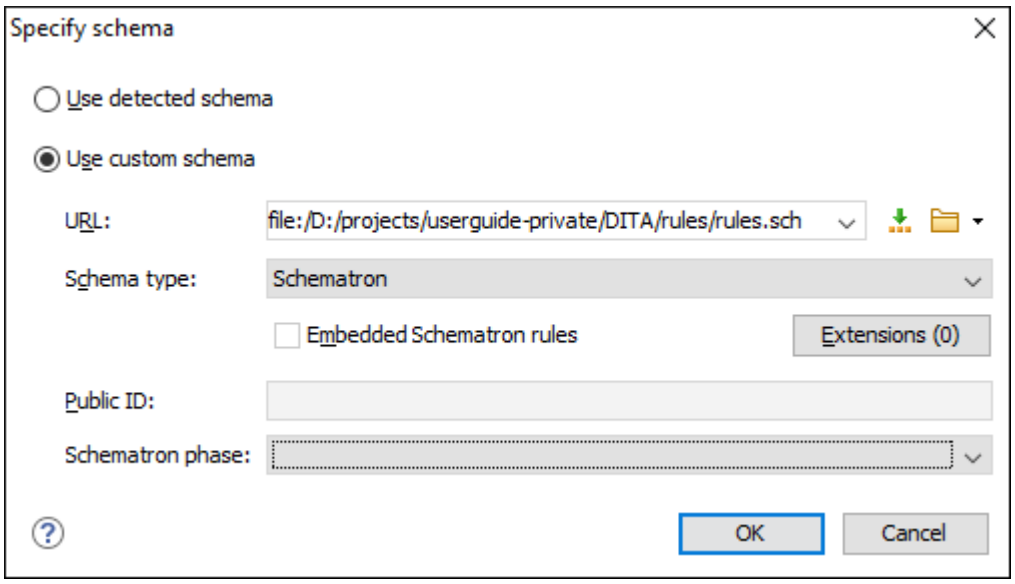
To associate a schema in validation scenarios defined in the *framework (on page 2305)* (document type) configuration, follow these steps:

1. Open the **Preferences** dialog box **(Options > Preferences)** *(on page 79)* 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 95)*.
3. Go to the **Validation** tab *(on page 121)*.
4. Create or edit a validation scenario:
   a. To create a new validation scenario *(on page 526)*, click the **New** button.
   b. To edit the properties of an existing validation scenario *(on page 536)*, 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 538)* according to your needs and click the **Specify Schema** button.

   Step Result: The **Specify Schema** dialog box is displayed:

Figure 145. 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 554)*.

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) or SFTPS (or a custom protocol *(on page 1890)*). You can specify the URL by using the text field, the history drop-down, the **Insert Editor Variables** *(on page 252)* 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 2306)* 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 2303)* 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.

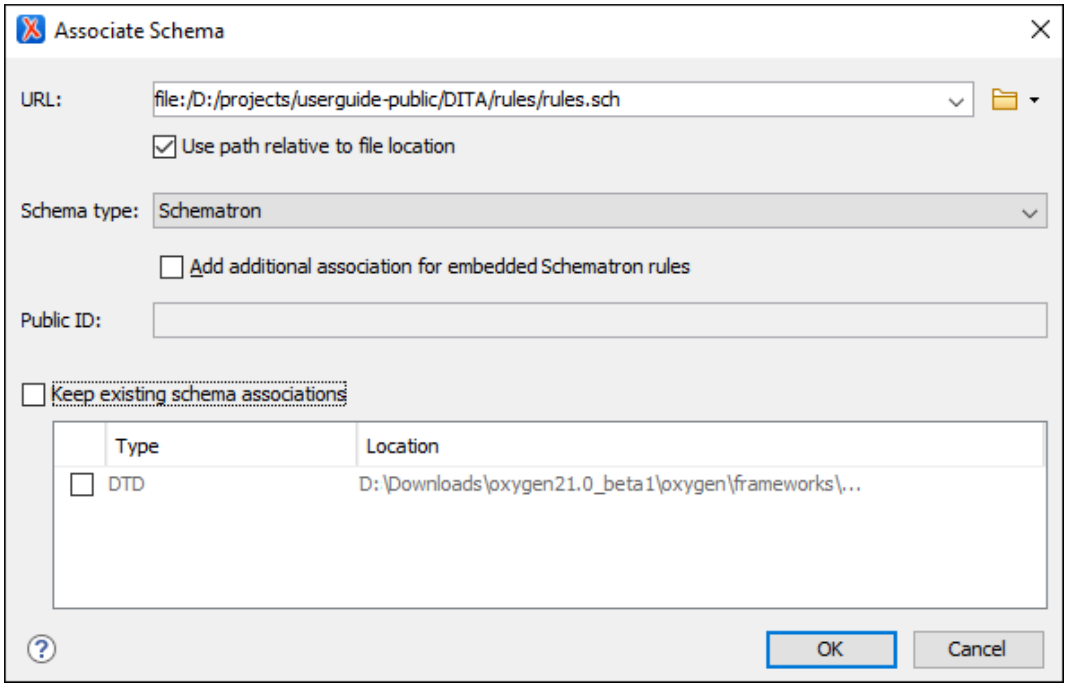
The association can specify a relative file path or a URL of the schema. The advantage of relative file path is that you can configure the schema at file level instead of *framework (on page 2305)* level.

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 146. 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) or SFTP (or a custom protocol *(on page 1890)*).
- **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 514)*
Content Completion Assistant in Text Mode *(on page 457)*

Associating a Schema in a Framework (Document Type) Configuration

The schema used to compute valid proposals in the *Content Completion Assistant (on page 2303)* and by the document validation engine to report errors and warnings can be defined in each particular *framework (on page 2305)* (document type). This schema will be used only if one is not detected in the current XML file *(on page 561)*.

To associate a schema in a particular *framework* (document type), follow these steps:

1. Open the **Preferences** dialog box **(Options > Preferences)** *(on page 79)* and go to **Document Type Association**.
2. Select your particular document type and click the **Edit** *(on page 93)*, **Extend** *(on page 94)*, or **Duplicate** *(on page 94)* 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 95)*.
3. Go to the **Schema** tab *(on page 99)*.
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 Developer can learn the document structure by parsing the document internally to provide an initialization source for content completion *(on page 457)* and document validation *(on page 514)*.

This feature is controlled by the **Learn on open document** option *(on page 144)* that is available in the **Editor > Content Completion** preferences page. This feature enabled by default. If you want to disable it, deselect the **Learn on open document** option.

You can choose to create a DTD from the learned structure and save it as a file using the **Learn Structure** and **Save Structure** actions.

Creating a DTD from Learned Document Structure

To create a DTD from the learned structure of an XML document, 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. A **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 file path for the DTD.
4. Click the **Save** button.

Related Information:
Detecting a Schema *(on page 554)*

Working with XML Catalogs

Oxygen XML Developer uses *XML Catalogs (on page 2309)* 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 Developer 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 Developer uses different catalog mappings.

Table 6. Catalog Mappings

Doc Type	Referenced Resource	Mappings
XML	DTD	*system* or *public* The **Prefer** option *(on page 169)* controls which one of the mappings should be used.
	XML Schema Relax NG Schematron NVDL	The following strategy is used (if one step fails to provide a resource, the next is applied): 1. Resolve the schema using *URI* catalog mappings. 2. Resolve the schema using *system* catalog mappings. This happens only if the **Resolve schema locations also through system mappings** option *(on page 169)* is selected (it is by default). 3. Resolve the root *namespace* using *URI* catalog mappings.
XSL	XSL/ANY	*URI*
CSS	CSS	*URI*
JSON	JSON	*URI*
XPROC	XPROC	*URI*

Table 6. Catalog Mappings (continued)

Doc Type	Referenced Resource	Mappings
XML Schema	XML Schema	The following strategy is used (if one step fails to provide a resource, the next is applied): 1. Resolve schema reference using *URI* catalog mappings. 2. Resolve schema reference using *system* catalog mappings. This happens only if the **Resolve schema locations also through system mappings** option *(on page 169)* is selected (it is by default). 3. Resolve schema *namespace* using *URI* catalog mappings. This happens only if the **Process namespaces through URI mappings for XML Schema** option *(on page 169)* is selected (it is not by default).
Relax NG	Relax NG	

Creating an XML Catalog with a Template

An *XML Catalog (on page 2309)* file can be easily created in Oxygen XML Developer starting from the document template called *OASIS XML Catalog*. It is available when creating new document templates *(on page 299)*.

How Oxygen XML Developer Determines which Catalog to Use

Oxygen XML Developer uses *XML Catalogs (on page 2309)* 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 2305)* (DocBook, DITA, TEI, XHTML, SVG, etc.)

Oxygen XML Developer includes default global catalogs and default catalogs for each of the built-in *frameworks*, and you can also create your own.

Oxygen XML Developer looks for catalogs in the following order:

- Global user-defined catalogs that are set in the **XML Catalog** preferences page *(on page 168)*.
- User-defined catalogs that are set for each *framework (on page 2305)* in the **Catalog** tab *(on page 119)* of the **Document Type** configuration dialog box *(on page 95)*.
- 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:

```
<topic xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
     xsi:noNamespaceSchemaLocation="urn:oasis:names:tc:dita:xsd:topic.xsd:1.1">
```

The URN can be resolved to a local schema file with an XML catalog entry like this:

```
<uri name="urn:oasis:names:tc:dita:xsd:topic.xsd:1.1"
    uri="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:

```
<xs:include schemaLocation="someFolder/common.xsd"/>
```

The reference can be resolved to a local schema file with an XML catalog entry like this:

```
<uriSuffix uriSuffix="someFolder/common.xsd" uri="relative/path/to/common.xsd"/>
```

Related information

XML Catalog Preferences *(on page 168)*

Resolving Schema Locations Through XML Catalogs

Schema locations can be mapped using an *XML Catalog (on page 2309)*. Oxygen XML Developer 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 169)* is selected in the **XML Catalog** preferences page, it attempts to resolve the schema location as a system ID (`system`, `systemSuffix`, `rewriteSuffix`, `rerwriteSystem` 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 169)* 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 554)* is used.

Related Information:

Working with XML Catalogs *(on page 564)*

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 Developer 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 2306)*.

You cat set a main XML document either using the *main files* support from the **Project** view *(on page 350)*, 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 Developer 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 2306)* include:

- Correct validation of a module in the context of a larger XML structure.
- *Content Completion Assistant (on page 2303)* displays all collected entities and IDs starting from the *main files*.
- Oxygen XML Developer 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 569)* for ID/IDREFS values and for updating references when renaming/moving a resource. Oxygen XML Developer performs the search and refactoring actions in the context that the *main files* determine, improving the speed of execution.

Resources

For more information about editing modular XML files in the *main files* context, watch our video demonstration:

https://www.youtube.com/embed/e2oo4RWNxW8

Related information

Contextual Project Operations Using 'Main Files' Support *(on page 350)*

XML Referenced/Dependent Resources View *(on page 570)*

Search and Refactoring Actions for IDs and IDREFS

Oxygen XML Developer allows you to search for ID declarations and references (IDREFS) and to define the scope of the search and refactor operations *(on page 569)*. 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 491)* 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 569)*. 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 569)* 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 569)*.

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 569)* dialog box, this scope will be used instead.

> **Note:**
> Available in the **Text** mode only.

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 569)*.

> **Note:**
> Available in the **Text** mode only.

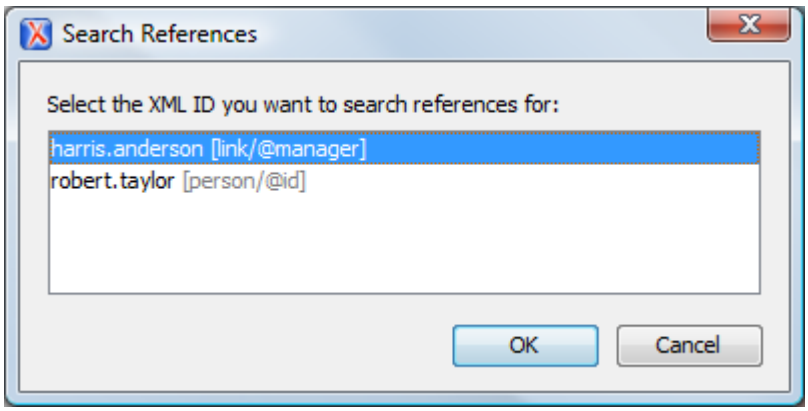
Search Occurrences in file

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 Developer 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 147. Selecting an ID in the Author Mode

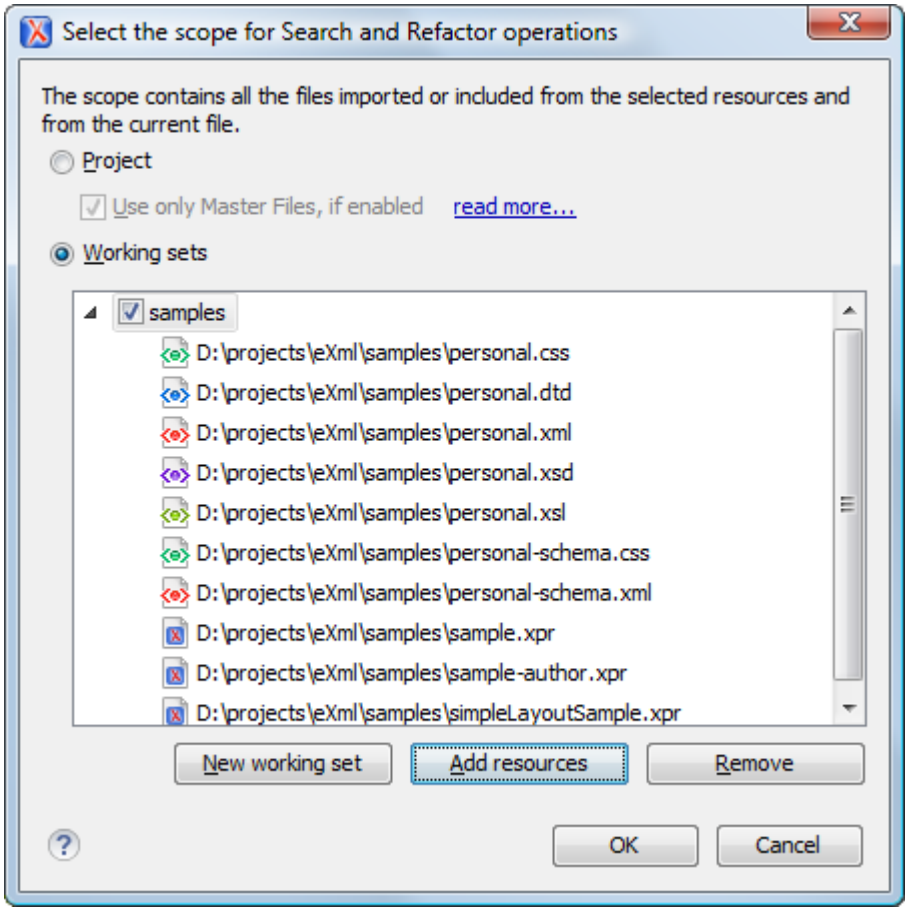


Related Information:
Modular Contextual XML Editing Using 'Main Files' Support *(on page 567)*

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 **Referenced/ Dependent Resources** view's toolbar. You can restrict the scope to the current project or to one or multiple *working sets (on page 2309)*. 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 350)*.

Figure 148. Change Scope Dialog Box



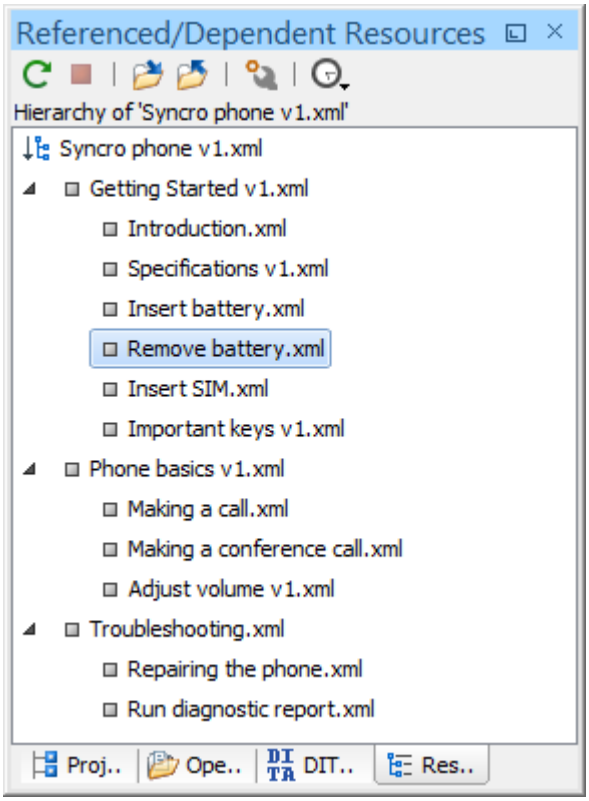
The scope you define is applied to all future search and refactor operations until you modify it. Contextual menu actions allow you to add or delete files, folders, and other resources to the *working set (on page 2309)* structure.

XML Referenced/Dependent Resources View

The **Referenced/Dependent Resources** 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 334)* and choose **Show referenced resources** or **Show dependent resources** from the contextual menu.

Figure 149. Referenced/Dependent Resources View



The following actions are available on the toolbar of the **Referenced/Dependent Resources** view:

Refresh

Refreshes the resource structure.

Stop

Stops the computing.

Show hierarchy for

Computes the hierarchical structure of the references for a resource.

Show dependencies for

Computes the structure of the dependencies for a resource.

Configure dependencies search scope

Allows you to configure a scope to compute the dependencies. 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 **Referenced/Dependent Resources** view contains the following actions:

Open

Opens the resource. You can also double-click a resource within the hierarchical 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.

Show references resources

Shows the references for the selected resource.

Show dependent resources

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 hierarchical structure.

Collapse All

Collapses all children of the selected resource from the hierarchical structure.



Tip:
When a recursive reference is encountered in the 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 573)*. 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 567)*
Search and Refactor Operations Scope *(on page 569)*

Moving/Renaming XML Resources

When you select the **Rename** action in the contextual menu of the **Referenced/Dependent Resources** 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 **Referenced/Dependent Resources** 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 2306)*. 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 575)* for assembling the parts together with the *main file*.

The main document looks like this:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE book SYSTEM "../xml/docbookx.dtd" [
```

```
<!ENTITY testing SYSTEM "testing.xml" > ]

>

<book>

<chapter> ...
```

The referenced document (*testing.xml*) looks like this:

```
<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:

```
... &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 Developer

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 transformationCreate a minimal XSLT stylesheet that simply copies the XML content, then create a transformation scenario that applies the stylesheet over the XML. The XSLT stylesheet would look like this:

```
<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 2306)* 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 573)* 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 Developer

The XInclude support in Oxygen XML Developer is enabled by default. It is controlled by the **Enable XInclude processing** option *(on page 171)* in the **XML > XML Parser** preferences page *(on page 171)*. When enabled, Oxygen XML Developer 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 version="1.0"?>
<!DOCTYPE chapter PUBLIC "-//OASIS//DTD DocBook XML V4.3//EN"
"http://www.oasis-open.org/docbook/xml/4.3/docbookx.dtd">
<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 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/dtd/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>
```

```
        <para>
          <emphasis>FIXME: MISSING XINCLUDE CONTENT</emphasis>
        </para>
      </xi:fallback>
    </xi:include>
</article>
```

In this example, note the following:

- The DOCTYPE declaration defines an entity that references a file containing the information to add the *xi* namespace to certain elements defined by the DocBook DTD.
- The href attribute of the *xi:include* element specifies that the `introduction.xml` file will replace the *xi:include* element when the document is parsed.
- If the `introduction.xml` file cannot be found, the parser will use the value of the *xi:fallback* element - a *FIXME* message.

Example: Using XInclude to Combine Fragments of Files

If you want to include only a fragment of a file in the *main file (on page 2306)*, the fragment must be contained in a tag having an `@xml:id` attribute and you must use an XPointer expression pointing to the `@xml:id` value.



Notice:
Oxygen XML Developer supports the XPointer *Framework* and the XPointer element() Scheme, but it does NOT support the XPointer xpointer() Scheme.

For example, if the *main file* is:

```
<?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 (`a.xml`) is:

```
<?xml version="1.0" encoding="UTF-8"?>
<test>
    <a xml:id="a1">test</a>
</test>
```

after resolving the XPointer reference, the document is:

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-model href="test.rng" type="application/xml"
          schematypens="http://relaxng.org/ns/structure/1.0"?>
<test>
    <a xml:id="a1" xml:base="a.xml">test</a>
</test>
```

Viewing the Expanded Content in Oxygen XML Developer

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 transformationCreate a minimal XSLT stylesheet that simply copies the XML content, then create a transformation scenario that applies the stylesheet over the XML. The XSLT stylesheet would look like this:

```
<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 Developer 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:

```
<xi:include href="section2.xml" xmlns:xi="http://www.w3.org/2001/XInclude"
  set-xml-id="sectInner1"/>
```

and `section2.xml` looks like this:

```
<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 Developer 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 334)*.

Note:
The built-in refactoring operations are also available from the **Refactoring** submenu in the contextual menu of **Text** mode. This is useful because by selecting the operations from the contextual menu, Oxygen XML Developer 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 581)* or previewed *(on page 581)* also appear in the **Refactoring** submenu of the contextual menu in the **Project** view.

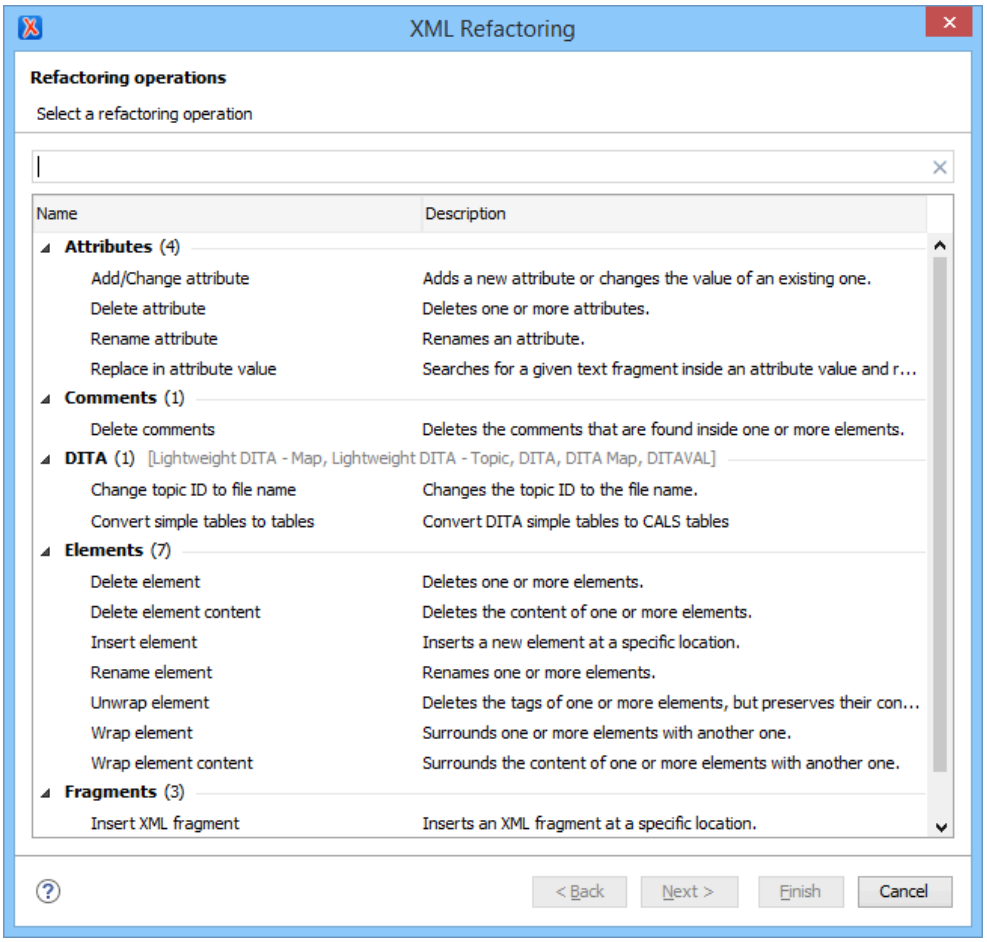
XML Refactoring Wizard

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 150. XML Refactoring Wizard

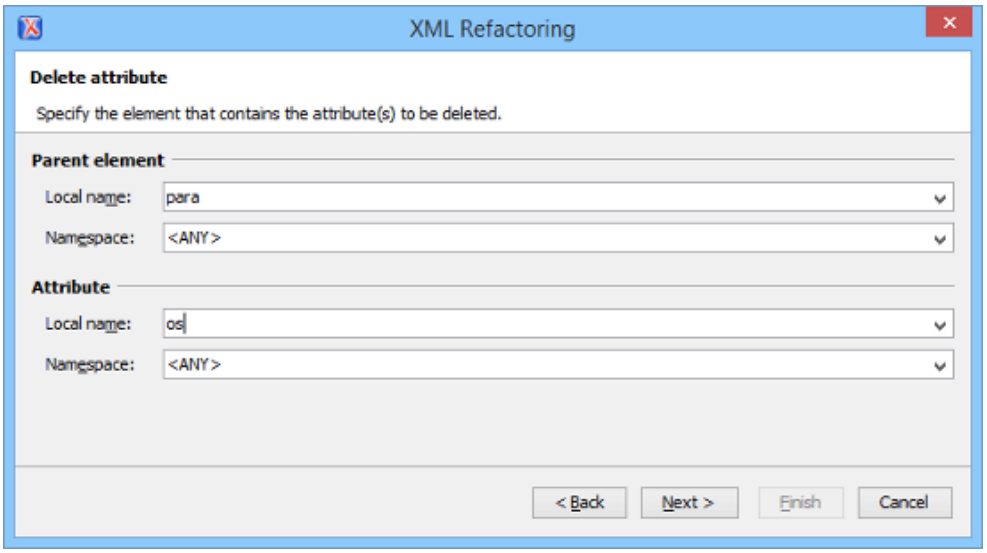


Configure Operation Parameters

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.

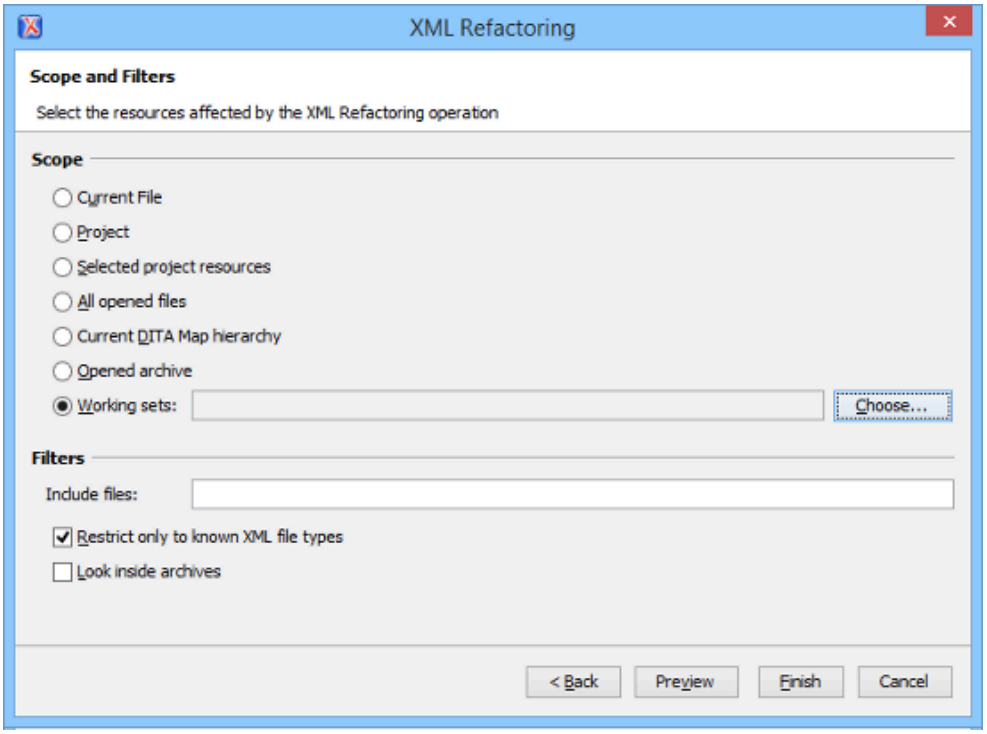
Figure 151. XML Refactoring 2nd Wizard Page (Delete Attribute Operation)



Scope and Filters

The last wizard page allows you to select the set of files that represent the input of the operation.

Figure 152. XML Refactoring - Scope and Filters Wizard Page



Scope section

You can specify the scope for the operation by selecting from predefined resource sets or you can define your own set of resources by creating a *working set (on*

page 2309) (select **Working sets** and click the **Choose** button to the right). If you select **Project**, all files attached to the current project will be used for the scope of the operation. If you select **Current DITA Map hierarchy**, the current DITA map that is open in the DITA Maps Manager along with all of its referenced topics and submaps (and topics referenced in those submaps) are used for the scope.

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 Developer 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 **Text** mode. The operations are also grouped by category in this submenu. When selecting the operations from the contextual menu, Oxygen XML Developer 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.

Convert Nested Topics to New Topics

Use this operation on topics that contain nested `<topic>` elements to convert each nested topic to a new topic.

Convert Sections to New Topics

Use this operation on topics that contain multiple sections to convert each section to a new 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).

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).

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

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**, **Entire 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. The new value is modified and reflected in DITA maps, DITA topics, and DITAVAL files.

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**, **Entire 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

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 1287)* that are included in a custom Publishing Template that was created in Oxygen XML Developer version 20.0 or 20.1 so that they will be compatible with Oxygen XML Developer version 21.0.

Migrate HTML Page Layout Files to v22

Use this operation to convert custom HTML page layout files *(on page 1287)* that are included in a custom Publishing Template that was created in Oxygen XML Developer versions 20.0 - 21.1 so that they will be compatible with Oxygen XML Developer version 22.0.

Update HTML Pages



Attention:
This operation is only used by Oxygen XML Developer 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 190)* 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.

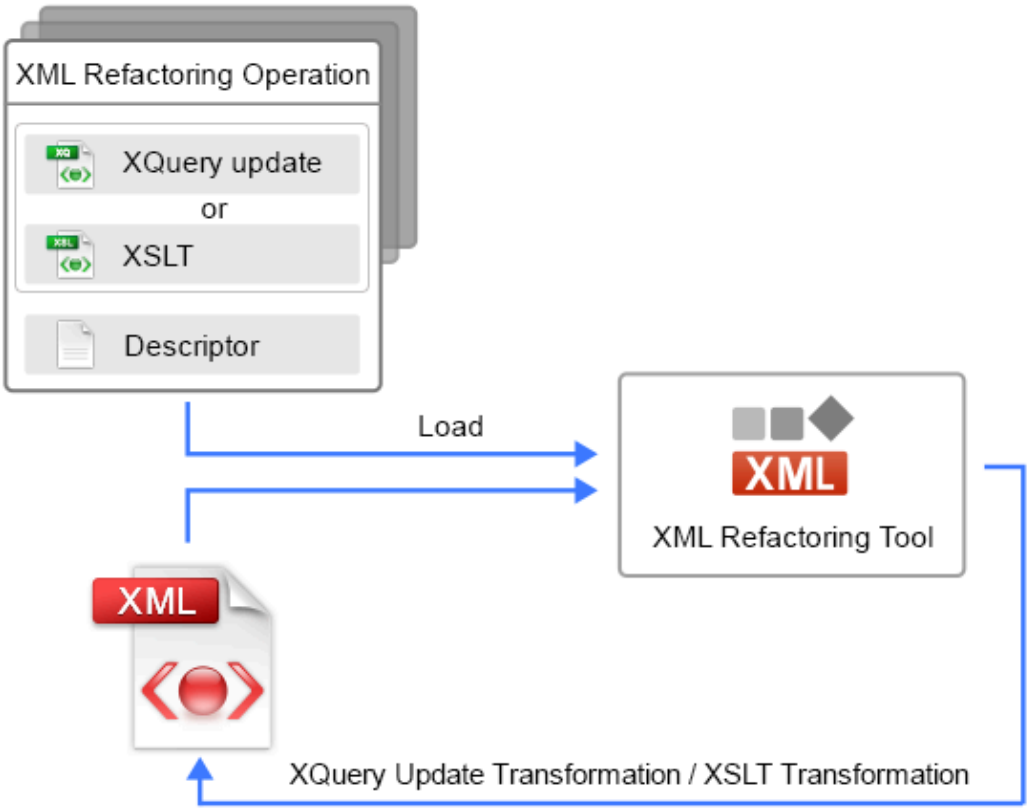
Custom Refactoring Operations

While Oxygen XML Developer 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 Developer 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 153. 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 579)*, along with the built-in operations.

After the user chooses an operation and specifies its parameters, Oxygen XML Developer 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 611)*.

  - 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 611)* 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 601)* or XSLT stylesheet file *(on page 606)*.
2. Create an XML refactoring operation descriptor file, that references the above script, as explained in these sections: Example descriptor file for an XQuery Update script *(on page 604)* or Example descriptor file for an XSLT stylesheet *(on page 609)*.
3. Store both files in one of the locations that Oxygen XML Developer *(on page 613)* 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 579)*.

Related information
Storing and Sharing Refactoring Operations *(on page 613)*

Custom Refactoring Script

The first step in creating a custom refactoring operation is to create an XQuery Update script *(on page 601)* or XSLT stylesheet *(on page 606)* 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 601)* or XSLT method example *(on page 606)* to help you with the content.

There are cases when it is necessary to add parameters in the XQuery script *(on page 601)* or XSLT stylesheet *(on page 606)*. 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 Developer. 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 1190)* 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 1122)*.

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 190)*.

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 604)* or XSLT stylesheet *(on page 609)*.

Related Information:
XQuery Update Script for Creating a Custom Operation *(on page 601)*
XSLT Stylesheet for Creating a Custom Operation *(on page 606)*

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 descriptor file root element specifies required attributes to define the operation `@name`, `@description`, and `@id` which are necessarily when loading an XML Refactoring operation. It also contains the path to the XQuery Update script *(on page 601)* or XSLT stylesheet *(on page 606)* that is associated with the particular operation through the `<script>` element.

The optional `@filesFilter` attribute can be specified to filter the resources by using a file pattern or list of file patterns separated by a comma (for example: `filesFilter="*.dita, *.xml"`). When set, its value is automatically populated in the **Include files** field within the **Scope and Filters** wizard page *(on page 581)* as a default value.

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 global 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 579)* 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 579)*.

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 Developer 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 190)* 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 Developer 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 2308)* for an element or attribute.
- **BOOLEAN** - Used to edit boolean parameters. For the **BOOLEAN** type, the possible values are "true" and "false". The equivalent parameter defined in the XSLT stylesheet or XQuery script must have the `xs:boolean` type.
- **TEXT_CHOICE** - It is useful for parameters whose value should be from a list of possible values. Oxygen XML Developer renders each possible value as a radio button option.

- **optional** - Specifies whether the parameter is optional or required. For optional parameters, the end user is not required to fill in a value in the XML refactoring wizard. By default if this attribute is missing a parameter is required.
- **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:

```
<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>
```

On a `<value>`, you can specify the `@default` attribute with the value `true` to mark it as the default presented value in the XML refactoring wizard. The text specified inside the `<value>` element is displayed as placeholder default text in the text entry box. If the dialog box is accepted with the placeholder text in place, the `@name` attribute value is passed to the refactoring script. Example:

```
<value name="my-actual-default-value" default="true">[default displayed]</value>
```

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 Developer 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>`:

```
<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>`:

```
<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>
```

Grouping Parameters in the Descriptor File

You can use `<section>` elements to group related parameters in the descriptor file:

```
<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>
```

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 609)*
Example of an Operation Descriptor File with an XQuery Update script *(on page 604)*

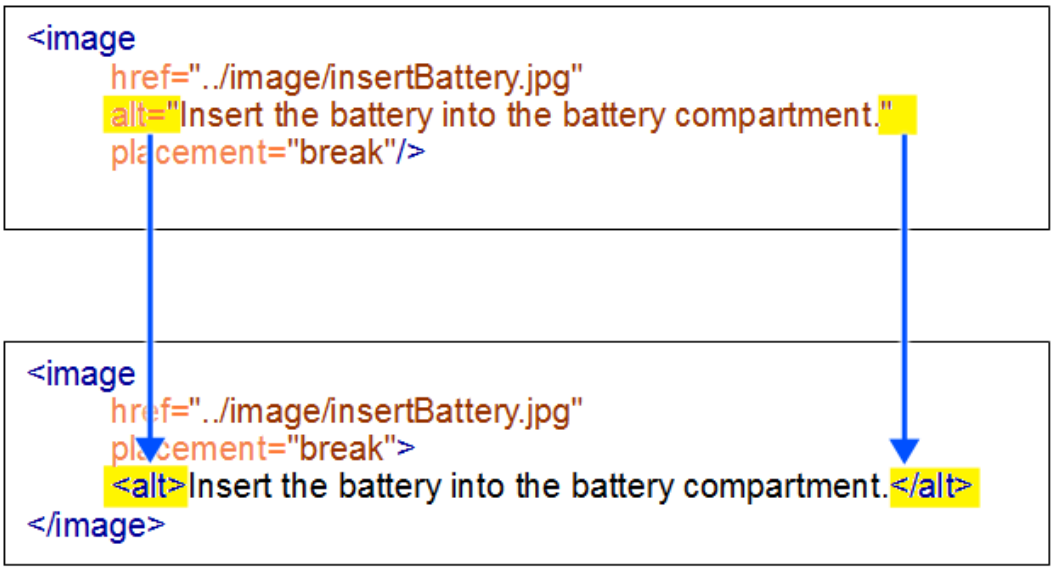
XQuery Update Script for Creating a Custom Operation

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.

Figure 154. Example: Custom XML Refactoring Operation



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 604)* 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 611)* 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 2308)* of the new element to be inserted and inserts it as the first child of the parent element.

```
(:
    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) = $element_localName)
  and
($element_namespace = '<ANY>' or $element_namespace = namespace-uri($node))


 return
   if (exists($attribute)) then
     (insert node element {QName($new_element_namespace, $new_element_qName)}
     {string($attribute)} as first into $node,
     delete node $attribute)
     else ()
```

Example of an Operation Descriptor File That References the XQuery Script for Creating a Custom Operation to *Convert an Attribute to an Element*

After you have developed the XQuery script (for example, named `convert-attribute-to-element.xq`), you have to create an XML Refactoring operation descriptor (for example, named `convert-attribute-to-element.xml`) that references the stylesheet and provides descriptions and possible values for its parameters. 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
 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>
 <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>
```

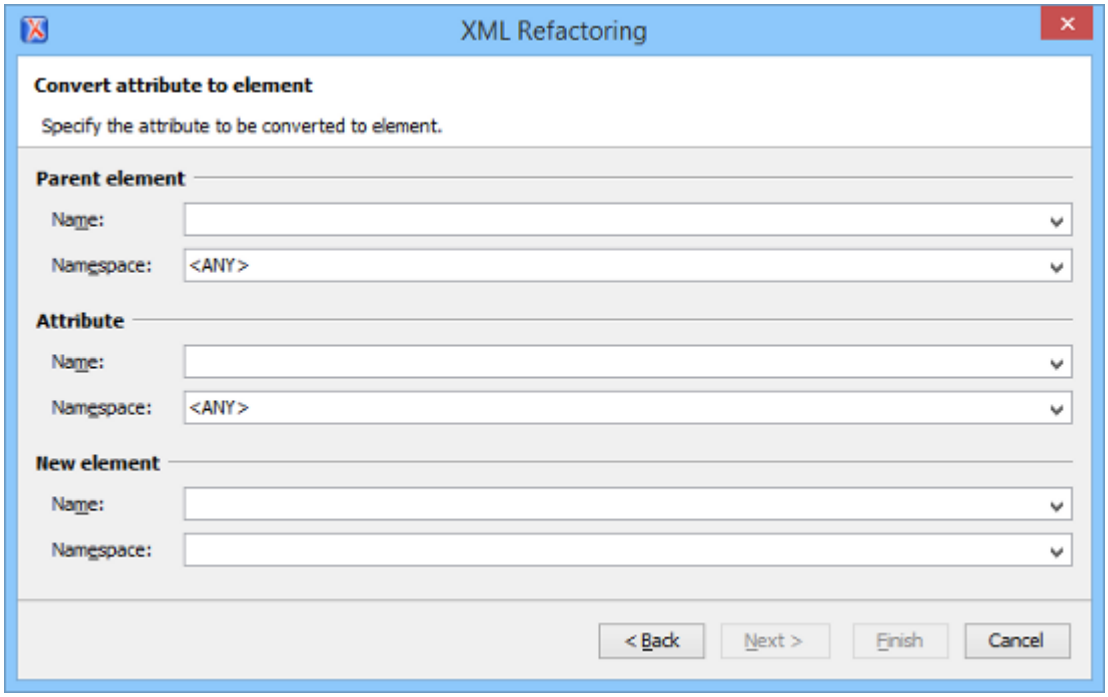
```
    </elementParameter>
   </section>
   <section label="Attribute">
    <attributeParameter dependsOn="elemID">
     <localName label="Name" name="attribute_localName">
      <description>Name of the attribute to be converted.</description>
     </localName>
    <namespace label="Namespace" name="attribute_namespace" 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>
  </parameters>
</refactoringOperationDescriptor>
```

Results

After you have created these files, copy them into a folder scanned by Oxygen XML Developer when it loads the custom operation *(on page 613)*. 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 155. Example: XML Refactoring Wizard for a Custom Operation



Debugging XQuery Refactoring Operations

You can use the XQuery trace() function to generate debugging information messages in the application's **Results** view *(on page 474)*.

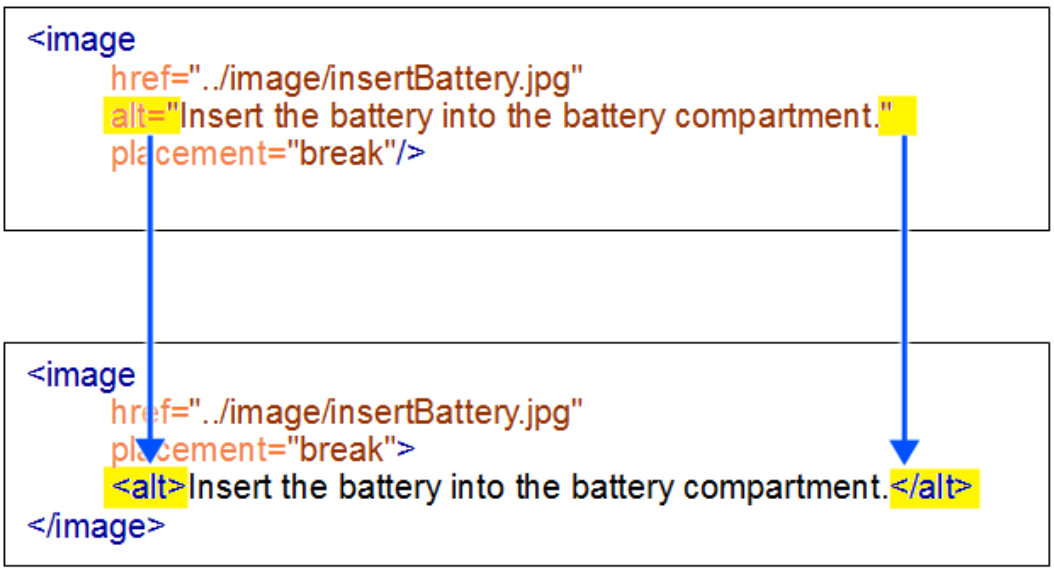
The Saxon processor used for XQuery update processing also supports the EXPath extensions, which also allow you to write debugging information to an output file.

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 156. 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 609)* 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 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:import
href="http://www.oxygenxml.com/ns/xmlRefactoring/resources/commons.xsl"/>

  <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>
                <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: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 2309)* set in the XML Refactoring *framework (on page 2305)*.

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 (for example, named `convert-attribute-to-element.xsl`), you have to create an XML Refactoring operation descriptor (for example, named `convert-attribute-to-element.xml`) that references the stylesheet and provides descriptions and possible values for its parameters. 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
 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>
 <script type="XSLT" href="convert-attribute-to-element.xsl"/>
  <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>
  </parameters>
</refactoringOperationDescriptor>
```

Note:

If you are using an XSLT file, the line with the `<script>` element would look like this:

```
<script type="XSLT" href="convert-attribute-to-element.xsl"/>
```

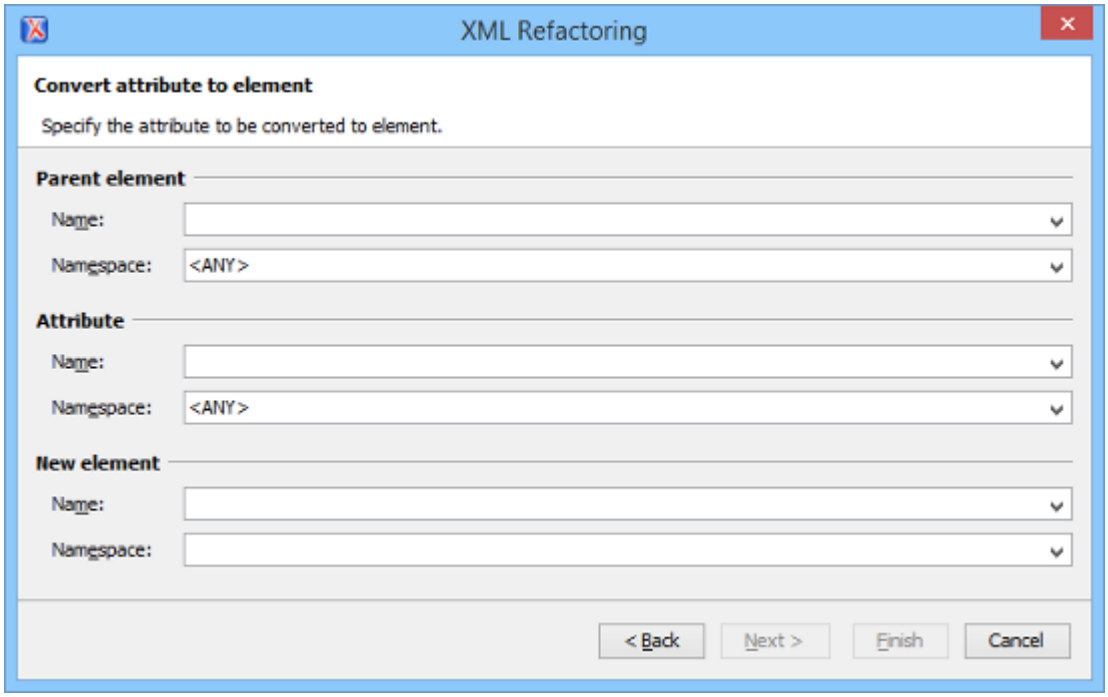
The code exemplified above and other refactoring examples can be found on the DITA Refactoring GitHub sample project.

Results

After you have created these files, copy them into a folder scanned by Oxygen XML Developer when it loads the custom operation *(on page 613)*. 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 157. Example: XML Refactoring Wizard for a Custom Operation



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 601)* 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 version="1.0" encoding="UTF-8"?>
<!-- comment before root -->
<?pi before root ?>
<root>
    <child></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:

  ```
  <!-- 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 version="1.0" encoding="UTF-8"?>
  <!-- comment before root -->
  <?pi before root ?>
  <root>
      <child></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 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:

  ```
  <!-- Inserted comment --><root>
      <child></child>
  </root>
  <!-- comment after root -->
  <?pi after root ?>
  ```

XSLT Example:

To process content after the root node, the XSLT would look like this:

```
<?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()"/>

        <xsl:variable name="processedContent"
                      select="concat($after-root-content, $commentAsText)"/>

        <!-- Update the content after the root element -->
        <xsl:value-of select="xrf:set-content-after-root($processedContent)"/>

        <xsl:apply-templates/>
    </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 Developer scans the following locations when looking for XML Refactoring operations to provide flexibility:

- A folder named `refactoring`, created inside the folder of the *framework* you are customizing. In the **Classpath** tab of the **Document type** configuration dialog box *(on page 99)*, you need to add a reference to the `refactoring` folder specific for the framework.
- A folder that you specify in the **Load additional refactoring operations from** text box *(on page 200)* in the **XML Refactoring** preferences page *(on page 200)*.

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 2308)*:
    a. Open the **Preferences** dialog box **(Options > Preferences)** *(on page 79)* and go to **XML > XML Refactoring** *(on page 200)*.
    b. Select **Project Options** *(on page 2308)* at the bottom of the dialog box.
3. In the **Load additional refactoring operations from** text box *(on page 200)*, use the *${pd}* editor variable *(on page 259)* so that the folder path is declared relative to the project.

- A folder specified by the XML Refactoring Operations Plugin Extension *(on page 1879)*.
- The `refactoring` folder from the Oxygen XML Developer installation directory (*`[OXYGEN_INSTALL_DIR]/refactoring/`*).

Sharing Custom Refactoring Operations

The purpose of Oxygen XML Developer 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 2305)* or projects.

After storing custom operations, you can share them with other users by sharing the resources.

Localizing XML Refactoring Operations

Oxygen XML Developer 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 `<description>` of the `<parameters>` element.
- The `label`, `description`, and `possibleValues` for each `parameter`.

Translated refactoring information uses the following form:

```
${i18n(translation_key)}
```

Oxygen XML Developer 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 79)*, 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 1879)*.
- The `refactoring/i18n` folder from the Oxygen XML Developer installation directory (*`[OXYGEN_INSTALL_DIR]`*`/refactoring/i18n`).

Example: Refactoring Operation Descriptor File with *i18n* Support

```
<?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>
```

Applying All Default Quick Fix Proposals

The **Apply all default quick fix proposals** tool can be used to apply quick fix proposals for all reported validation errors in one or more documents when one or more quick fix proposals have been detected for reported validation errors. If multiple quick fixes are available for the same validation error, the default quick fix defined in the Schematron validation schema using the `sqf:default-fix` attribute is automatically selected to be applied. If no default quick fix is explicitly specified, the first quick fix proposal in the list is considered. All quick fix proposals are then automatically executed in bulk, one after the other.

Important Notes to Consider:

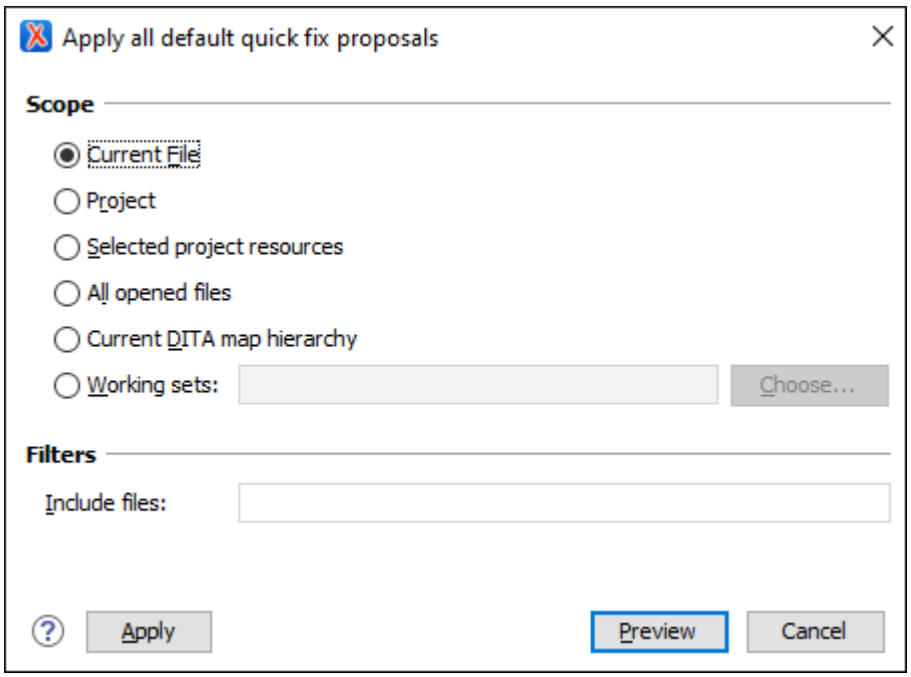
- To maintain the accuracy of the initially calculated error validation ranges, the quick fix proposals are applied in the reverse order of their selection.
- If two or more quick fixes act on the same "area" within the document, only one is applied (no changes can be made to changes already made).
- Quick fixes that involve "user-entered values" that normally present a dialog box to facilitate data entry will not be executed (the automatic process of applying all selected quick fixes cannot be interrupted by the presence of the respective dialog boxes).

The **Apply all default quick fix proposals** tool can be accessed from the following interface locations:

- **[Single Document]** The contextual menu when you right-click anywhere in **Text** mode *(on page 492)* when there is a detected quick fix proposal in the current document.
- **[One or More Documents]** The **Tools** menu.
- **[One or More Documents]** The **Refactoring** submenu when you right-click one or more files in the **Project** view *(on page 334)*.

Invoking the tool opens the **Apply all default quick fix proposals** dialog box where you can choose the scope for the operation and optionally specify a filter to narrow the scope.

Figure 158. Apply All Default Quick Fix Proposals Dialog Box



The **Scope** section allows you to choose from the following scopes:

- **Current File** - The quick fix proposals are applied on the current file only.
- **Project** - The quick fix proposals are applied on the current project.

- **Selected project resources** - The quick fix proposals are applied on the selected files from the current project.
- **All opened files** - The quick fix proposals are applied on all opened files.
- **Entire DITA map hierarchy** (for DITA documents) - The quick fix proposals are applied on current DITA map along with all of its referenced topics and submaps (and topics referenced in those submaps).
- **Working sets** - The quick fix proposals are applied on a user-defined set of resources, referred to as a *working set (on page 2309)*.

The **Include files** option in the **Filters** section can be used to narrow the scope of the operation by filtering the selected resources using a file pattern. For example, to restrict the operation to only parse and update XML documents from the selected files, you could use `{*}.xml` for the file pattern.

You can click the **Preview** button to see a comparison preview that provides an overview of the content changes that will be made, according to the quick fixes that will be applied. The comparison panel also informs you of any problems encountered. While previewing the changes, you can also choose to exclude certain files from having the quick fixes be applied if you do not agree with the changes for the particular files. If you agree with the changes presented, click **Apply** to trigger the quick fixes and update the content.

Troubleshooting:
If a quick fix did not apply, a notification with some information about the error will be displayed in the **Results** view at the bottom of the application.

Tip:
If the operation takes longer than expected, you can use the **Stop** button in the progress bar to cancel the operation.

Resources

For more information about the **Apply all default quick fix proposals** tool, see our video: Introducing the Apply All Default Quick Fix Proposals Tool.

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 2303)* 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 2303)* document. The *canonical (on page 2303)* form of an XML document is physical representation of the document produced by the method described in this specification. The *XML canonicalization (on page 2303)* 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 2303)* 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 Developer:

- **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 2303)* 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 620)*, **Sign** *(on page 622)*, and **Verify Signature** *(on page 625)*, are available from the **Source** submenu when invoking the contextual menu in **Text** mode or from the **Tools** menu.

Related Information:
Certificates *(on page 620)*
Canonicalizing Files *(on page 620)*
Signing Files *(on page 622)*

Verifying Signature *(on page 625)*
Example of How to Digitally Sign XML Files or Content *(on page 625)*

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 *keystores (on page 2306)*.

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 Developer 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*.

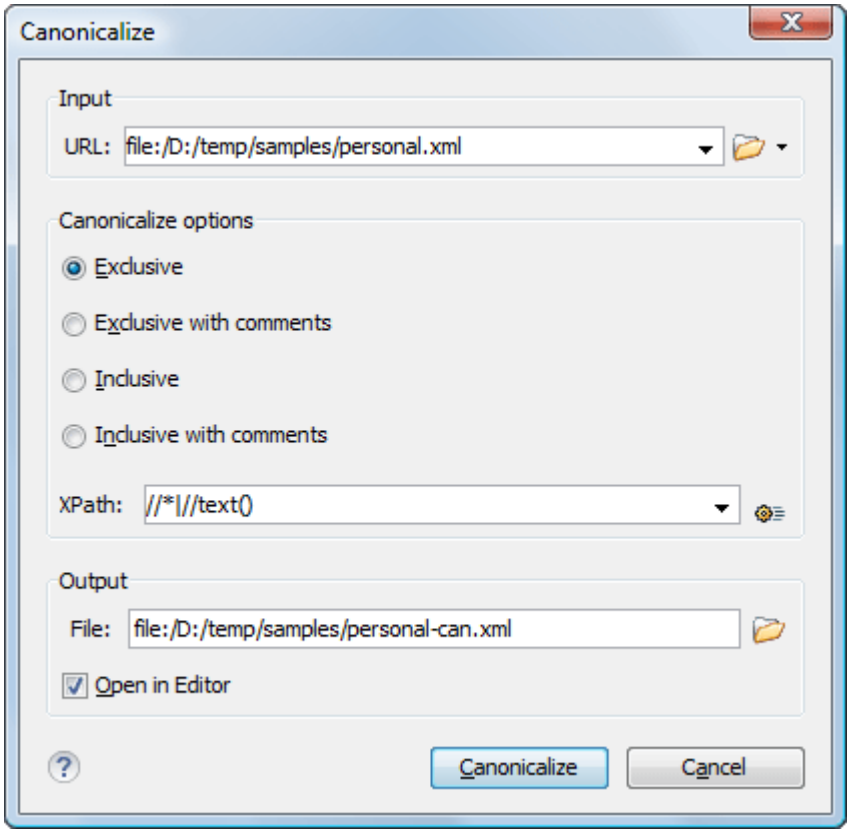
To configure the options for a certificate or to validate it, open the **Preferences** dialog box **(Options > Preferences)** *(on page 79)* and go to **XML > XML Signing Certificates**. This opens the certificates preferences page *(on page 199)*.

Related Information:
Digital Signatures Overview *(on page 617)*

Canonicalizing Files

You can select the *canonicalization (on page 2303)* 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.

Figure 159. Canonicalization Settings Dialog Box



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 2303)* 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 2303)* method is used.
- **Inclusive** - If selected, the inclusive (uncommented) *canonicalization (on page 2303)* 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.

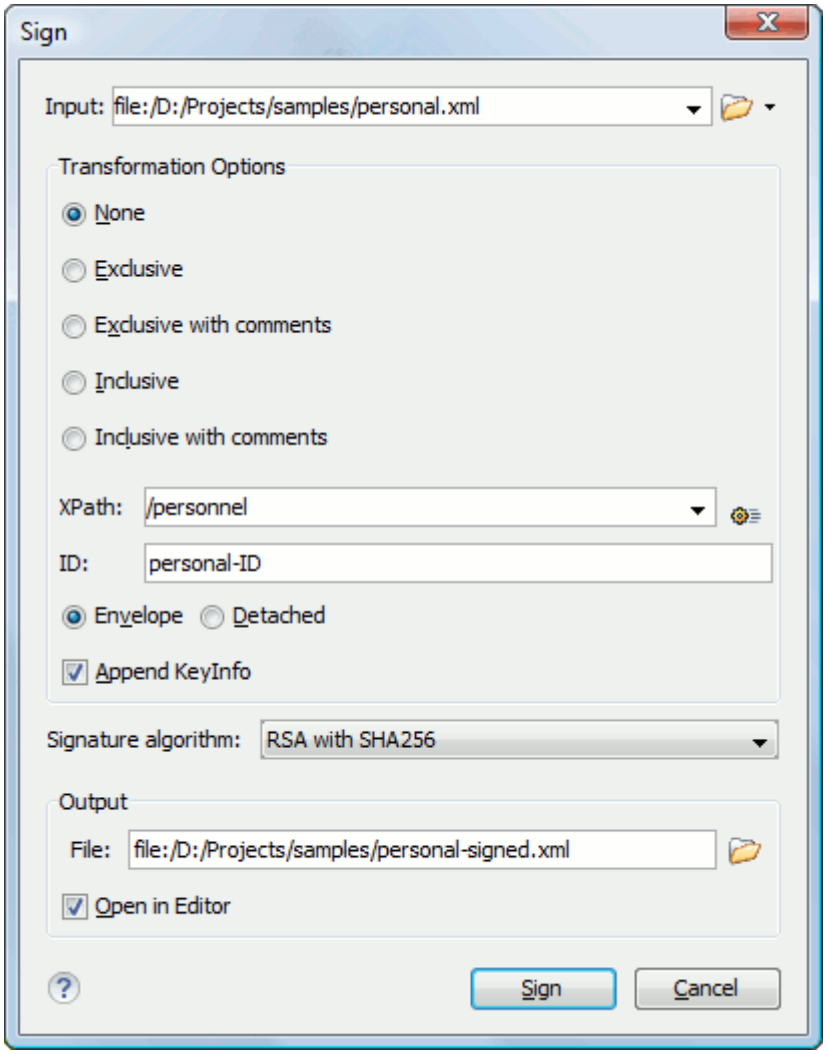
- **Inclusive with comments** - If selected, the inclusive with comments *canonicalization (on page 2303)* method is used.
- **XPath** - The XPath expression provides the fragments of the XML document to be signed.
- **Output** - Available if the **Canonicalize** action was selected from the **Tools** 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 617)*

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 160. Signature Settings Dialog Box



The following options are available:

Note:
If Oxygen XML Developer 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 199)* 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 617)* section for more information about these options.

- **None** - If selected, no *canonicalization (on page 2303)* algorithm is used.
- **Exclusive** - If selected, the exclusive (uncommented) *canonicalization (on page 2303)* 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 2303)* method is used.
- **Inclusive** - If selected, the inclusive (uncommented) *canonicalization (on page 2303)* 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 2303)* 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 617)* for more information.
- **Detached** - If selected, the *detached* signature is used. See the Digital Signature Overview *(on page 617)* 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**, **RSA with SHA512**, **ECDSA with SHA1**, **ECDSA with SHA256**, **ECDSA with SHA384**, and **ECDSA 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 Developer.

Related Information:
Digital Signatures Overview *(on page 617)*
Verifying Signature *(on page 625)*
Example of How to Digitally Sign XML Files or Content *(on page 625)*

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 617)*
Signing Files *(on page 622)*
Example of How to Digitally Sign XML Files or Content *(on page 625)*

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 Developer includes a signature tool that allows you to digitally sign XML documents or specific content.

The Oxygen XML Developer 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 Developer 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 Developer 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 199)* 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 623)*. 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 624)* 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 617)*
Signing Files *(on page 622)*
Verifying Signature *(on page 625)*

Editing XSLT Stylesheets

Oxygen XML Developer 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 299)*.
- **Open and Edit XSLT files** - XSLT files can be opened and edited in the source editor (**Text** mode *(on page 442)*).
- **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 Developer makes XSLT files more readable.

Resources

For more information about working with XSLT in Oxygen XML Developer, 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 Developer 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 cat set a main XSLT stylesheet either using the *main files* support from the **Project** view *(on page 350)*, 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 Developer 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 2306)* include:

- Correct validation of a module in the context of a larger stylesheet structure.
- *Content Completion Assistant (on page 2303)* displays all components valid in the current context.
- The **Outline** view *(on page 639)* displays the components collected from the entire stylesheet structure.

Resources

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 Referenced/Dependent Resources View *(on page 646)*
XSLT Component Dependencies View *(on page 649)*

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 Developer performs the validation of XSLT documents with the help of an XSLT processor that you can configure in the preferences pages *(on page 180)* according to the XSLT version.

For XSLT 1.0, the options are: Xalan (Deprecated), Saxon 6.5.5 and Saxon 12.9. For XSLT 2.0, the option is: Saxon 12.9. For XSLT 3.0, the option is Saxon 12.9.

To access the XSLT preferences *(on page 180)* quickly, use the **Validation options** action from the **Document > Validate** menu.

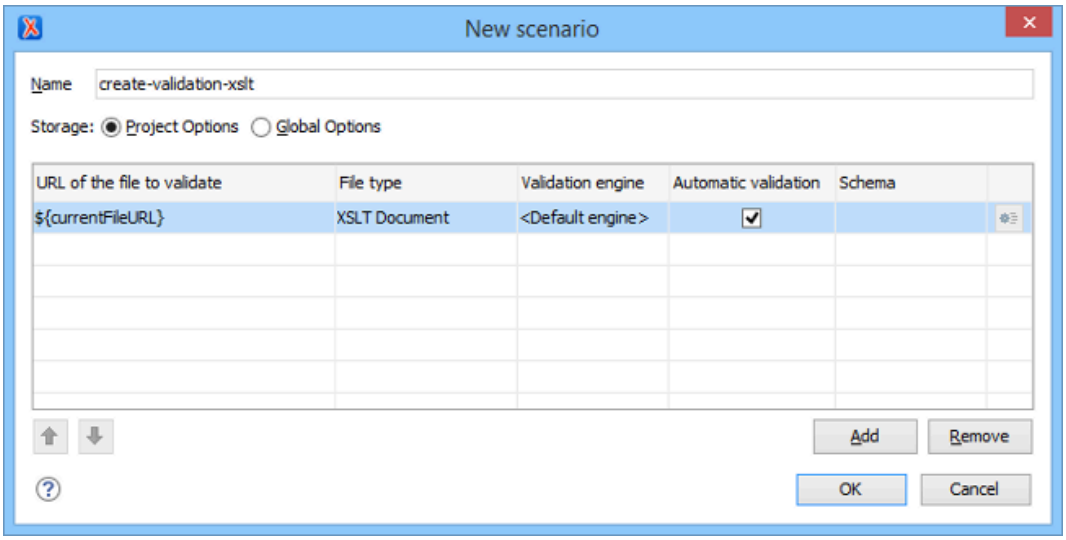
Creating a Validation Scenario for XSLT Stylesheets

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 79)* 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 2306)* 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 Developer, 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 334)*.
   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 Developer 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 161. Add / Edit a Validation Unit

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 2308)* or **Global Options** *(on page 2305)*.

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** drop-down menu or **Insert Editor Variable** *(on page 252)* 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.

5. To add or edit extensions, click the **Edit extensions** 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** or **Move down** 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)** 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 Developer preferences. After a custom validation engine is properly configured *(on page 161)*, 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.

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 on XML scenario *(on page 1159)* for your XSLT document (select **Configure Transformation Scenario(s)** action from the toolbar, then click **New**, and select **XSLT transformation on XML**).
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 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 1855)*

XSLT Quick Fix Support

The Oxygen XML Developer *Quick Fix* support *(on page 2308)* 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 ( ) is displayed in the stripe on the left side of the editor. If you click this icon, Oxygen XML Developer 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 + Option + 1 on macOS)** on your keyboard.

Note:
The *Quick Fixes* are available only when validating an XSLT file with Saxon HE/PE/EE.

Figure 162. Example of an Undefined XSLT Functions Quick Fix

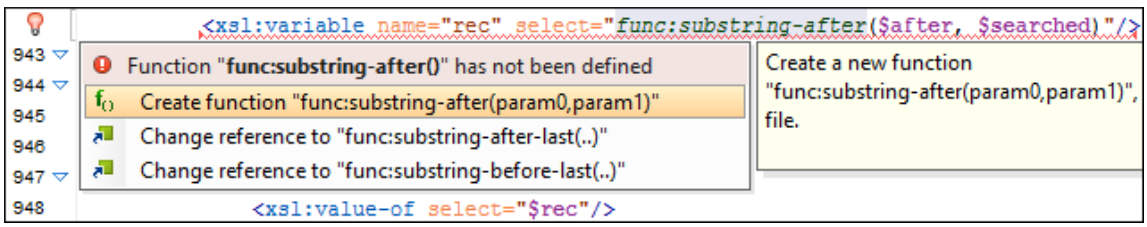
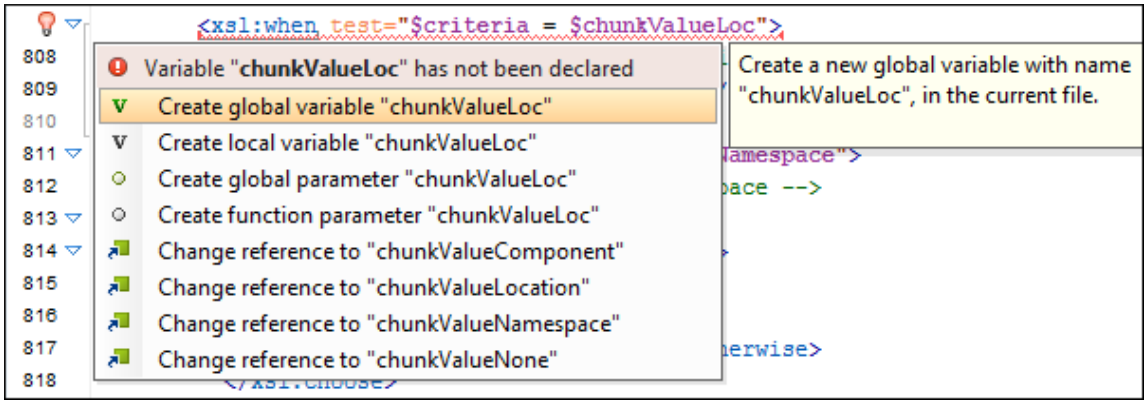


Figure 163. Example of an Undeclared XSLT Variables/Parameters Quick Fix



Oxygen XML Developer 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:

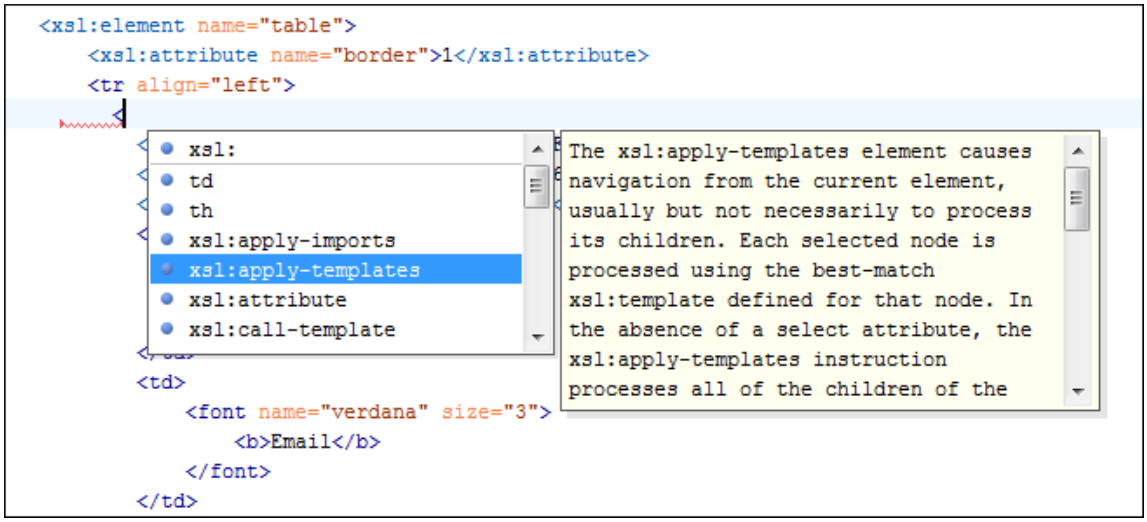
    - **Create parameter "paramName" in the template "templateName"** - creates a new parameter with the specified name in the referenced template.
    - **Change "paramName" parameter reference to "newParamName"** - changes the parameter reference from the `<call-template>` element to a parameter that is 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 "chacterMap" 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 2303)* 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** 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 146)* page and can be any of the following: XML Schema, DTD, RELAX NG schema, or NVDL schema.

Figure 164. XSLT Content Completion Assistant



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 146)* from the **Content Completion** preferences page.
- After typing a partial element or attribute name, you can manually activate it by pressing **Ctrl + Space** or **Alt + ForwardSlash (Command + Option + 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 462)* into stylesheets.

Note:
For XSL and XSD resources, the *Content Completion Assistant* collects its components starting from the *main files (on page 2306)*. 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 350)*.

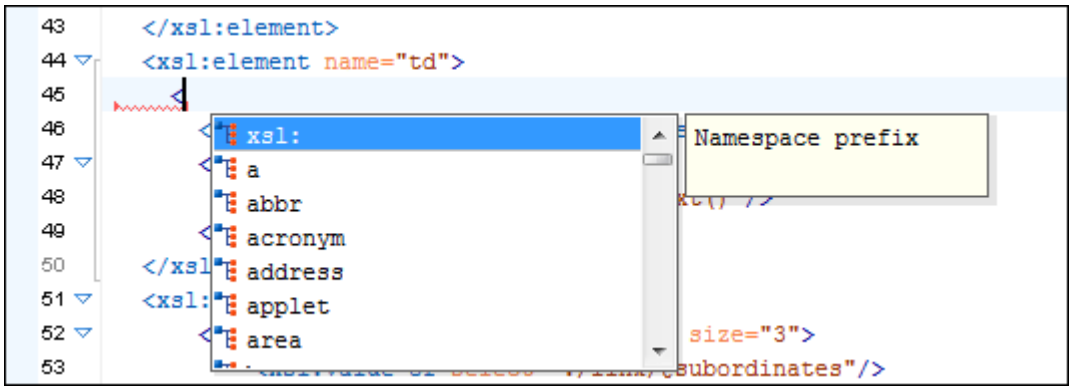
The extension functions included in the Saxon 6.5.5 and 12.9 transformation engines are presented in the content completion list only if the Saxon namespace (*http://saxon.sf.net* for XSLT version 2.0 / 3.0 or *http://icl.com/saxon* for XSLT version 1.0) is declared and one of the following conditions is true:

- The edited file has a transformation scenario that uses as transformation engine Saxon 6.5.5 (for XSLT version 1.0), Saxon 12.9 PE or Saxon 12.9 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 12.9 PE or Saxon 12.9 EE (for version 2.0 / 3.0).
- The validation engine specified in Options *(on page 180)* page is Saxon 6.5.5 (for version 1.0), Saxon 12.9 PE or Saxon 12.9 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 165. 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 Developer provides an easy mode to declare them by proposing a prefix for these namespaces.

Note:
For XSLT documents that are unversioned or have an unsupported version, the content completion in Oxygen XML Developer uses version 3.0 as the fallback.

Content Completion in XPath Expressions

In XSLT stylesheets, the *Content Completion Assistant (on page 2303)* provides all the features available in the XML editor *(on page 457)* 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 644)*.

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** 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:

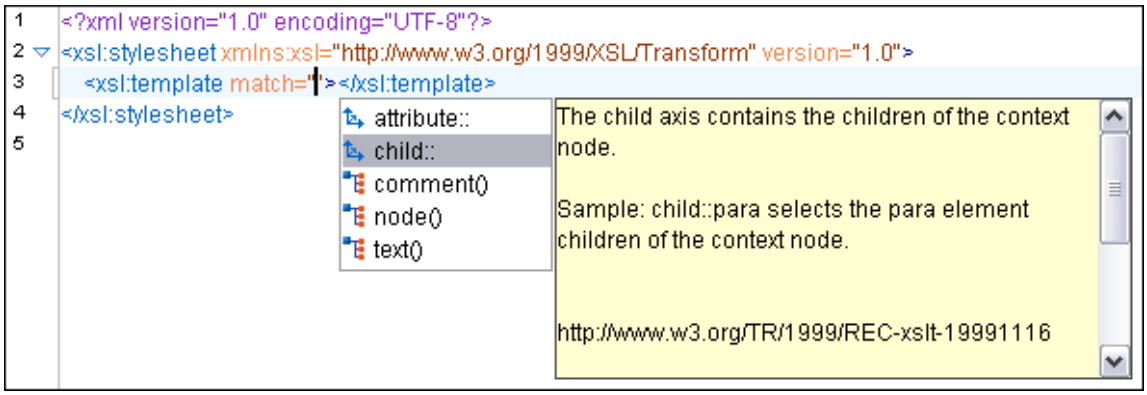
```
<personnel>
    <person id="Big.Boss">
        <name>
            <family>Boss</family>
            <given>Big</given>
        </name>
        <email>chief@oxygenxml.com</email>
        <link subordinates="one.worker"/>
    </person>
    <person id="one.worker">
        <name>
            <family>Worker</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.

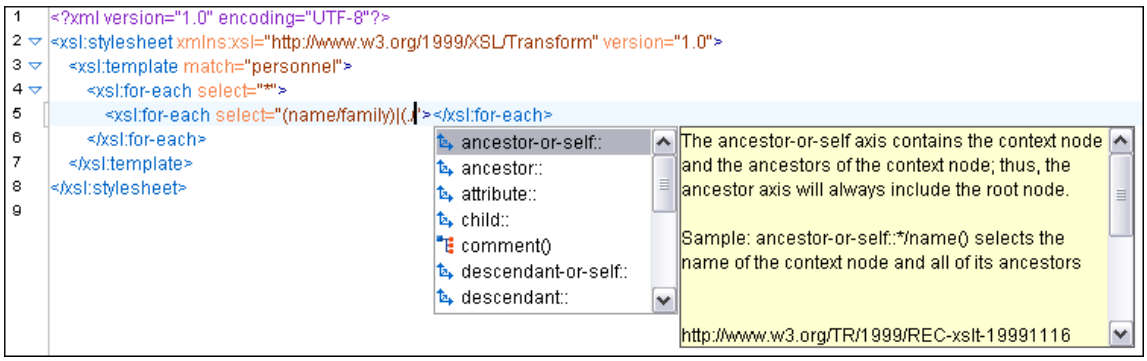
The set of XPath functions depends on the XSLT version declared in the root element `xsl:stylesheet`: 1.0, 2.0, or 3.0. Functions from other namespaces, such as maps, arrays, and math, are presented only if the namespaces are declared.

Figure 166. Content Completion in the `@match` Attribute



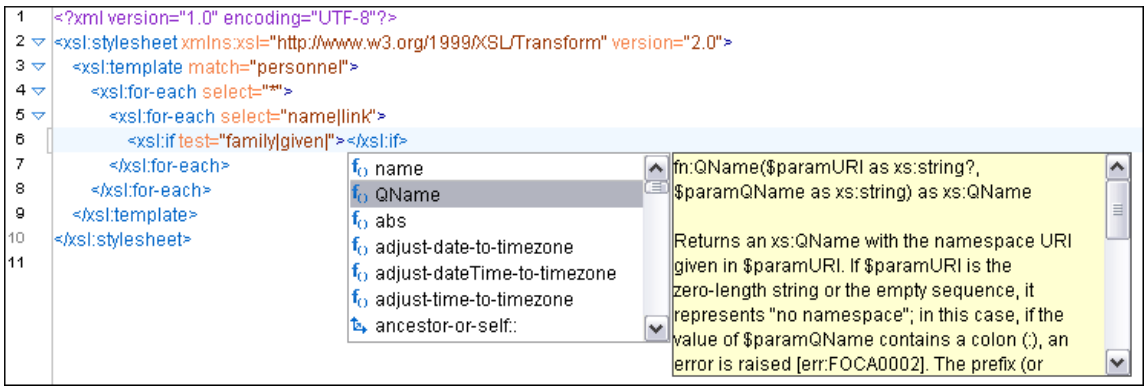
If the cursor is inside the `@select` attribute of an `<xsl:for-each>`, `<xsl:apply-templates>`, `<xsl:value-of>` or `<xsl:copy-of>` element the content completion proposals depend on the path obtained by concatenating the XPath expressions of the parent XSLT elements `<xsl:template>` and `<xsl:for-each>` as shown in the following figure:

Figure 167. Content Completion in the `@select` Attribute



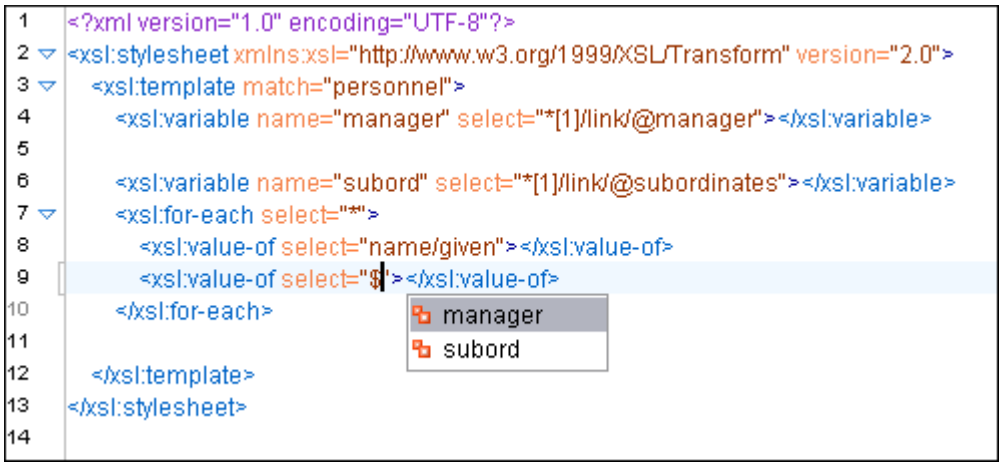
Also XPath expressions typed in the `@test` attribute of an `<xsl:if>` or `<xsl:when>` element benefit of the assistance of the content completion.

Figure 168. Content Completion in the `@test` Attribute



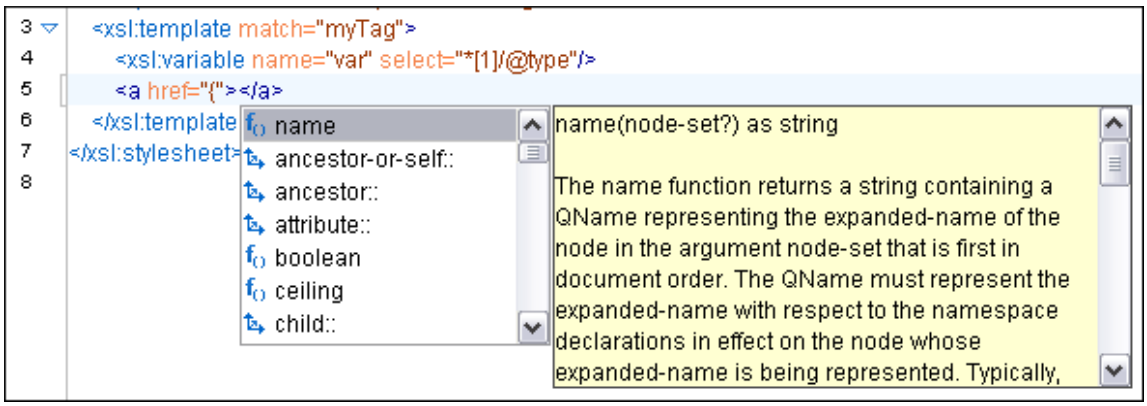
XSLT variable references are easier to insert in XPath expressions with the help of the content completion pop-up triggered by the `$` character, which signals the start of such a reference in an XPath expression.

Figure 169. Content Completion in the `@test` Attribute



If the `{` character is the first one in the value of the attribute, the same *Content Completion Assistant* is available also in attribute value templates of non-XSLT elements.

Figure 170. Content Completion in Attribute Value Templates



The time delay (configured in the **Content Completion** preferences page *(on page 146)*) is also applied for the content completion in XPath expressions.

Related Information:
Working with XPath Expressions *(on page 1726)*

Tooltip Helper for the XPath Functions Arguments

When editing the arguments of an XPath/XSLT function, Oxygen XML Developer 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*:

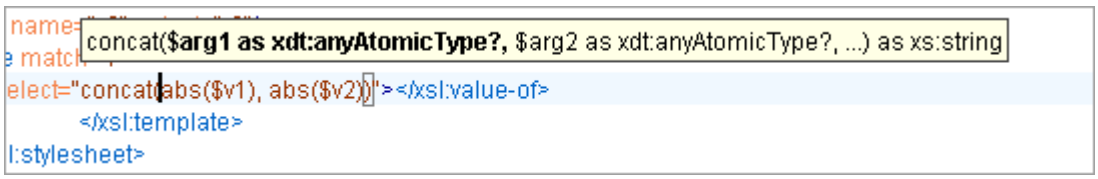
```
<xsl:template match="/">
    <xsl:value-of select="concat(abs($v1), abs($v2))"></xsl:value-of>
</xsl:template>
```

When moving the cursor before the first `abs` function, Oxygen XML Developer 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).

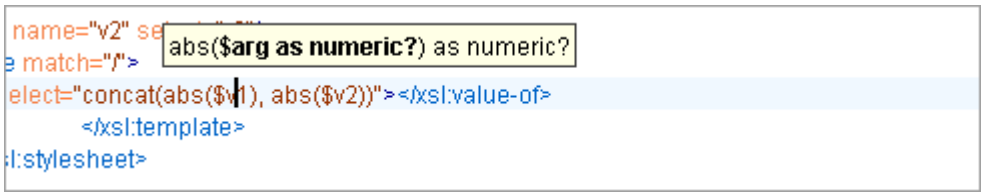
The function also takes other arguments that have the same type and returns a `xs:string`.

Figure 171. XPath Tooltip Helper - Identify the `concat` Function's First Argument



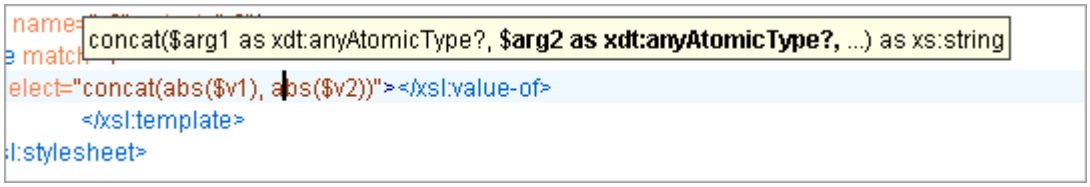
Moving the cursor on the first variable `$v1`, the editor identifies the `abs` as context function and shows its signature:

Figure 172. XPath Tooltip Helper - Identify the `abs` Function's Argument



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 173. XPath Tooltip Helper - Identify the `concat` Function's Second Argument



Note:
The tooltip helper is also available in the **XPath Builder** view *(on page 1729)* and XPath toolbar *(on page 1727)*.

Related Information:
Working with XPath Expressions *(on page 1726)*

Syntax Highlighting in XSLT

Oxygen XML Developer 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 79)*.
2. Go to **Editor > Syntax Highlight** *(on page 158)*.
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 Developer 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 159)*.

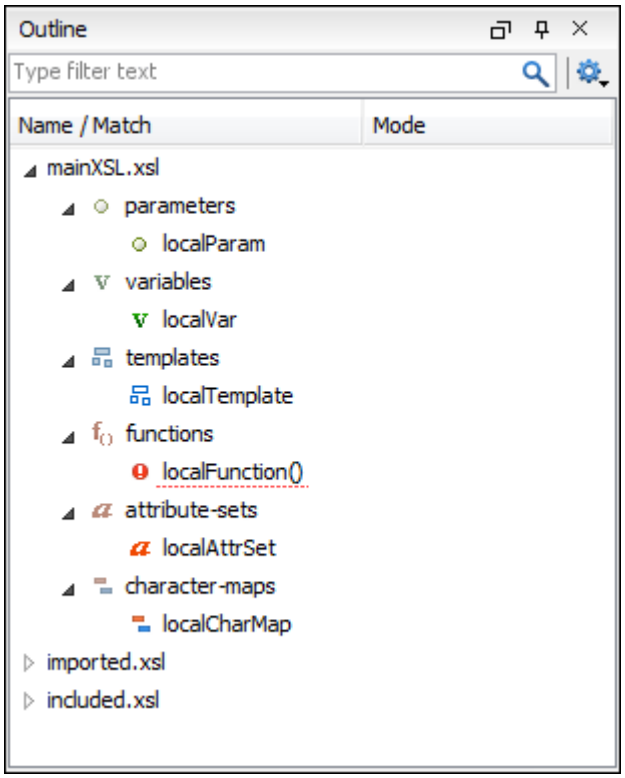
Related Information:
Customize Syntax Highlight colors *(on page 158)*

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 page 2306)*. 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 350)*.

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 174. 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 2306)*. 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 235)*.

Configure displayed attributes

Displays the **XML Structured Outline** preferences page *(on page 235)*.

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 651)* 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 651)* for more details.

Component Dependencies

Opens the **Component Dependencies** view *(on page 649)* that allows you to see the dependencies for the currently selected component.

Show referenced resources

Opens the **Referenced/Dependent Resources** view *(on page 646)* that displays the references for the currently selected resource.

Show dependent resources

Opens the **Referenced/Dependent Resources** view *(on page 646)* that displays the dependencies of the currently selected resource.

Rename Component in

Renames the selected component. See XSLT Refactoring Actions *(on page 655)* 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 649)* 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 Developer 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 Developer 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.

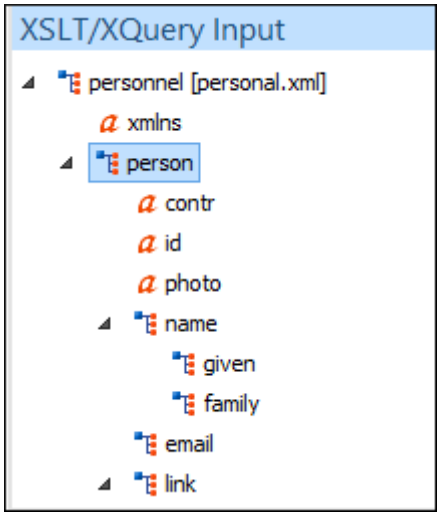
Note:
When no grouping is available and the table is not sorted, Oxygen XML Developer sorts the components depending on their order in the document. Oxygen XML Developer also takes into account the name of the file that the components are part of.

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.

Figure 175. XSLT Input View



Example:

For the following XML document:

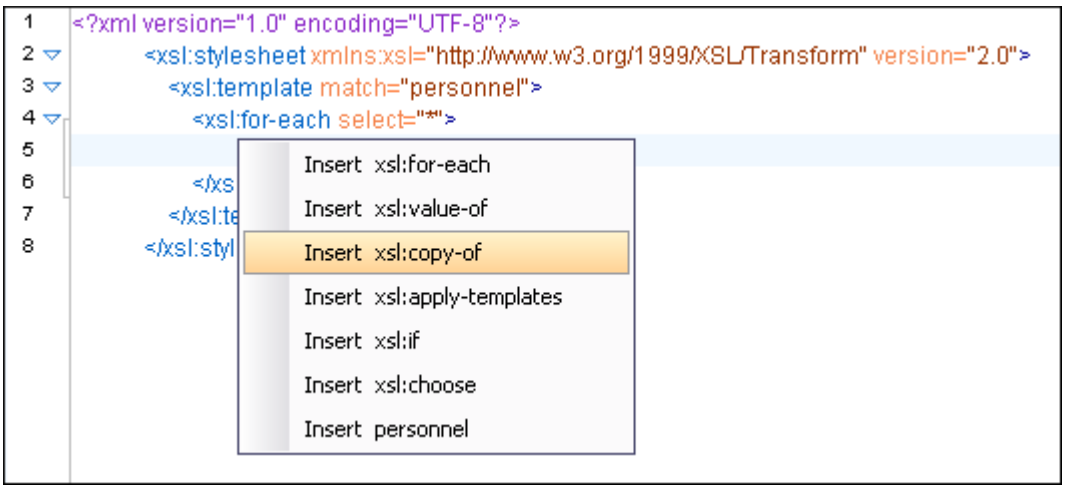
```
<personnel>
    <person id="Big.Boss">
        <name>
            <family>Boss</family>
            <given>Big</given>
        </name>
        <email>chief@oxygenxml.com</email>
        <link subordinates="one.worker"/>
    </person>
    <person id="one.worker">
        <name>
            <family>Worker</family>
            <given>One</given>
        </name>
        <email>one@oxygenxml.com</email>
        <link manager="Big.Boss"/>
    </person>
</personnel>
```

and the following XSLT stylesheet:

```
<?xml 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:for-each>
    </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:



if you select **Insert xsl:copy-of** (for example), the resulting document will look like this:

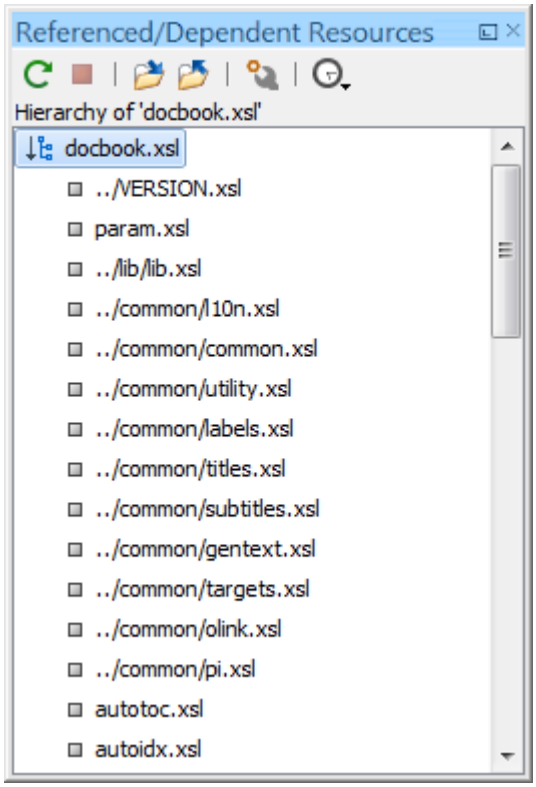
```
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="2.0">
    <xsl:template match="personnel">
        <xsl:for-each select="*">
            <xsl:copy-of select="name/given"/>
        </xsl:for-each>
    </xsl:template>
</xsl:stylesheet>
```

XSLT Referenced/Dependent Resources View

The **Referenced/Dependent Resources** 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 references or dependencies of a stylesheet, select the desired stylesheet in the **Project** view *(on page 334)* and choose **Show referenced resources** or **Show dependent resources** from the contextual menu.

Figure 176. Referenced/Dependent Resources View



The following actions are available on the toolbar of the **Referenced/Dependent Resources** view:

Refresh

Refreshes the resource structure.

Stop

Stops the computing.

Show hierarchy for

Computes the hierarchical structure of the references for a resource.

Show dependencies for

Computes the structure of the dependencies for a resource.

Configure dependencies search scope

Allows you to configure a scope to compute the dependencies. 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 **Referenced/Dependent Resources** view contains the following actions:

Open

Opens the resource. You can also double-click a resource within the hierarchical 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.

Show references resources

Shows the references for the selected resource.

Show dependent resources

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 hierarchical structure.

Collapse All

Collapses all children of the selected resource from the hierarchical structure.

 **Tip:**
When a recursive reference is encountered in the view, the reference is marked with a special icon .

Related Information:
Modular Contextual XML Editing Using 'Main Files' Support *(on page 567)*
Search and Refactor Operations Scope *(on page 569)*

Moving/Renaming XSLT Resources

You can move and rename a resource presented in the **Referenced/Dependent Resources** 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 **Referenced/Dependent Resources** 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 **Referenced/Dependent Resources** 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.

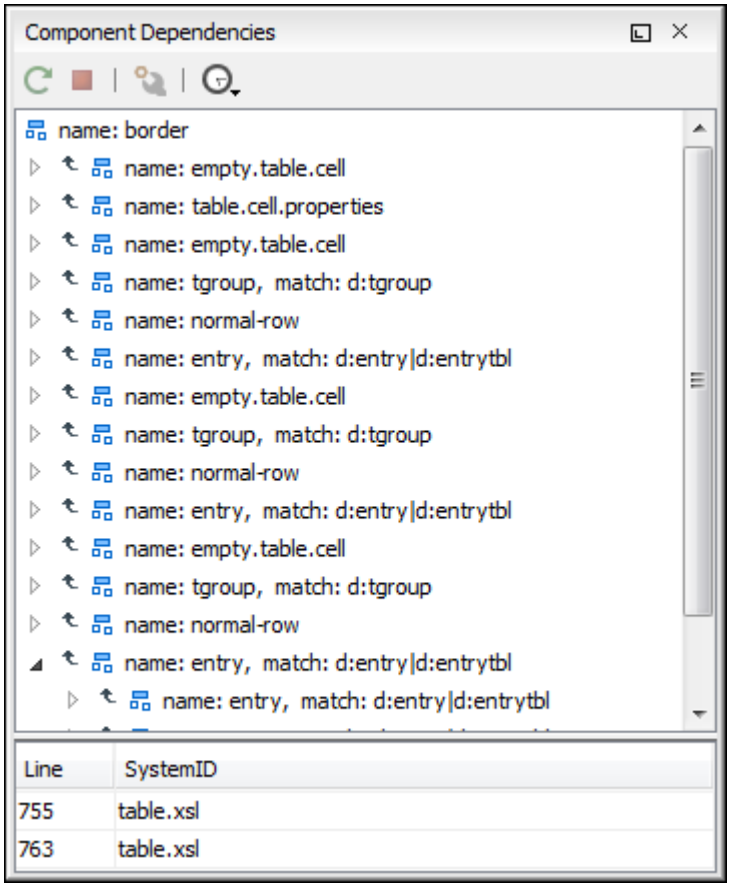
To see the dependencies of an XSLT component:

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 .

Figure 177. Component Dependencies View



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 569)*

Highlight Component Occurrences

When a component (for example variable or named template) is found at current cursor position, Oxygen XML Developer 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 Developer 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 79)* 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 474)*.

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 569)*

XSLT Stylesheet Component Documentation Support

Oxygen XML Developer offers built-in support for documenting XSLT stylesheets. If the expanded *QName (on page 2308)* 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 Developer 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 146)*.

Content Completion

When content completion is invoked inside an XSLT editor by pressing **Ctrl + Space**, 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 + Option + D on macOS)** or select **Add component documentation** from the contextual menu.

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 Developer Built-in Schema

```
<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:ref name="f:substring-after-last" type="function"
        xmlns:f="http://www.oxygenxml.com/doc/xsl/functions">See also
    </xd:ref>
    </xd:p>
  </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 Developer 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

```
<xd:doc xmlns:xd="http://www.oxygenxml.com/ns/doc/xsl" id="thisDoc">
   <xd:desc>
      <xd:p>
         <xd:ref name="test" type="variable">My test variable</xd:ref>
            <xd:a docid="thisDoc">Link to this documentation, see
the the id="thisDoc" above</xd:a>
            <xd:a docid="otherDocID" href="included.xsl">Link to
otherDocID defined in included.xsl</xd:a>
      </xd:p>
```

```
    </xd:desc>
</xd:doc>
```

Related Information:

Generating Documentation for an XSLT Stylesheet *(on page 665)*

XSLT 3.0 Text Value Templates

Oxygen XML Developer 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:

```
<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 634)*

XSLT 3.0 Packages (xsl:package Element)

Oxygen XML Developer 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 Developer 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 299)* 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 Developer 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. XPath expressions are rebuilt based on the current context and the context is passed as parameters in the new functions. 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 Developer 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:

```
<person id="Big{test}Boss"/>
```

is converted to:

```
<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"><xsl:text>john.doe@example.com</xsl:text>
  </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>
        |
    </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:

```
<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:

```
<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 Developer 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 Developer allows you to edit the name of the variable.

> **Note:**
> Oxygen XML Developer checks if the selected expression depends on local variables or parameters that are not available in the global context where the new variable is created.

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 Developer 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 Developer allows you to edit the name of the parameter.

Note:
Oxygen XML Developer 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.

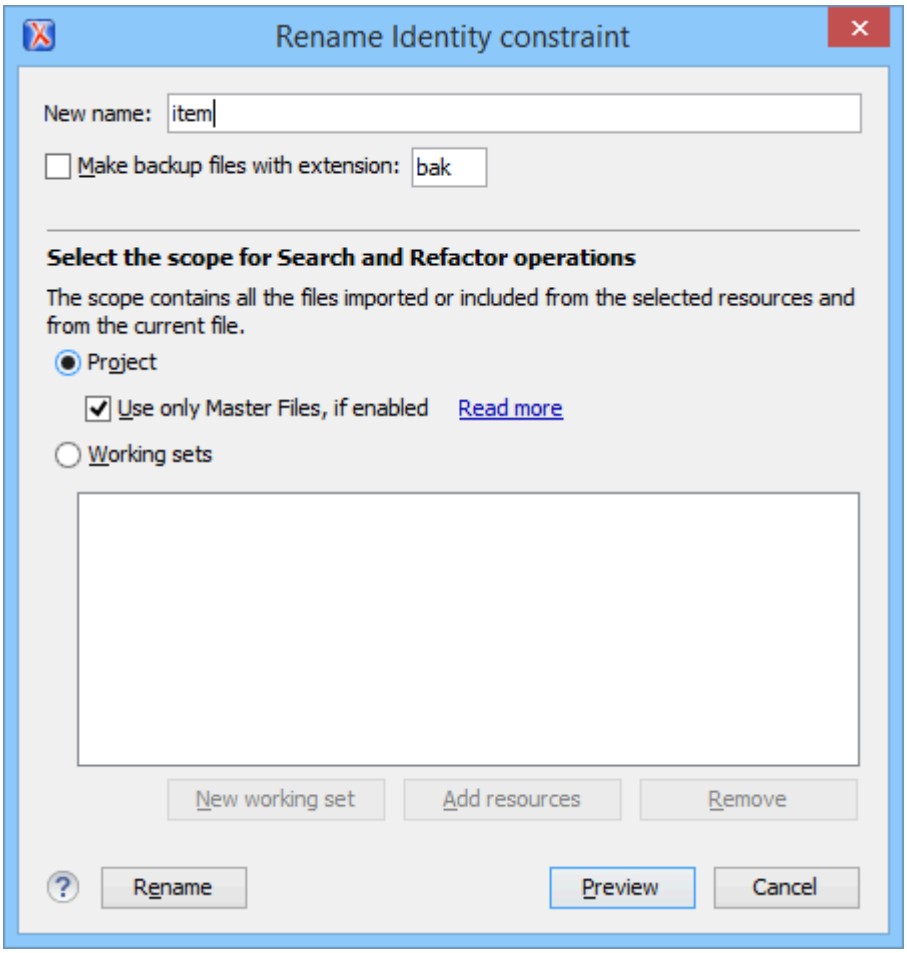
Rename Component

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.

Figure 178. Rename Identity Constraint Dialog Box



Note:
Many of these refactoring actions are also proposed by the *Quick Assist* support *(on page 660)*.

Resources

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 2308)* 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 + Option + 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.

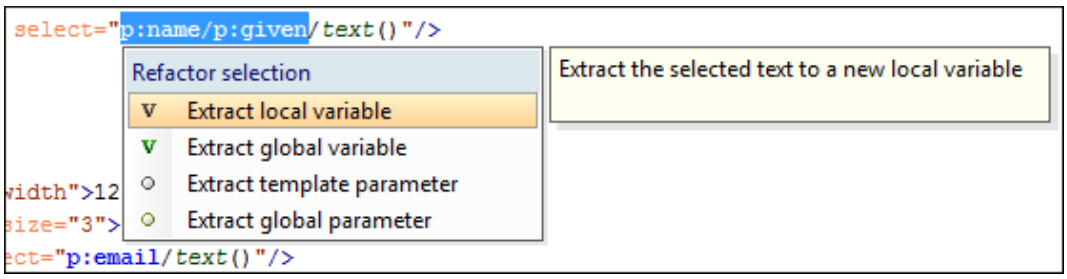
  Extract template parameter

  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 179. XSLT Quick Assist Support - Refactoring Actions



- Actions available when the cursor is positioned over the name of a component:

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.

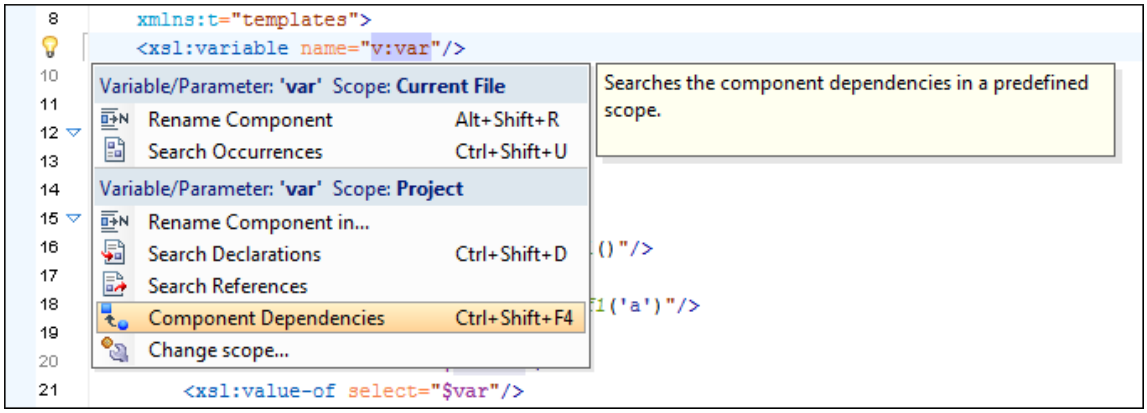
Rename Component

Allows you to rename the current component in-place.

Search Occurrences

Searches all occurrences of the component within the current file.

Figure 180. XSLT Quick Assist Support - Component Actions



Related information

Component Dependencies View *(on page 649)*

XSLT Hierarchy View *(on page 646)*

XSLT Refactoring Actions *(on page 655)*

Search and Refactor Operations Scope *(on page 569)*

XSLT Unit Test (XSpec)

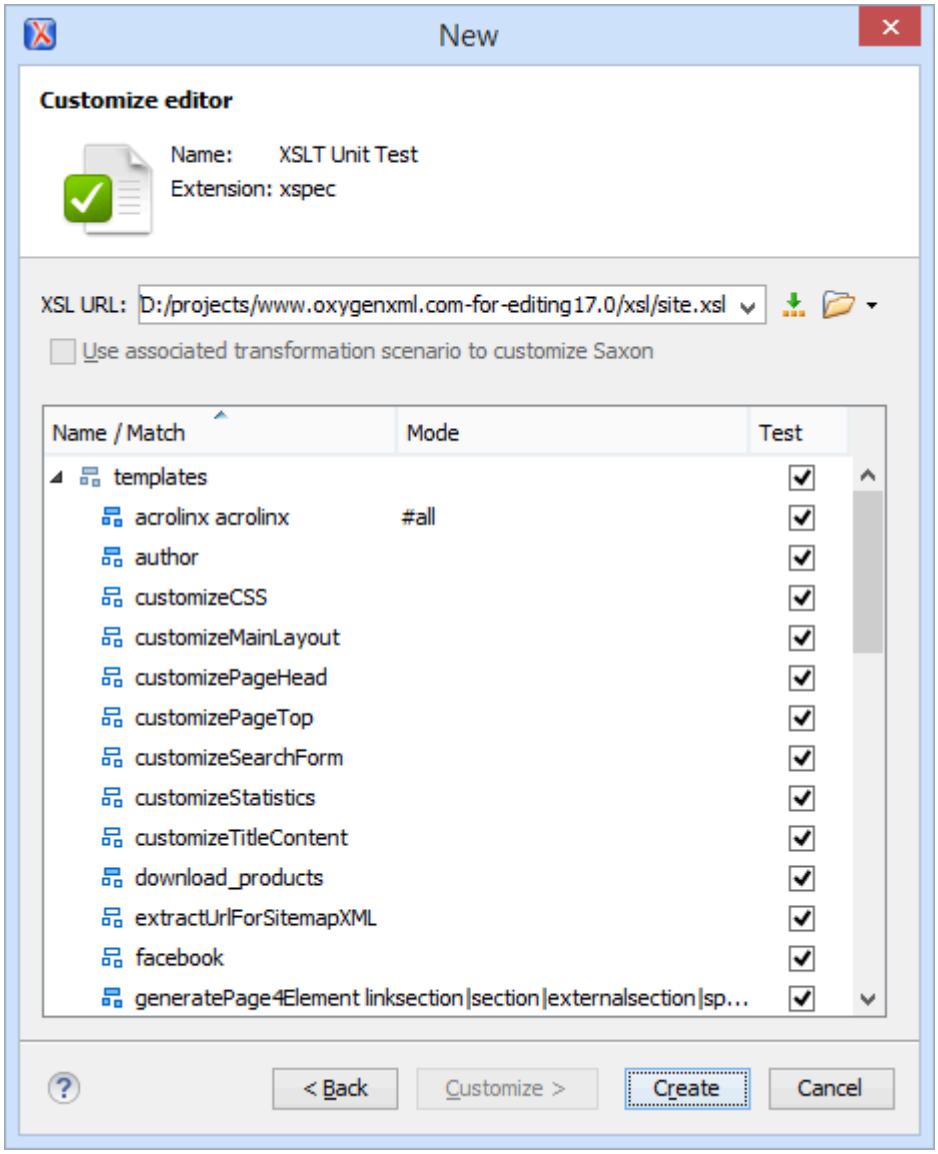
XSpec is a behavior driven development (BDD) *framework* for XSLT, XQuery, and Schematron. XSpec consists of syntax for describing the behavior of your XSLT, XQuery, or Schematron code, and some code that enables you to test your code against those descriptions.

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 334)*. Oxygen XML Developer 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 Developer uses information from the validation and transformation scenarios associated with the XSL file. From the transformation scenario Oxygen XML Developer uses extensions and properties of Saxon 12.9, improving the Ant scenario associated with the XSpec document.

Figure 181. New XSLT Unit Test Wizard



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 2305)*.

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.

```
<x:scenario label="when processing a para element">
    <x:context>
```

```
        <para>...</para>
    </x:context>
    ...
</x:scenario>
```

- Based on an external file, or on a part of an external file extracted with an XPath expression.

```
<x:scenario label="when processing a para element">
    <x:context href="source/test.xml" select="/doc/body/p[1]" />
    ...
</x:scenario>
```

- **Test a function**:

```
<x:scenario label="when capitalising a string">
  <x:call function="eg:capital-case">
    <x:param select="'an example string'" />
    <x:param select="true()" />
  </x:call>
  ...
</x:scenario>
```

- **Test a template with a name**:

```
<x:scenario label="when creating a table">
  <x:call template="createTable">
    <x:param name="nodes">
      <value>A</value>
      <value>B</value>
    </x:param>
    <x:param name="cols" select="2" />
  </x:call>
</x:scenario>
```

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:

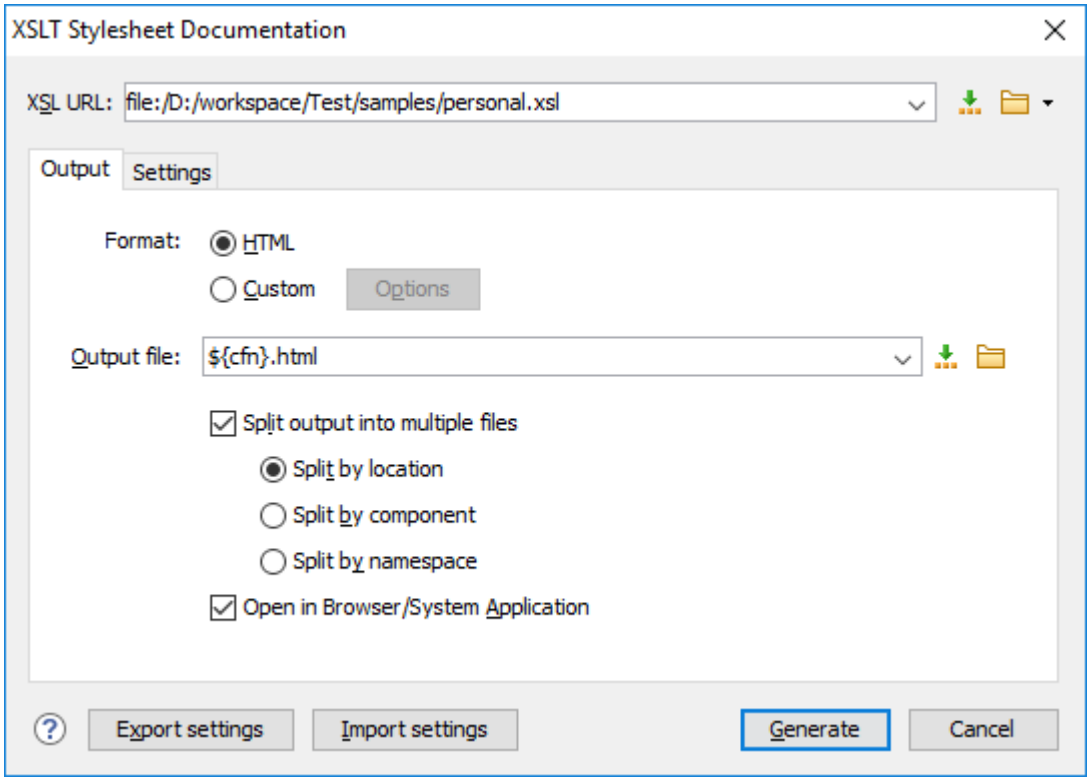
- If you are using a project *(on page 330)* in Oxygen XML Developer, create a `catalog.xml` file in the project directory. This catalog will then be loaded automatically.
- Edit *(on page 1207)* the **Run XSpec Test** transformation scenario, go to the **Parameters** tab *(on page 1143)*, and set the value of the `catalog` parameter to the location of your catalog file.

Generating Documentation for an XSLT Stylesheet

You can use Oxygen XML Developer 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 671)*, 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 334)*. You can also open the tool by using the **Generate Documentation** toolbar button.

Figure 182. 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:

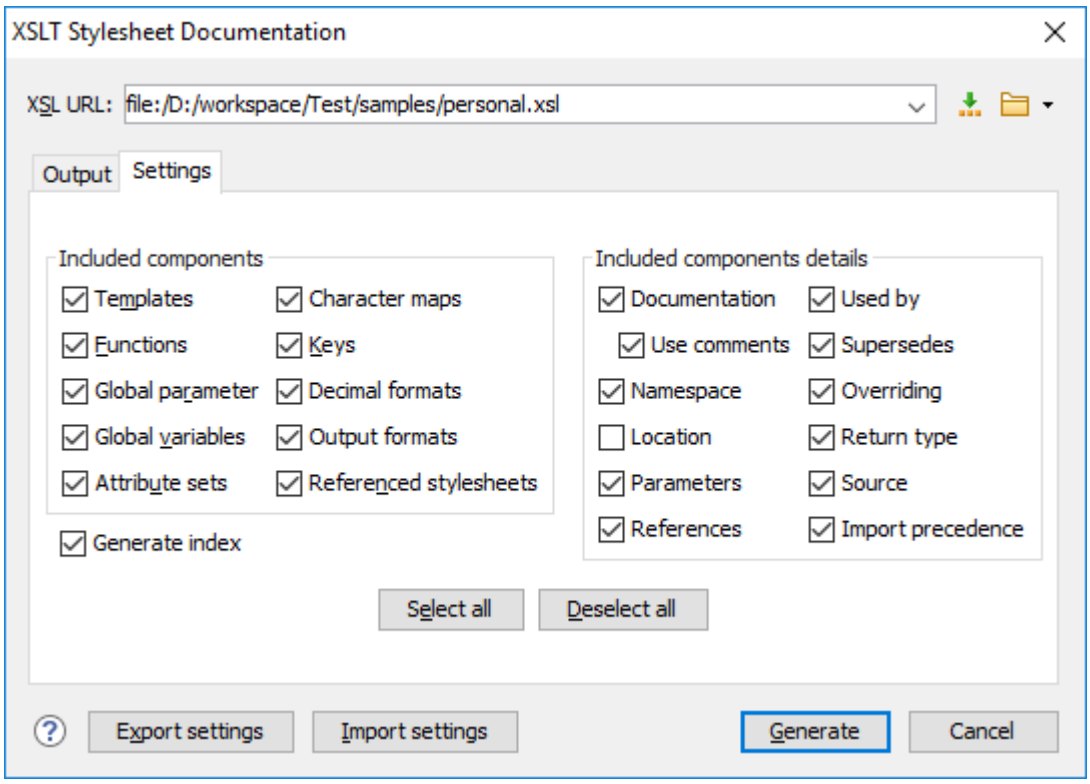
- **Format** - Allows you to choose between the following formats:
  - **HTML** - The documentation is generated in HTML output format *(on page 668)*.
  - **Custom** - The documentation is generated in a custom output format *(on page 671)*, allowing you to control the output. Click the **Options** button to open a **Custom format options** dialog box *(on page 672)* 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 79)*, 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 183. 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 Developer 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 2305)*. 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 671)* instead of the built-in HTML format *(on page 668)* 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.



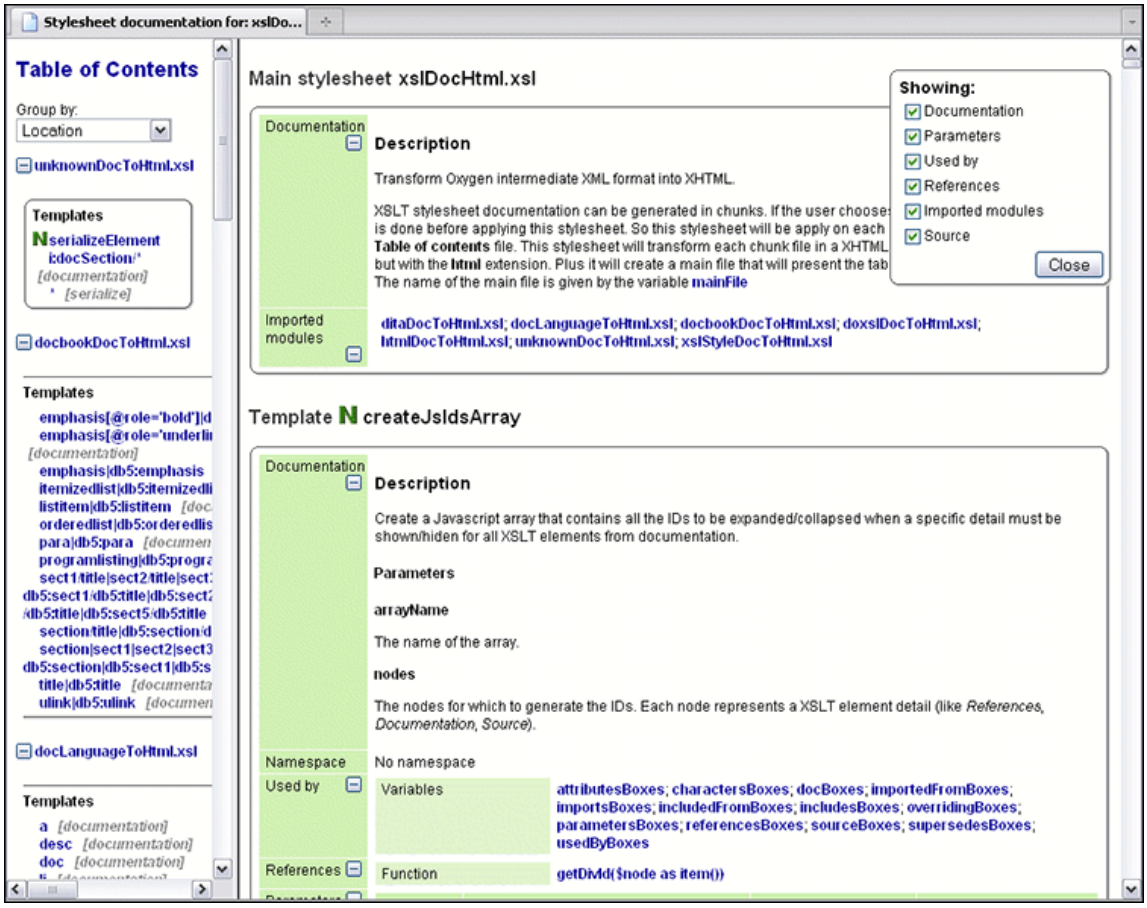
Tip:
This function can be executed from an automated command-line script, for more details, see Scripting Oxygen *(on page 2263)*.

Related Information:
XSLT Stylesheet Component Documentation Support *(on page 652)*

Generate XSLT Documentation in HTML Format

When using the XSLT Stylesheet Documentation dialog box *(on page 665)* to generate XSLT documentation in HTML format, it is presented in a visual diagram style with various sections, hyperlinks, and options.

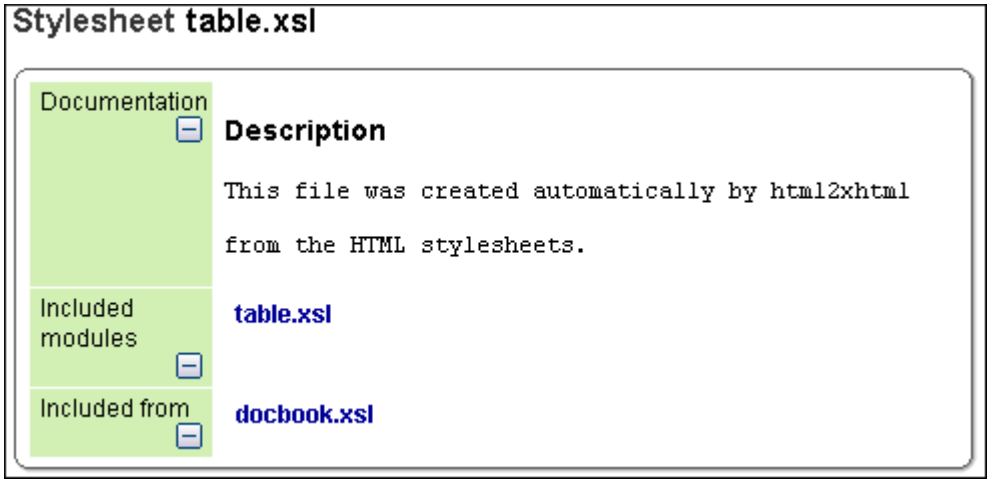
Figure 184. XSLT Stylesheet Documentation Example



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.

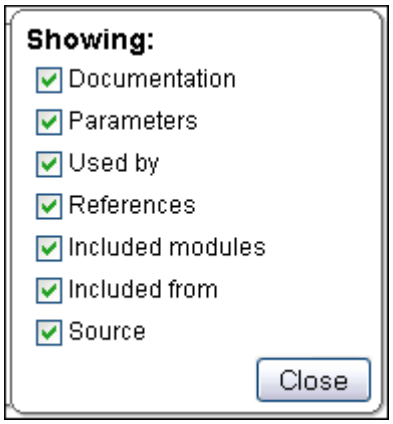
Figure 185. Information About an XSLT Stylesheet



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.

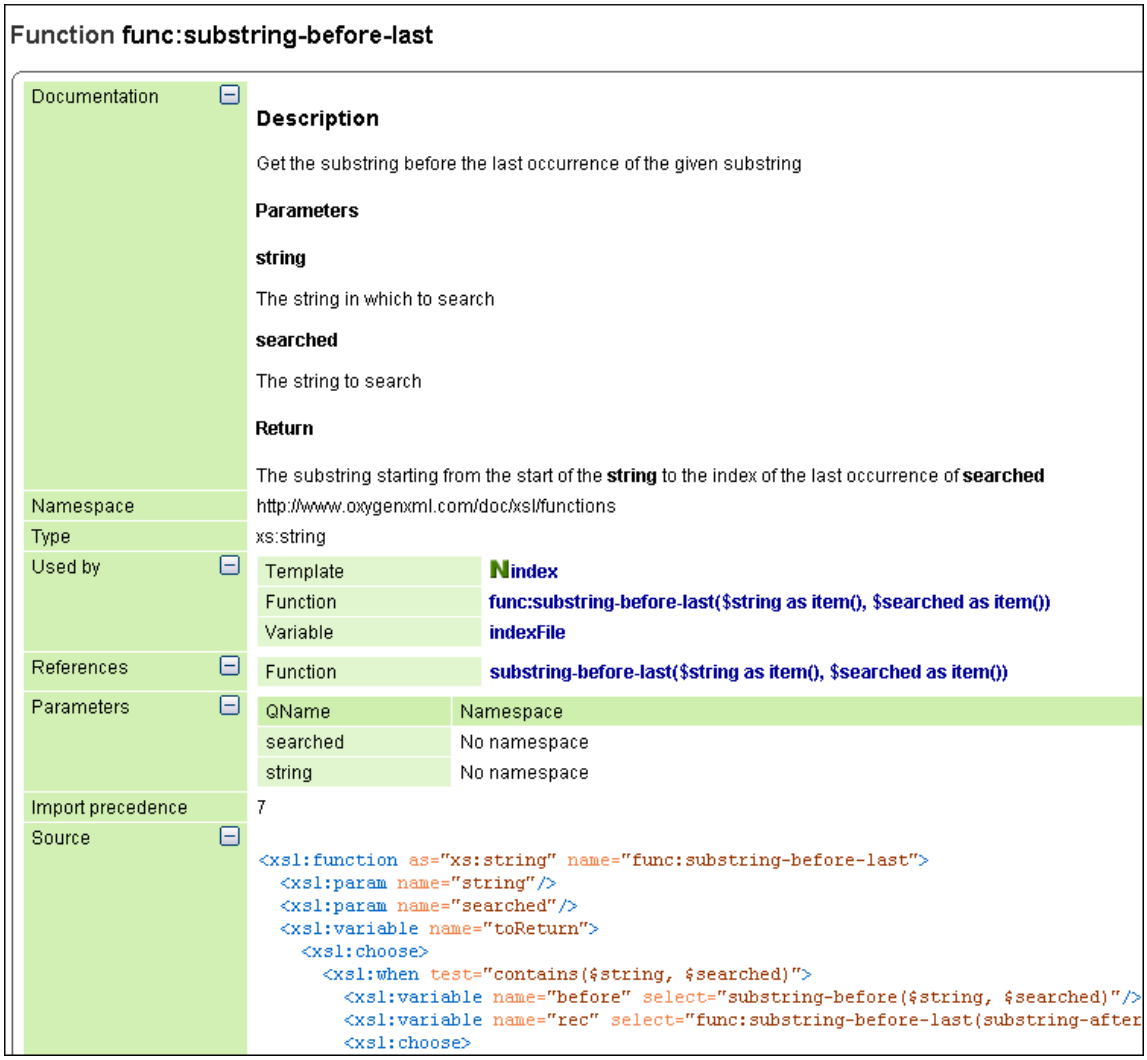
After the documentation is generated, you can collapse or expand details for some stylesheet XSLT elements by using the **Showing** options or the **Collapse** or **Expand** buttons.

Figure 186. Showing Options



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).

Figure 187. Documentation for an XSLT Element



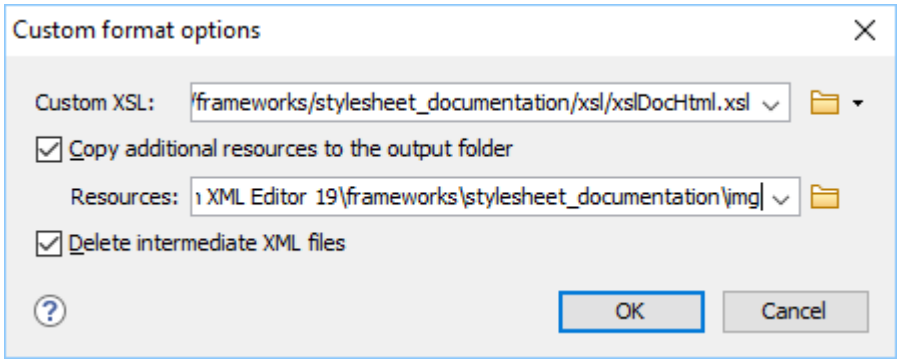
Generate XSLT Documentation in a Custom Format

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 665)*.
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**.

Figure 188. Custom Format Options Dialog Box



Compiling an XSL Stylesheet for Saxon

As of Saxon 12.9, 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 Developer 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 Developer** - You can also use a *stylesheet export file* (SEF) in Oxygen XML Developer 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 673)*. When configuring the XSLT transformation, you will specify the SEF file in the **XSL URL** field *(on page 1103)*.

Compiling an SEF File

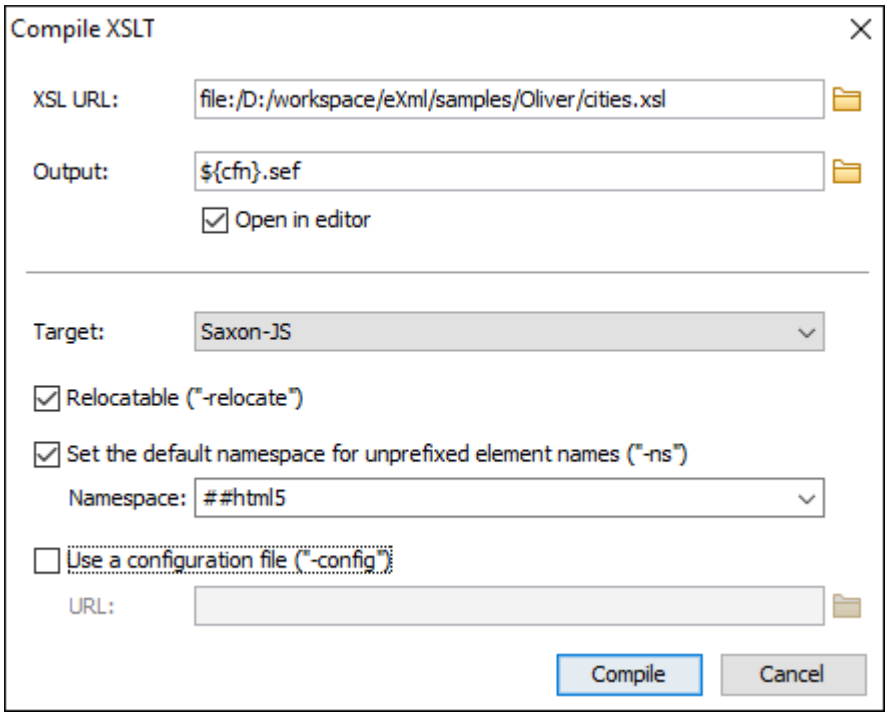
The **Compile XSL Stylesheet for Saxon** tool can be found in the **Tools** menu and it compiles a specified stylesheet as a JSON or an XML file (*stylesheet export file*).

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.

Figure 189. Compile XSLT Stylesheet for Saxon Dialog Box



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 Developer 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 12.9 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.

Attention:
Oxygen XML Developer does not support the `ALLOWED_PROTOCOLS` Saxon property (*http://saxon.sf.net/feature/allowedProtocols*). This feature is specific to Saxon and controls access by Saxon (but not by underlying software, such as the XML parser). For more information, see https://www.saxonica.com/documentation12/index.html#!configuration/config-features.

Compile

Use this button to generate the *stylesheet export file* according the options selected in this dialog box.

Convert Between JSON and XML from XSLT

Two XPath extension functions are available as built-in **Oxygen** functions for converting between JSON and XML formats from an XSLT file:

- **xs:string xml-to-json(xs:string jsonText)** - Converts the input `jsonText` to XML, using the JSON to XML conversion tool *(on page 873)* provided by Oxygen XML Developer.
- **xs:string json-to-xml(xs:string xmlText)** - Converts the input `xmlText` to JSON, using the XML to JSON conversion tool *(on page 876)* provided by Oxygen XML Developer.

> **Tip:**
> These functions are available in the list of content completion proposals when editing XPath expressions in XSLT 3.0 (or later).

The namespace of the functions is: `xmlns:oxy=http://www.oxygenxml.com/extension` (content completion is available for the `xmlns:oxy` prefix).

Examples of XSLT files that perform a conversion:

XML to JSON

```
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
                xmlns:oxy="http://www.oxygenxml.com/extension" version="3.0">


    <xsl:output method="text" indent="false" omit-xml-declaration="true"/>
    <xsl:template match="/">
        <xsl:value-of select="oxy:xml-to-json(serialize(node()))"/>
    </xsl:template>
</xsl:stylesheet>
```

JSON to XML

```
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
                xmlns:oxy="http://www.oxygenxml.com/extension" version="3.0">


    <xsl:template match=".">
        <xsl:variable name="xmlContent"
select="oxy:json-to-xml(serialize(., map { 'method' : 'json', 'indent' : true() }))"/>
        <xsl:value-of select="$xmlContent" disable-output-escaping="true"/>
    </xsl:template>
</xsl:stylesheet>
```

Editing Ant Build Files

Oxygen XML Developer 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 2308)* and *Quick Fix (on page 2308)* support, as well as numerous common editing and search features.

Related Information:
Editing XML Documents in Text Mode *(on page 442)*

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 Developer 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 cat set a main Ant build file either by using the *main files* support from the **Project** view *(on page 350)*, or a validation scenario *(on page 677)*.

To set a *main file* using a validation scenario, add validation units that point to the main modules. Oxygen XML Developer 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 2306)* include:

- Correct validation of a module in the context of a larger build structure.
- *Content Completion Assistant (on page 2303)* displays all components valid in the current context.
- The **Outline** view *(on page 680)* displays the components collected from the entire build file structure.

Validating Ant Build Files

Oxygen XML Developer 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 2303)* libraries. The path to these libraries can be configured in the **Ant** preferences page *(on page 197)*. The validation processor accesses the parameters set in the associated Ant transformation scenario *(on page 1143)* and uses them as Ant properties when validating the current build script.

Oxygen XML Developer 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 2306)* file that contains the task implementation to the library directory of the built-in Ant distribution that comes bundled with Oxygen XML Developer (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 1140)*.
2. Edit the transformation scenario and set all parameters *(on page 1143)* 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 1210)*). 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 Developer 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 1210)*:

- **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 2303)* version 1.10.9) that is included with Oxygen XML Developer, 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 Developer 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 1141)* 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 1207)*
Configure Transformation Scenario(s) Dialog Box *(on page 1210)*
Applying Associated Transformation Scenarios *(on page 1210)*
Ant Transformation *(on page 1140)*

Ant Quick Fix Support

The Oxygen XML Developer *Quick Fix* support *(on page 2308)* 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 Developer 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 + Option + 1 on macOS)** on your keyboard.

Oxygen XML Developer 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 (on page 2303)* 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** 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 2306)* file that contains the task implementation to the library directory of the built-in Ant distribution that comes bundled with Oxygen XML Developer (for example, `[OXYGEN_INSTALL_DIR]/tools/ant/lib` folder).

Note:
For Ant resources, the proposals are collected starting from the *main files (on page 2306)*. 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 350)*.

Related Information:
http://ant.apache.org/manual/properties.html

Syntax Highlighting in Ant Files

Oxygen XML Developer 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 79)*.
2. Go to **Editor > Syntax Highlight** *(on page 158)*.
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.

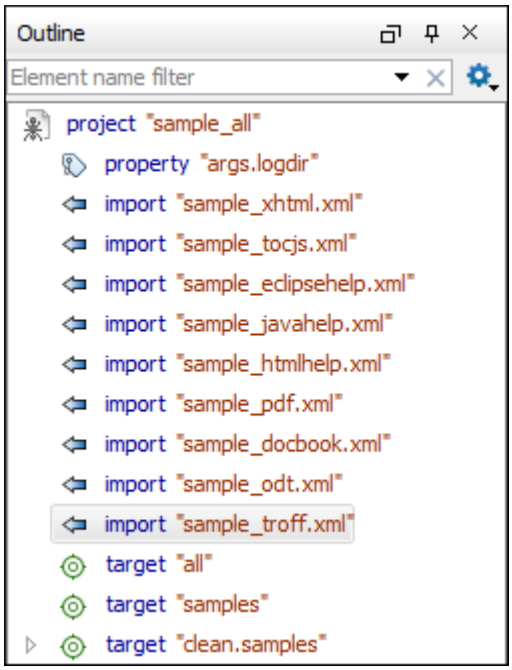
Related Information:
Syntax Highlight Preferences *(on page 158)*

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 2306)*. 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 350)*.

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 190. Ant 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. 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 2306)*. 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 235)*.

Configure displayed attributes

Displays the **XML Structured Outline** preferences page *(on page 235)*.

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 686)* 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 686)* for more details.

Component Dependencies

Opens the Ant **Component Dependencies** view *(on page 685)* 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 687)* 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 Developer 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 Developer sorts the components depending on their order in the document. Oxygen XML Developer also takes into account the name of the file that the components are part of.

Ant Referenced/Dependent Resources View

The **Referenced/Dependent Resources** 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 references or dependencies of a build file, select it in the **Project** view *(on page 334)* and choose **Show referenced resources** or **Show dependent resources** from the contextual menu.

The following actions are available on the toolbar of the **Referenced/Dependent Resources** view:

Refresh

Refreshes the resource structure.

Stop

Stops the computing.

Show hierarchy for

Computes the hierarchical structure of the references for a resource.

Show dependencies for

Computes the structure of the dependencies for a resource.

Configure dependencies search scope

Allows you to configure a scope to compute the dependencies. 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 **Referenced/Dependent Resources** view contains the following actions:

Open

Opens the resource. You can also double-click a resource within the hierarchical 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.

Show references resources

Shows the references for the selected resource.

Show dependent resources

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 hierarchical structure.

Collapse All

Collapses all children of the selected resource from the hierarchical structure.

Tip:
When a recursive reference is encountered in the 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 569)*

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 Developer 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 configured it, open the **Preferences** dialog box **(Options > Preferences)** *(on page 79)* 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 474)*.

Related Information:
Mark Occurrences Preferences *(on page 159)*

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 569)*

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:

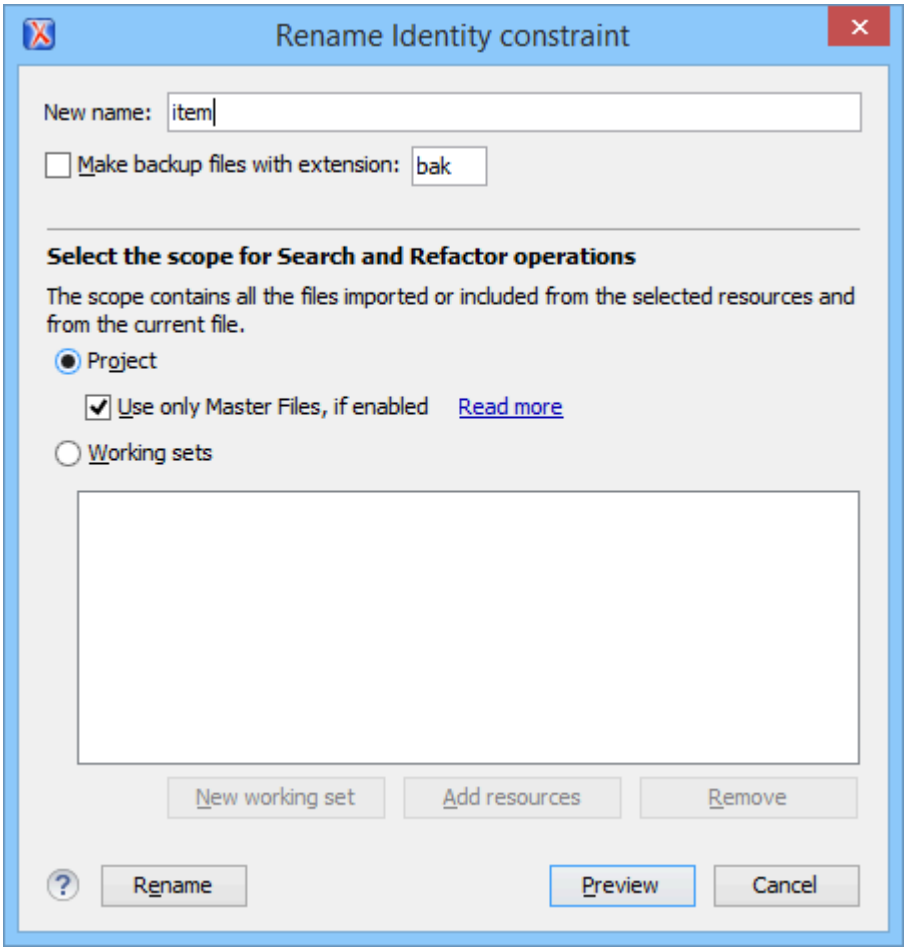
Rename Component

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 191. Rename Identity Constraint Dialog Box



Ant Quick Assist Support

The *Quick Assist* support *(on page 2308)* 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 + Option + 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 685)*

Ant Referenced/Dependent Resources View *(on page 683)*

Ant Refactoring Actions *(on page 687)*

Search and Refactor Operations Scope *(on page 569)*

Editing XML Schemas (XSD)

An XML Schema (XSD) 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 Developer 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 731)*** - Allows you to edit XML Schema files in a source editing mode.
- **Grid editing mode *(on page 279)*** - Displays XML Schema files in a structured spreadsheet-like grid.
- **Design editing mode *(on page 280)*** - Visual schema designer that helps you understand the structure and develop complex schemas.

For information about applying and detecting schemas, see Associating a Schema to XML Documents *(on page 554)*.

Resources

For more information about editing XML Schemas, see our video demonstration:

https://www.youtube.com/embed/vz1eIZELQgc

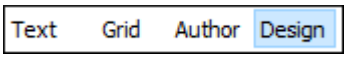
Related Information:

Associating a Schema to XML Documents *(on page 554)*

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 Developer 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 Developer 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 131)*. The same zoom factor is applied for the print and save actions.

Resources

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:

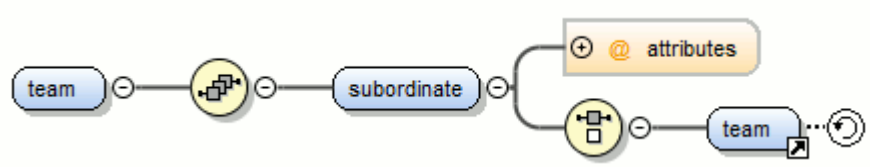
- Move/reference components in the diagram using drag-and-drop actions.
- Select consecutive components on the diagram (components from the same level) using the *Shift* key. You can also make discontinuous selections in the schema diagram using the **Ctrl (Meta on macOS)** key. To deselect one of the 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 735)*:
  - **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 364)* or the Quick find toolbar *(on page 378)*. 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 Developer.

> **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.

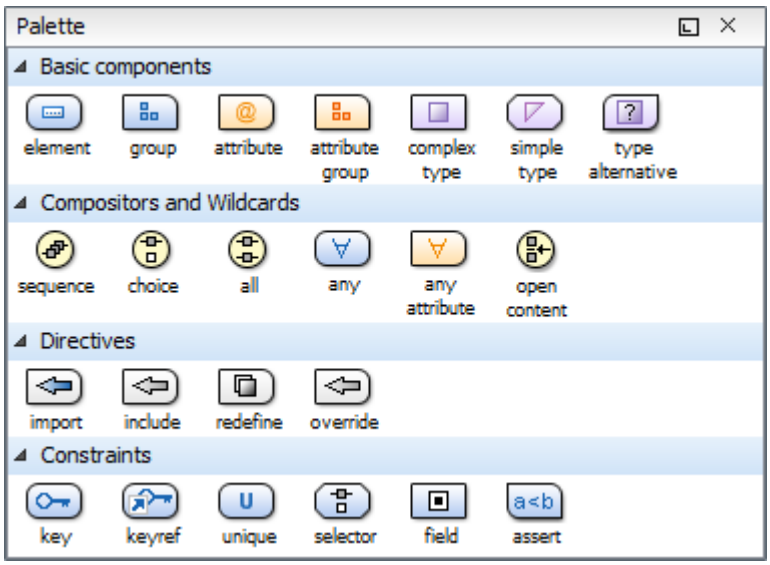
Figure 192. Recursive Reference



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 172)*. If the view is not displayed, it can be opened by selecting it from the **Window > Show View** menu.

Figure 193. Palette View



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 ⃠. 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 160)* (default color is red).

Resources

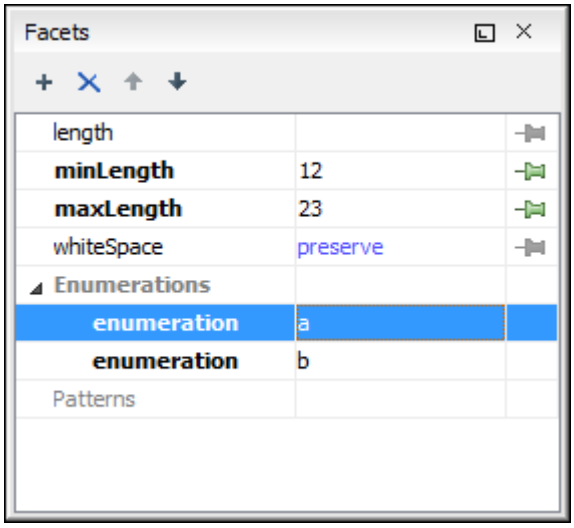
For more information about the Schema palette, watch our video demonstration:

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

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 194. 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 160)*.

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 769)* 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 735)* 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 132)*.

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 744)*
XML Schema Component Dependencies View *(on page 742)*
XML Schema Referenced/Dependent Resources View *(on page 739)*
Generating Sample XML Files *(on page 748)*
Schema Design Preferences *(on page 131)*

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 737)* and the **Facets** view *(on page 692)*. 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 737)* 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.

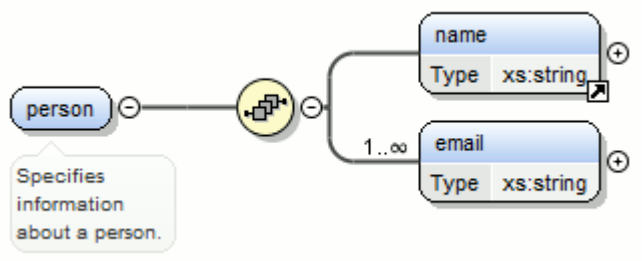
Change XML Schema Version

Use this action to change the XML Schema version of the current document.

Extract Global Element

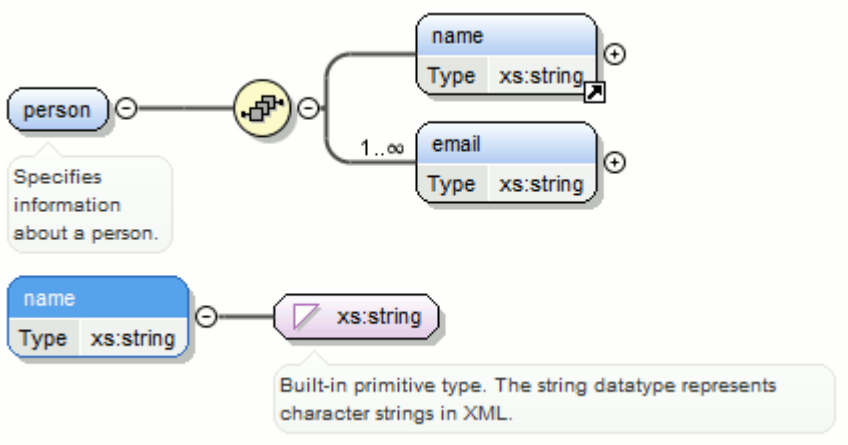
This action is available for local elements. A local element is made global and is replaced with a reference to the global element. The local element properties that are also valid for the global element declaration are kept.

Figure 195. Extracting a Global Element



If you use the **Extract Global Element** action on a `<name>` element, the result is:

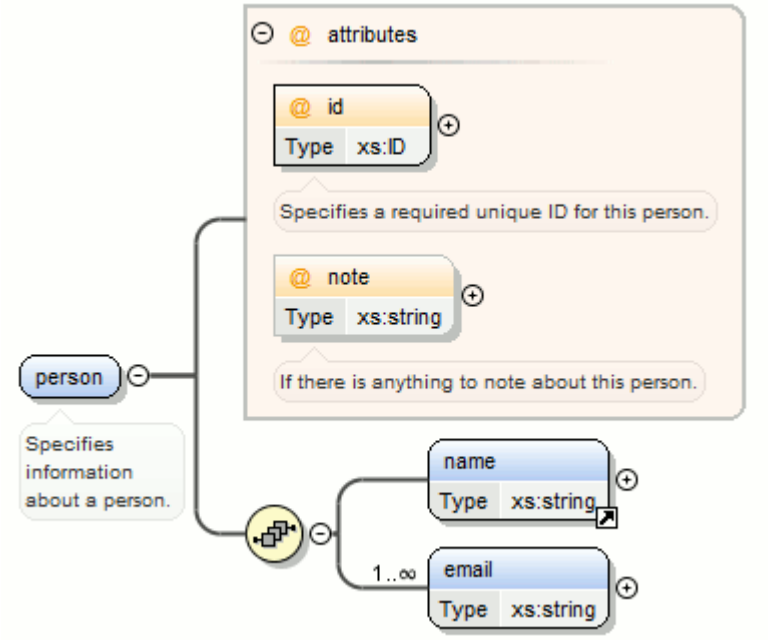
Figure 196. Extracting a Global Element on a `<name>` Element



Extract Global Attribute

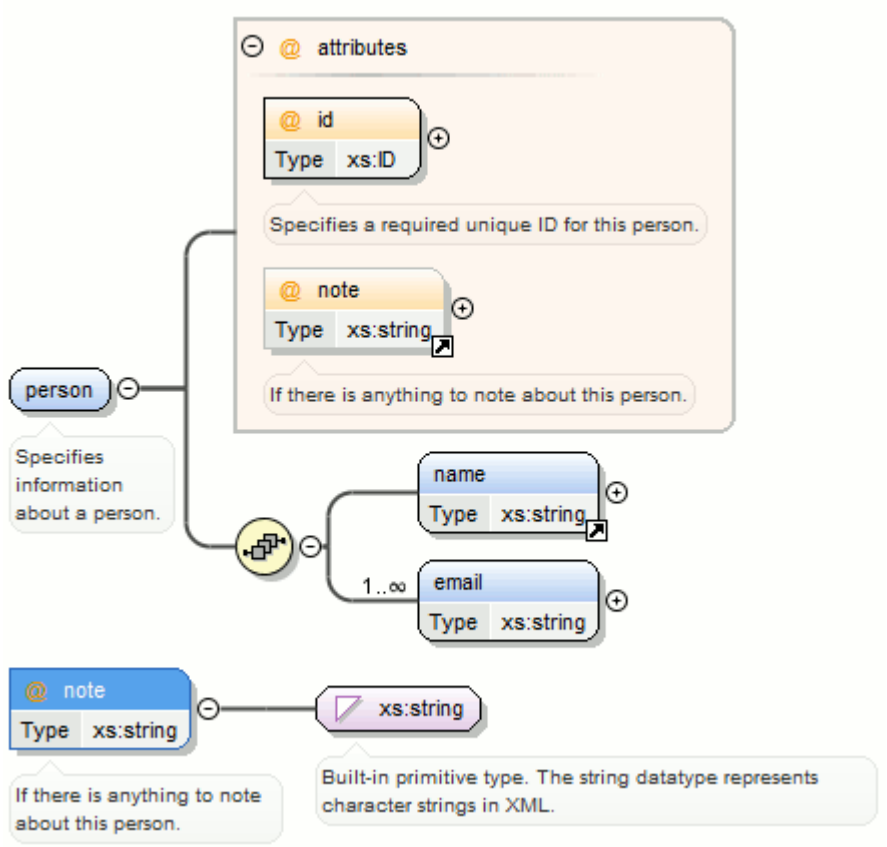
This action is available for local attributes. A local attribute is made global and replaced with a reference to the global attribute. The properties of local attribute that are also valid in the global attribute declaration are kept.

Figure 197. Extracting a Global Attribute



If you use the **Extract Global Attribute** action on a `@note` attribute, the result is:

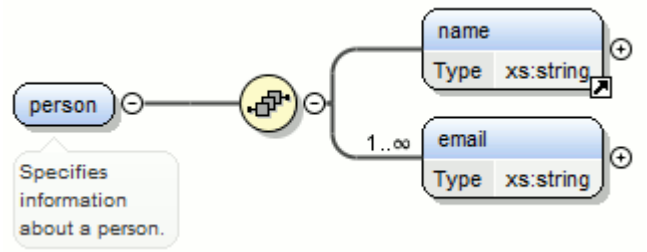
Figure 198. Extracting a Global Attribute on a `@note` Attribute



Extract Global Group

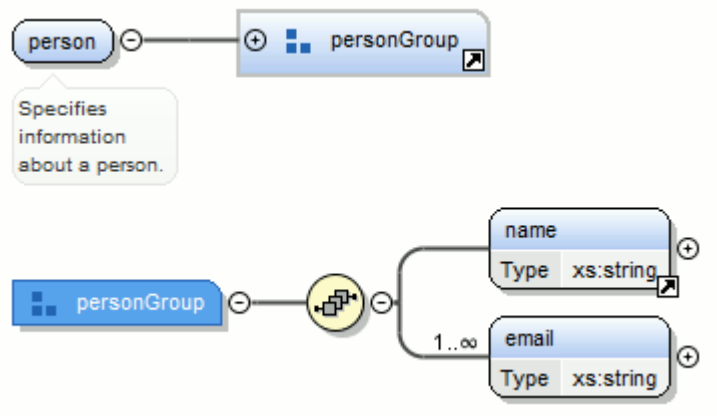
This action is available for compositors (sequence, choice, all). This action extracts a global group and makes a reference to it. The action is available only if the parent of the compositor is not a group.

Figure 199. 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:

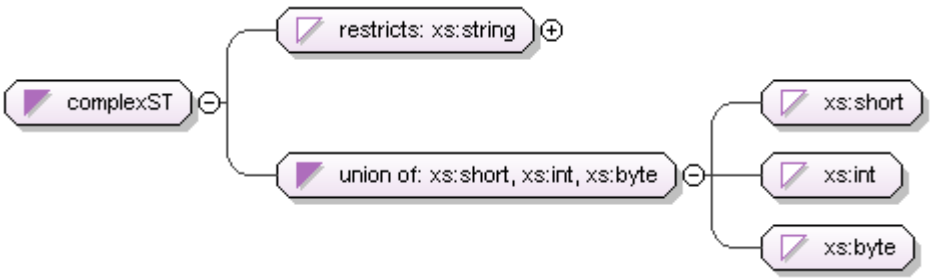
Figure 200. Extracting a Global Group on a `<sequence>` Element



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 201. 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 202. Extracting a Global Simple Type on a `union` Component

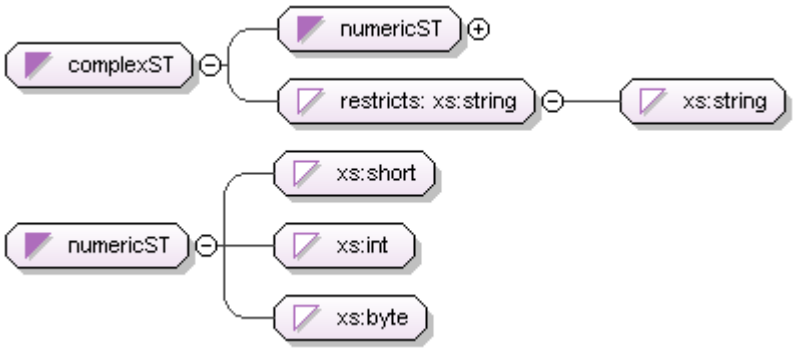
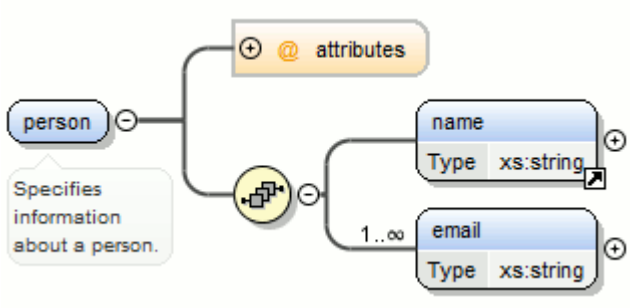
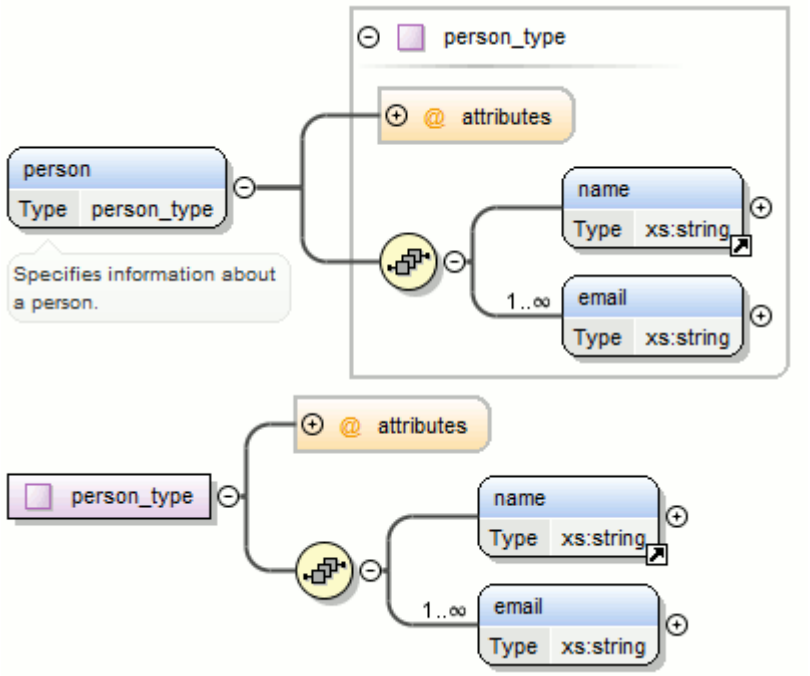


Figure 203. Extracting a Global Complex Type



If you use the action on a `<person>` element and choose `person_type` for the new complex type name, the result is:

Figure 204. Extracting a Global Complex Type on a `<person>` Element



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 742)* that allows you to see the dependencies for the currently selected component.

Show referenced resources

Opens the **Referenced/Dependent Resources** view *(on page 739)* that allows you to see the references for the currently selected resource.

Show dependent resources

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 748)* 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

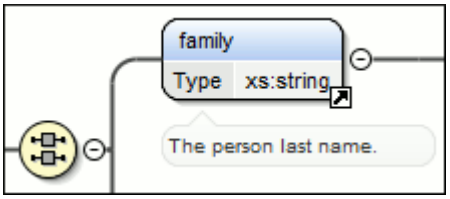
Opens the Schema preferences page *(on page 131)*.

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 205. Example: Required Component



- A thin line to identify a connection with an optional component (in the following image, `<email>` is an optional element).

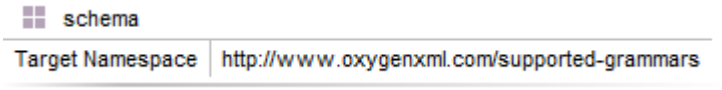
Figure 206. Example: Optional Component



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 207. 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 7. *xs:schema* Properties

Property Name	Description	Possible Values
Target Namespace	The schema target namespace	Any URI
Element Form Default	Determining whether or not local element declarations will be namespace-qualified by default	qualified, unqualified, [Empty] (default value is unqualified)
Attribute Form Default	Determining whether or not local attribute declarations will be namespace-qualified by default	qualified, unqualified, [Empty] (default value is unqualified)
Block Default	Default value of the *block* attribute of *xs:element* and *xs:complexType*	#all, extension, restriction, substitution, restriction extension, restriction substitution, extension substitution, restriction extension substitution, [Empty]
Final Default	Default value of the *final* attribute of *xs:element* and *xs:complexType*	#all, restriction, extension, restriction extension, [Empty]
Default Attributes	Specifies a set of attributes that apply to every complex Type in a schema document	Any
XPath Default Namespace	The default namespace used when the XPath expression is evaluated	##defaultNamespace, ##targetNamespace, ##local
Version	Schema version	Any token

Table 7. *xs:schema* Properties (continued)

Property Name	Description	Possible Values
ID	The schema ID	Any ID
Component	The edited component name	Not editable property
SystemID	The schema system ID	Not editable property

xs:element

Figure 208. 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.

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 8. *xs:element* Properties

Property Name	Description	Possible Values	Mentions
Name	The element name (always required)	Any NCName for global or local elements, any *QName (on page 2308)* for element references	If missing, will be displayed as '[element]' in diagram
Is Reference	When set, the local element is a reference to a global element	true/false	Appears only for local elements
Type	The element type	All declared or built-in types. In addition, the fol-	For all elements.

Table 8. *xs:element* Properties (continued)

Property Name	Description	Possible Values	Mentions
		lowing anonymous types are available: [ST-restriction], [ST-union], [ST-list], [CT-anonymous], [CT-extension SC], [CT-restriction SC], [CT-restriction CC], [CT-extension CC].	For references, the value is set in the referenced element.
Base Type	The extended/restricted base type	All declared or built-in types	For elements with complex type, with simple or complex content
Mixed	Defines if the complex type content model will be mixed	true/false	For elements with complex type
Content	The content of the complex type	simple/complex	For elements with complex type that extends/restricts a base type. It is automatically detected
Content Mixed	Defines if the complex content model will be mixed	true/false	For elements with complex type that has a complex content

Table 8. *xs:element* Properties (continued)

Property Name	Description	Possible Values	Mentions
Default	Default value of the element. A default value is automatically assigned to the element when no other value is specified	Any string	The fixed and default attributes are mutually exclusive
Fixed	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.	Any string	The fixed and default attributes are mutually exclusive
Min Occurs	Minimum number of occurrences of the element	A numeric positive value. Default value is 1	Only for references/local elements
Max Occurs	Maximum number of occurrences of the element	A numeric positive value (default value is 1)	Only for references/local elements
Substitution Group	Qualified name of the head of the substitution group that this element belongs to	All declared elements. For XML Schema 1.1 this property supports multiple values.	For global and reference elements
Abstract	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.	true/false	For global elements and element references
Form	Defines if the element is "qualified" (belongs to the target namespace) or "unqualified" (doesn't belong to any namespace)	unqualified/qualified	Only for local elements

Table 8. *xs:element* Properties (continued)

Property Name	Description	Possible Values	Mentions
Nillable	When this attribute is set to true, the element can be declared as nil using an *xsi:nil* attribute in the instance documents	true/false	For global elements and element references
Target Namespace	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.	Not editable property	For all elements
Block	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*.	#all, restriction, extension,substitution, extension restriction, extension substitution, restriction substitution, restriction extension substitution	For global elements and element references
Final	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*.	#all, restriction, extension, restriction extension, [Empty]	For global elements and element references
ID	The component ID	Any ID	For all elements
Component	The edited component name	Not editable property	For all elements
Namespace	The component namespace	Not editable property	For all elements
System ID	The component system ID	Not editable property	For all elements

xs:attribute

Figure 209. The *xs:attribute* Component



Defines an attribute. See more info at 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 9. *xs:attribute* Properties

Property Name	Description	Possible Value	Mentions
Name	Attribute name (always required)	Any NCName for global/local attributes, all declared attributes' *QName (on page 2308)* for references	For all local or global attributes. If missing, will be displayed as '[attribute]' in the diagram.
Is Reference	When set, the local attribute is a reference	true/false	For local attributes
Type	Qualified name of a simple type	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.	For all attributes. For references, the type is set to the referenced attribute.
Default	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.	Any string	For all local or global attributes. For references the value is from the referenced attribute.

Table 9. *xs:attribute* Properties (continued)

Property Name	Description	Possible Value	Mentions
Fixed	When specified, the value of the attribute is fixed and must be equal to this value. The default and fixed attributes are mutually exclusive.	Any string	For all local or global attributes. For references the value is from the referenced attribute.
Use	Possible usage of the attribute. Marking an attribute "prohibited" is useful to exclude attributes during derivations by restriction.	optional, required, prohibited	For local attributes
Form	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.	unqualified/qualified	For local attributes
Inheritable	Specifies if the attribute is inheritable. Inheritable attributes can be used by <alternative> element on descendant elements	true/false	For all local or global attributes. The default value is false. This property is available for XML Schema 1.1.
Target Namespace	Specifies the target namespace for local attribute declarations. The namespace URI may be different from the schema target namespace.	Any URI	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.
ID	The component ID	Any ID	For all attributes
Component	The edited component name	Not editable property	For all attributes
Namespace	The component namespace	Not editable property	For all attributes

Table 9. *xs:attribute* Properties (continued)

Property Name	Description	Possible Value	Mentions
System ID	The component system ID	Not editable property	For all attributes

xs:attributeGroup

Figure 210. The *xs:attributeGroup* Component



Defines an attribute group to be used in complex type definitions. See more info at http://www.w3.org/TR/xmlschema11-1/#element-attributeGroup.

Table 10. *xs:attributeGroup* Properties

Property Name	Description	Possible Values	Mentions
Name	Attribute group name (always required)	Any NCName for global attribute groups, all declared attribute groups for reference	For all global or referenced attribute groups. If missing, will be displayed as '[attributeGroup]' in diagram.
ID	The component ID	Any ID	For all attribute groups
Component	The edited component name	Not editable property	For all attribute groups
Namespace	The component namespace	Not editable property	For all attribute groups
System ID	The component system ID	Not editable property	For all attribute groups

xs:complexType

Figure 211. The *xs:complexType* Component



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 11. *xs:complexType* Properties

Property Name	Description	Possible Values	Mentions
Name	The name of the complex type (always required)	Any NCName	Only for global complex types. If missing, will be displayed as '[complexType]' in diagram.
Base Type Definition	The name of the extended/restricted types	Any from the declared simple or complex types	For complex types with simple or complex content
Derivation Method	The derivation method	restriction/ extension	Only when base type is set. If the base type is a simple type, the derivation method

Table 11. *xs:complexType* Properties (continued)

Property Name	Description	Possible Values	Mentions
			is always extension.
Content	The content of the complex type	simple/ complex	For complex types that extend/restrict a base type. It is automatically detected.
Content Mixed	Specifies if the complex content model will be mixed	true/ false	For complex contents
Mixed	Specifies if the complex type content model will be mixed	true/ false	For global and anonymous complex types
Abstract	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	true/ false	For global and anonymous complex types
Block	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 "unblock" them), which can also be blocked in the element definition. The default value is defined by the *blockDefault* attribute of *xs:schema*.	all, extension, restriction, extension restriction, [Empty]	For global complex types
Final	Controls whether the complex type can be further derived by extension or restriction to create new complex types	all, extension, restriction, extension restriction, [Empty]	For global complex types

Table 11. *xs:complexType* Properties (continued)

Property Name	Description	Possible Values	Mentions
Default Attributes Apply	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.	true/ false	This property is available only for XML Schema 1.1
ID	The component ID	Any ID	For all complex types
Component	The edited component name	Not editable property	For all complex types
Namespace	The component namespace	Not editable property	For all complex types
System ID	The component system ID	Not editable property	For all complex types

xs:simpleType

Figure 212. The *xs:simpleType* Component



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 12. *xs:simpleType* Properties

Name	Description	Possible Values	Scope
Name	Simple type name. Always required.	Any NCName	Only for global simple types. If missing, will be displayed as '[simpleType]' in diagram.
Derivation	A simple type category	restriction, list, or union	For all simple types
Base Type	A simple type definition component. Required if derivation method is set to restriction.	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.	For global and anonymous simple types with the derivation method set to restriction
Item Type	A simple type definition component. Required if derivation method is set to list.	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.	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).
Member Types	Category for grouping union members	Not editable property	For global and anonymous simple types with the derivation method set to union
Member	A simple type definition component. Re-	All global simple types and built-in simple types(from schema for schema). In addition another 3 proposals are present: [anonymous re-	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-

Table 12. *xs:simpleType* Properties (continued)

Name	Description	Possible Values	Scope
	quired if derivation method is set to union.	striction], [anonymous list], [anonymous union] for easily create anonymous simple types.	erence (through the memberTypes attribute) or embedded as simple datatype local definitions in the xs:union element. Both styles can be mixed.
Final	Blocks any further derivations of this datatype (by list, union, derivation or all)	#all, list, restriction, union, list restriction, list union, restriction union. In addition, [Empty] proposal is present for set empty string as value.	Only for global simple types
ID	The component ID	Any ID	For all simple types
Component	The name of the edited component	Not editable property	Only for global and local simple types
Namespace	The component namespace	Not editable property	For global simple types
System ID	The component system ID	Not editable property	Not present for built-in simple types

xs:alternative

The *type alternatives* mechanism allows you to specify type substitutions on an element declaration.



Note:
xs:alternative is available for XML Schema 1.1.

Figure 213. The *xs:alternative* Component

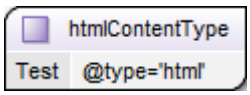


Table 13. *xs:alternative* Properties

Name	Description	Possible Values
Type	Specifies type substitutions for an element, depending on the value of the attributes	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]
Test	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.	An XPath expression
XPath Default Namespace	The default namespace used when the XPath expression is evaluated	##defaultNamespace, ##targetNamespace, ##local
ID	Specifies the component ID	Any ID
Component	Specifies the type of XML schema component	Not editable property
System ID	Points to the document location of the schema	Not editable property

xs:group

Figure 214. The *xs:group* Component



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 14. *xs:group* Properties

Property Name	Description	Possible Values	Mentions
Name	The group name (always required)	Any NCName for global groups, all declared groups for reference	If missing, will be displayed as '[group]' in diagram
Min Occurs	Minimum number of occurrences of the group	A numeric positive value. Default value is 1	Appears only for reference groups
Max Occurs	Maximum number of occurrences of the group	A numeric positive value. Default value is 1	Appears only for reference groups
ID	The component ID	Any ID	For all groups
Component	The edited component name	Not editable property	For all groups
Namespace	The component namespace	Not editable property	For all groups
System ID	The component system ID	Not editable property	For all groups

xs:include

Figure 215. 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 15. *xs:include* properties

Property Name	Description	Possible Values
Schema Location	Included schema location	Any URI
ID	Include ID	Any ID
Component	The component name	Not editable property

xs:import

Figure 216. 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.

Table 16. *xs:import* Properties

Property Name	Description	Possible Values
Schema Location	Imported schema location	Any URI
Namespace	Imported schema namespace	Any URI
ID	Import ID	Any ID
Component	The component name	Not editable property

xs:redefine

Figure 217. 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.

Table 17. *xs:redefine* Properties

Property Name	Description	Possible Values
Schema Location	Redefine schema location	Any URI
ID	Redefine ID	Any ID
Component	The component name	Not editable property

xs:override

Figure 218. 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.

Table 18. *xs:override* Properties

Property Name	Description	Possible Values
Schema Location	Redefine schema location	Any URI
ID	Redefine ID	Any ID

xs:notation

Figure 219. The *xs:notation* Component



Describes the format of non-XML data within an XML document. See more info at http://www.w3.org/TR/xmlschema11-1/#element-notation.

Table 19. *xs:notation* Properties

Property Name	Description	Possible values	Mentions
Name	The notation name (always required)	Any NCName	If missing, will be displayed as '[notation]' in diagram
System Identifier	The notation system identifier	Any URI	Required if public identifier is absent (otherwise, optional)
Public Identifier	The notation public identifier	A Public ID value	Required if system identifier is absent (otherwise, optional)
ID	The component ID	Any ID	For all notations
Component	The edited component name	Not editable property	For all notations
Namespace	The component namespace	Not editable property	For all notations
System ID	The component system ID	Not editable property	For all notations

xs:sequence / xs:choice / xs:all

Figure 220. *xs:sequence*



xs:sequence specifies that the child elements must appear in a sequence. Each child element occurs once by default. See more info at http://www.w3.org/TR/xmlschema11-1/#element-sequence.

Figure 221. *xs:choice*



xs:choice allows only one of the elements contained in the declaration to be present within the containing element. See more info at http://www.w3.org/TR/xmlschema11-1/#element-choice.

Figure 222. *xs:all*



xs:all specifies that the child elements can appear in any order. See more info at http://www.w3.org/TR/xmlschema11-1/#element-all.

The compositor 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 compositor are drawn using dotted lines if the compositor is optional.

Table 20. *xs:sequence*, *xs:choice*, *xs:all* Properties

Property Name	Description	Possible Values	Mentions
Compositor	Compositor type	sequence, choice, all	'all' is only available as a child of a group or complex type
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
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
ID	The component ID	Any ID	For all compositors
Component	The edited component name	Not editable property	For all compositors
System ID	The component system ID	Not editable property	For all compositors

xs:any

Figure 223. The *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.

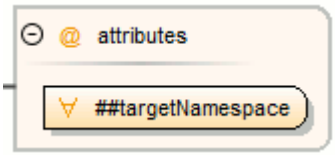
The graphical representation also contains the value for the *minOccurs* and *maxOccurs* properties (for 0..1 and 1..1 occurs the values are implied by the connector style) and the connectors to the wildcard are drawn using dotted lines if the wildcard is optional.

Table 21. *xs:any* Properties

Property Name	Description	Possible Values
Namespace	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
notNamespace	Specifies the namespace that extension elements or attributes cannot come from	##local, ##targetNamespace
notQName	Specifies an element or attribute that is not allowed	##defined
Process Contents	Type of validation required on the elements allowed for this wildcard	skip, lax, strict
Min Occurs	Minimum occurrences of any	A numeric positive value. Default is 1
Max Occurs	Maximum occurrences of any	A numeric positive value. Default is 1
ID	The component ID	Any ID
Component	The name of the edited component	Not editable property
System ID	The component system ID	Not editable property

xs:anyAttribute

Figure 224. The *xs:anyAttribute* Component



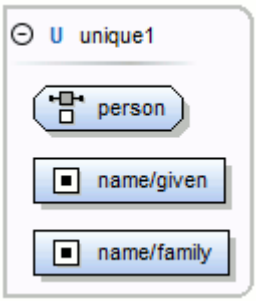
Enables the author to extend the XML document with attributes not specified by the schema. See more info at http://www.w3.org/TR/xmlschema11-1/#element-anyAttribute.

Table 22. *xs:anyAttribute* Properties

Property Name	Description	Possible Value
Namespace	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
Process Contents	Type of validation required on the elements allowed for this wildcard	skip, lax, strict
ID	The component ID	Any ID
Component	The name of the edited component	Not editable property
System ID	The component system ID	Not editable property

xs:unique

Figure 225. The *xs:unique* Component



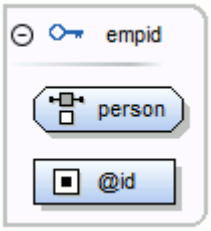
Defines that an element or an attribute value must be unique within the scope. See more info at http://www.w3.org/TR/xmlschema11-1/#element-unique.

Table 23. *xs:unique* Properties

Property Name	Description	Possible Values
Name	The unique name (always required)	Any NCName
ID	The component ID	Any ID
Component	The edited component name	Not editable property
Namespace	The component namespace	Not editable property
System ID	The component system ID	Not editable property

xs:key

Figure 226. The *xs:key* Component



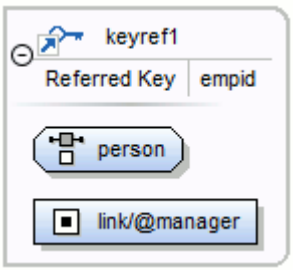
Specifies an attribute or element value as a key (unique, non-nullable and always present) within the containing element in an instance document. See more info at http://www.w3.org/TR/xmlschema11-1/#element-key.

Table 24. *xs:key* Properties

Property Name	Description	Possible Value
Name	The key name (always required)	Any NCName
ID	The component ID	Any ID
Component	The edited component name	Not editable property
Namespace	The component namespace	Not editable property
System ID	The component system ID	Not editable property

xs:keyRef

Figure 227. 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.

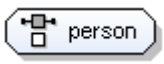
A keyref by default displays the *Referenced Key* property when rendered.

Table 25. *xs:keyRef* Properties

Property Name	Description	Possible Values
Name	The keyref name (always required)	Any NCName
Referenced Key	The name of referenced key	Any declared element constraints
ID	The component ID	Any ID
Component	The edited component name	Not editable property
Namespace	The component namespace	Not editable property
System ID	The component system ID	Not editable property

xs:selector

Figure 228. 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.

Table 26. *xs:selector* Properties

Property Name	Description	Possible Values
XPath	Relative XPath expression identifying the element that the constraint applies to	An XPath expression
ID	The component ID	Any ID

Table 26. *xs:selector* Properties (continued)

Property Name	Description	Possible Values
Component	The edited component name	Not editable property
System ID	The component system ID	Not editable property

xs:field

Figure 229. 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 27. *xs:field* Properties

Property Name	Description	Possible Values
XPath	Relative XPath expression identifying the field(s) composing the key, key reference, or unique constraint	An XPath expression
ID	The component ID	Any ID
Component	The edited component name	Not editable property
System ID	The component system ID	Not editable property

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.

Figure 230. The *xs:assert* Component

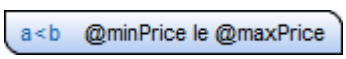


Table 28. *xs:assert* Properties

Property Name	Description	Possible Values
Test	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.	An XPath expression
XPath Default Namespace	The default namespace used when the XPath expression is evaluated	##default-Namespace, ##target-Namespace, ##local
ID	Specifies the component ID	Any ID
Component	The edited component name	Not editable property
System ID	The component system ID	Not editable property

xs:openContent

Figure 231. 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.

Table 29. *xs:openContent* Properties

Property Name	Description	Possible Value
Mode	Specifies where the extension elements can be inserted	The value can be: "interleave", "suffix" or "none". The default value is "interleave".
ID	The component ID	Any ID
Component	The edited component name	Not editable property
System ID	The component system ID	Not editable property

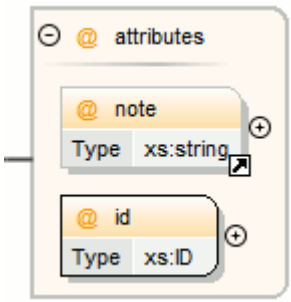
Note:
This component is available for XML Schema 1.1 only. To change the version of the XML Schema, open the **Preferences** dialog box **(Options > Preferences)** *(on page 79)* and go to **XML > XML Parser > XML Schema**.

Constructs Used to Group Schema Components

This section explains the components that can be used for grouping other schema components.

Attributes

Figure 232. Attributes Construct



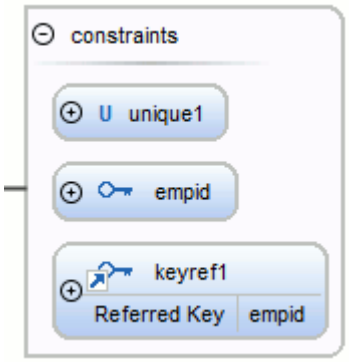
Groups all attributes and attribute groups belonging to a complex type.

Table 30. *attributes* Properties

Property Name	Description	Possible Values
Component	The element that has the attributes displayed	Not editable property
System ID	The component system ID	Not editable property

Constraints

Figure 233. Constraints Construct



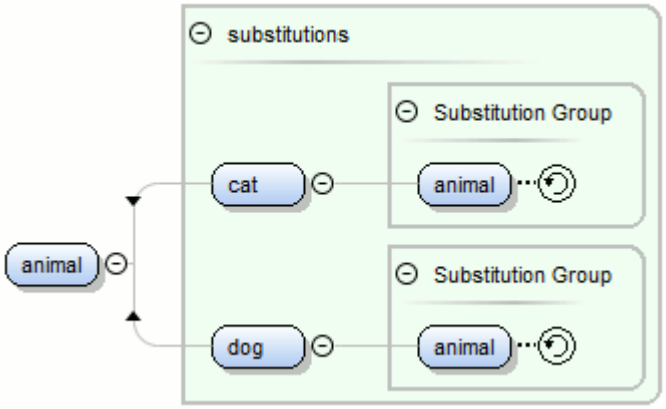
Groups all constraints (*xs:key (on page 724)*, *xs:keyRef (on page 725)*, or *xs:unique (on page 723)*) belonging to an element.

Table 31. *constraints* Properties

Property Name	Description	Possible Values
Component	The element that has the constraints displayed	Not editable property
System ID	The component system ID	Not editable property

Substitutions

Figure 234. Substitutions Construct



Groups all elements that can substitute the current element.

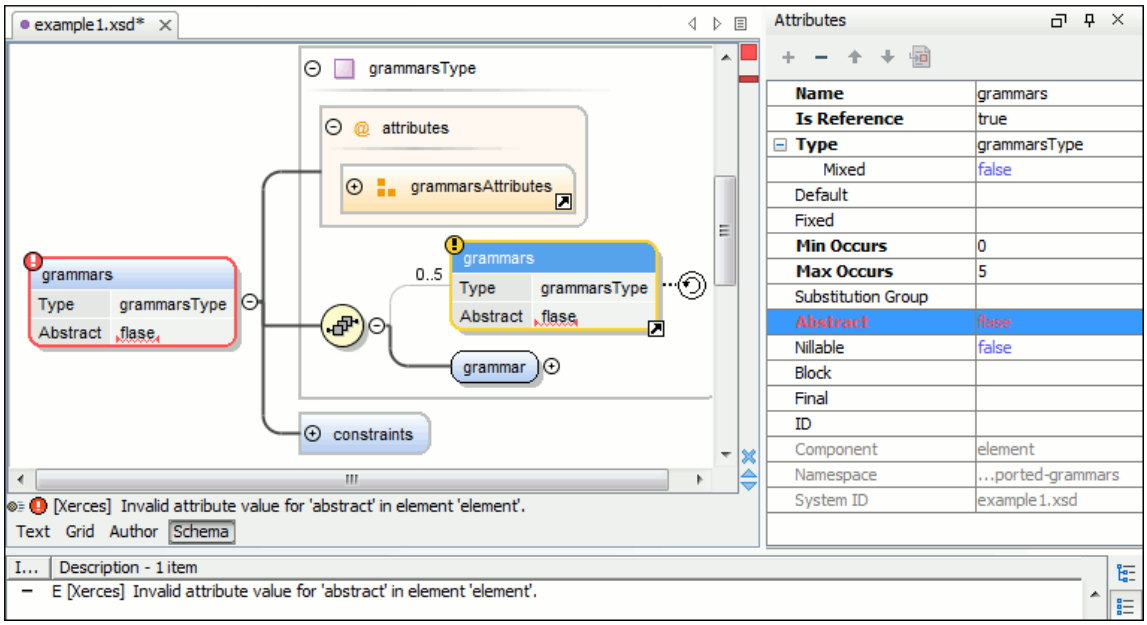
Table 32. *substitutions* Properties

Property Name	Description	Possible Values
Component	The element that has the substitutions displayed	Not editable property
System ID	The component system ID	Not editable property

Schema Validation

Validation for the **Design** mode is seamlessly integrated in the Oxygen XML Developer XML documents validation *(on page 514)* capability.

Figure 235. XML Schema Validation



A schema validation error is presented by highlighting the invalid component:

- In the Attributes View *(on page 737)*.
- 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 ( ) below the diagram view.

Invalid facets for a component are highlighted in the Facets View *(on page 692)*.

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 160)*. 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 564)* 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 Developer **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 744)*. The Outline view has two display modes: the standard outline *(on page 465)* mode and the components *(on page 735)* 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 129)* in the **Diagram** preferences page *(on page 128)*.

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 Developer 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 350)*, 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 Developer 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 (on page 2306)* include:

- Correct validation of a module in the context of a larger schema structure.
- *Content Completion Assistant (on page 2303)* 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 *(on page 735)* 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)** *(on page 79)*, go to **Editor > Document Checking**, and deselect the **Enable automatic validation** option *(on page 160)*.

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 Developer 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 *(on page 474)* 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 Developer window. Otherwise, for large values of the `@minOccurs` and `@maxOccurs` attributes, the validator fails with an **OutOfMemory** error that might make Oxygen XML Developer 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 *(on page 564)* 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 *(on page 516)*
Embedding Schematron Rules in XML Schema or RELAX NG *(on page 967)*
Validation Scenario *(on page 525)*
Associating a Schema to XML Documents *(on page 554)*
Presenting Validation Errors in Text Mode *(on page 519)*

Quick Fixes for DTD, XSD, and Relax NG Errors

Oxygen XML Developer offers *Quick Fixes (on page 2308)* 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 .

Oxygen XML Developer provides *Quick Fixes* for numerous types of problems, including the following:

Problem Type	Available Quick Fixes
A specific element is required in the current context	Insert the required element
An element is invalid in the current context	Remove the invalid element
The content of the element should be empty	Remove the element content
An element is not allowed to have child elements	Remove all child elements
Text is not allowed in the current element	Remove the text content
A required attribute is missing	Insert the required attribute
An attribute is not allowed to be set for the current element	Remove the attribute
The attribute value is invalid	Propose the correct attribute values
ID value is already defined	Generate a unique ID value
References to an invalid ID	Change the reference to an already defined ID

Related Information:
Schematron Quick Fixes (SQF) *(on page 553)*
Ignoring/Unignoring Validation Problems *(on page 549)*

Content Completion in XML Schema

The intelligent *Content Completion Assistant (on page 2303)* 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 79)*, go to **Editor > Content Completion**, and deselect the **Enable content completion** option *(on page 144)*.

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** 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 772)*, Oxygen XML Developer populates the proposals list with information taken either from XML Schema 1.0 or 1.1.

Oxygen XML Developer 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 148)* preferences page.

Syntax Highlighting in XML Schema

Oxygen XML Developer 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 79)*.
2. Go to **Editor > Syntax Highlight** *(on page 158)*.
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 Developer 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 159)*.

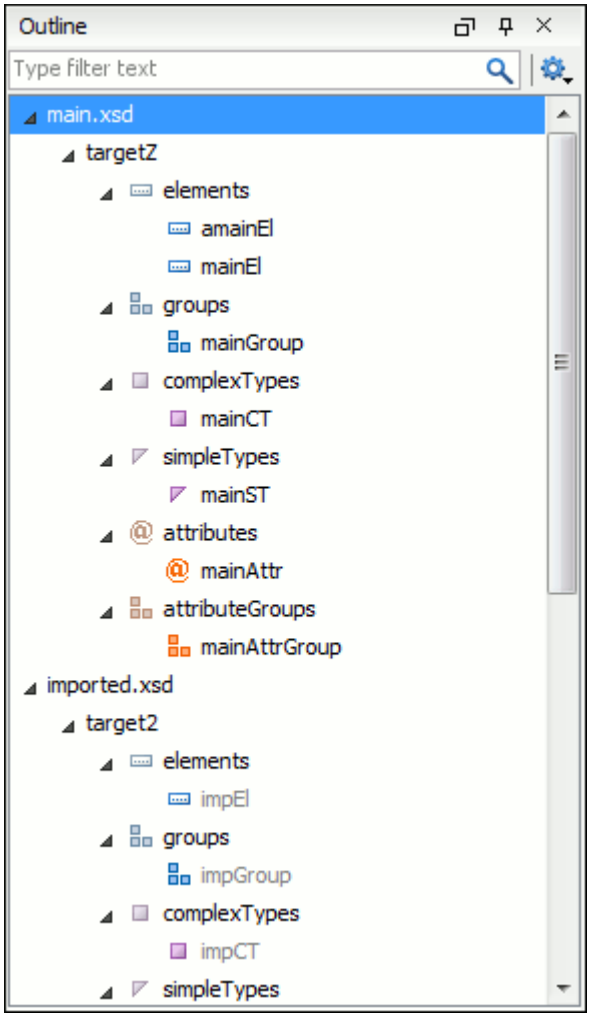
Related Information:

Customize Syntax Highlight colors *(on page 158)*

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 236. 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 2306)*. 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 2306)*) 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 742)* that allows you to see the dependencies for the currently selected component.

Show referenced resources (F4)

Opens the **Referenced/Dependent Resources** view *(on page 570)* that allows you to see the references for the currently selected resource.

Show dependent resources (Shift + F4)

Opens the **Referenced/Dependent Resources** view *(on page 570)* 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.

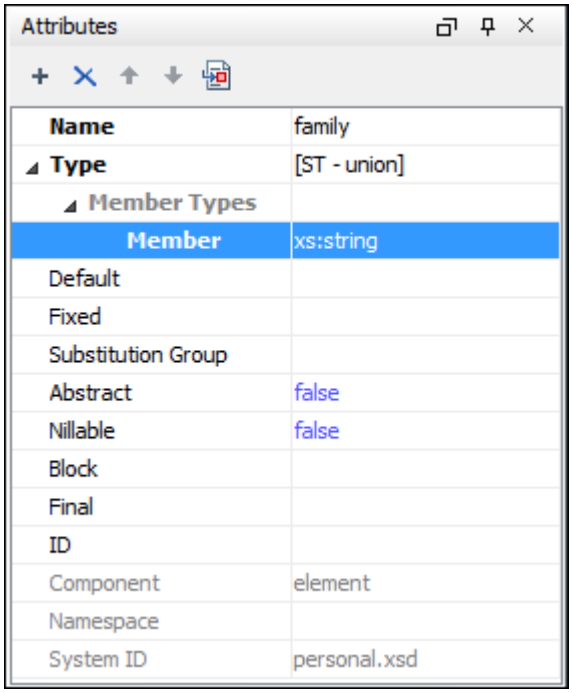
The content of the **Outline** view and the editing area are synchronized. When you select a component in the **Outline** view, its definition is highlighted in the editing area.

Related Information:
Searching and Refactoring Actions in XML Schemas *(on page 744)*
XML Schema Component Dependencies View *(on page 742)*
XML Schema Referenced/Dependent Resources View *(on page 739)*
Generating Sample XML Files *(on page 748)*
Modular Contextual Relax NG Schema Editing Using 'Main Files' Support *(on page 824)*

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.

Figure 237. Attributes View



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 160)*.

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 731)* 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.

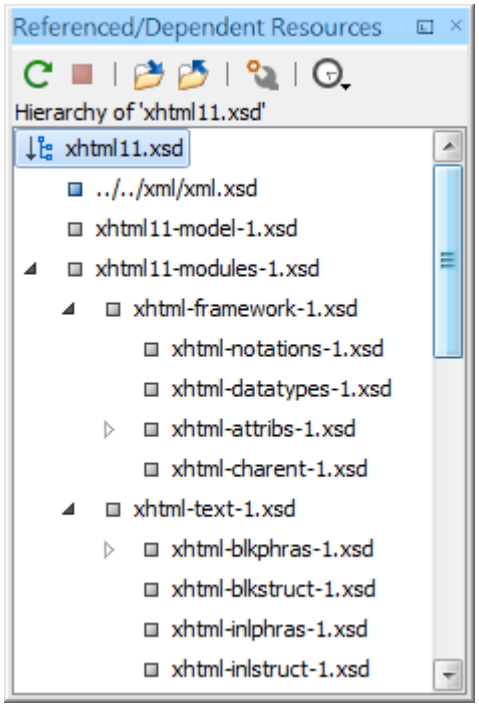
XML Schema Referenced/Dependent Resources View

The **Referenced/Dependent Resources** 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 **Referenced/Dependent Resources** 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 references or dependencies of an XML schema, select the desired schema in the **Project** view *(on page 334)* and choose **Show referenced resources** or **Show dependent resources** from the contextual menu.

Figure 238. Referenced/Dependent Resources View



The following actions are available on the toolbar of the **Referenced/Dependent Resources** view:

Refresh

Refreshes the resource structure.

Stop

Stops the computing.

Show hierarchy for

Computes the hierarchical structure of the references for a resource.

Show dependencies for

Computes the structure of the dependencies for a resource.

Configure dependencies search scope

Allows you to configure a scope to compute the dependencies. 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 **Referenced/Dependent Resources** view contains the following actions:

Open

Opens the resource. You can also double-click a resource within the hierarchical 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.

Show references resources

Shows the references for the selected resource.

Show dependent resources

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 hierarchical structure.

Collapse All

Collapses all children of the selected resource from the hierarchical structure.

Tip:
When a recursive reference is encountered in the 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 741)*.

Related Information:
Modular Contextual XML Editing Using 'Main Files' Support *(on page 567)*
Search and Refactor Operations Scope *(on page 569)*

Moving/Renaming XML Schema Resources

You can move and rename a resource presented in the **Referenced/Dependent Resources** 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 **Referenced/Dependent Resources** 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 **Referenced/Dependent Resources** 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.

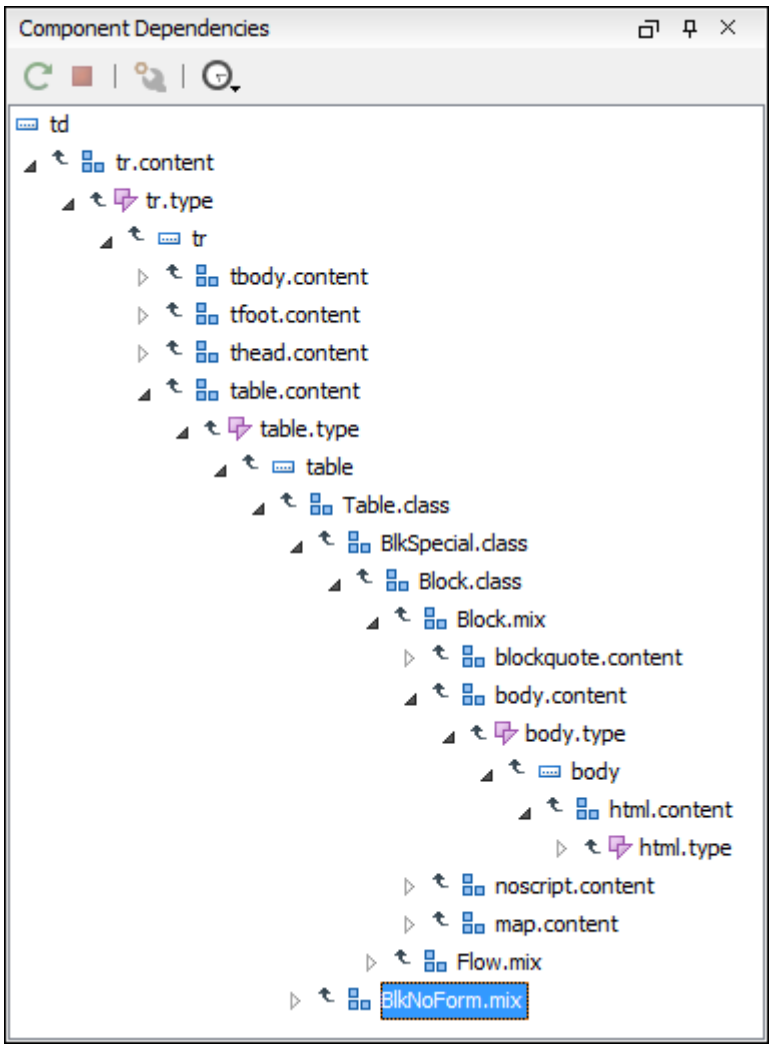
To see the dependencies of an XML Schema component:

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 .

Figure 239. Component Dependencies View



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.

Resources

For more information, see the **Maintain Complex XML Schemas** section of our **Developing XML Schemas** video demonstration:

https://www.youtube.com/embed/vz1eIZELQgc

Related Information:
Search and Refactor Operations Scope *(on page 569)*

Highlight Component Occurrences

When a component (for example types, elements, attributes) is found at current cursor position, Oxygen XML Developer 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 79)* 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 474)*.

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:

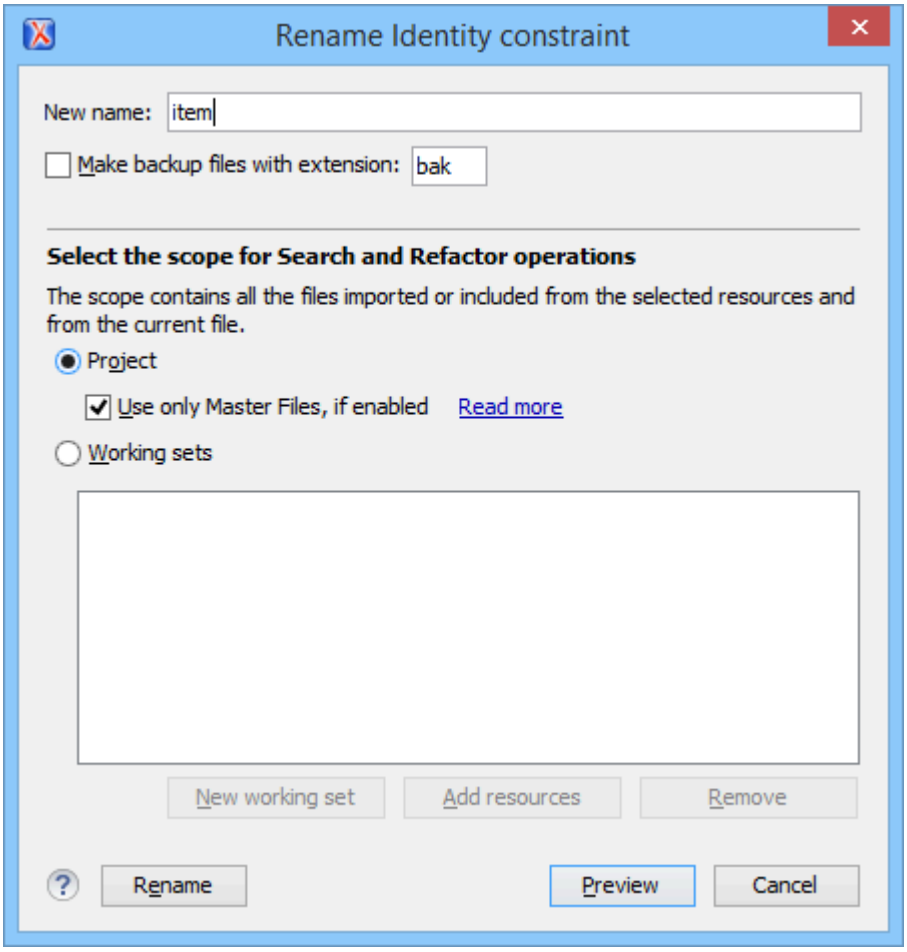
Rename Component

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.

Figure 240. Rename Identity Constraint Dialog Box



Related Information:

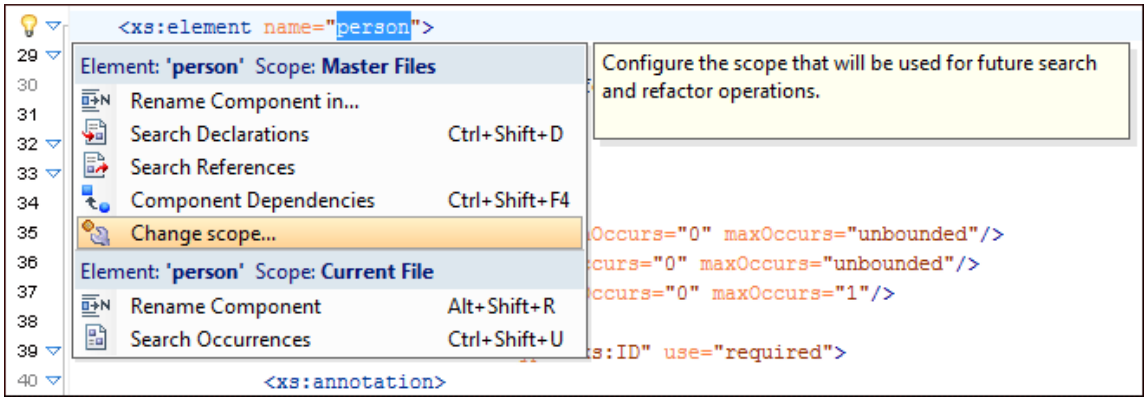
Search and Refactor Operations Scope *(on page 569)*

XML Schema Quick Assist Support

The *Quick Assist* support *(on page 2308)* 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 2308)* 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.

Figure 241. Quick Assist Support



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.

Resources

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:

Referenced/Dependent Resources View *(on page 739)*
Component Dependencies View *(on page 742)*
Searching and Refactoring Actions *(on page 744)*

Generating Sample XML Files

Oxygen XML Developer 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 172)*.

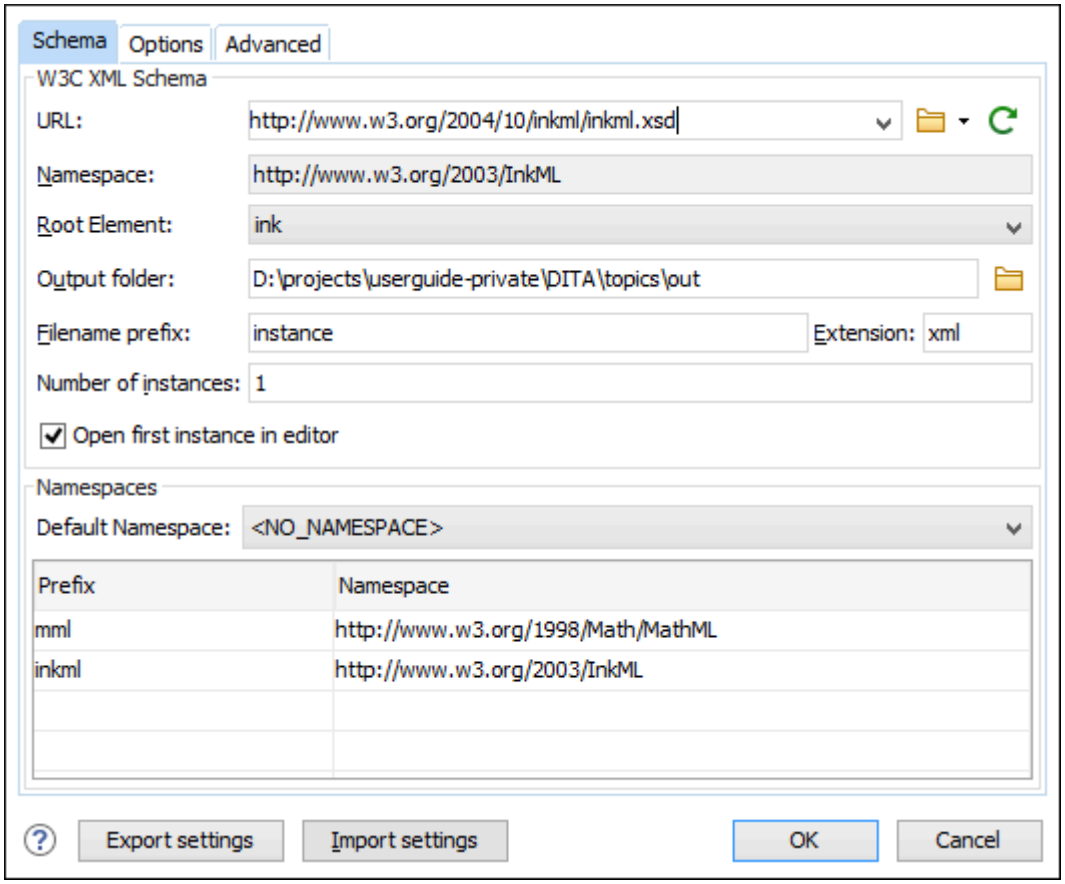
To generate sample XML files from an XML Schema, use the **Generate Sample XML Files** action from the **Tools** menu. This action is also available in the contextual menu of the schema Design mode *(on page 694)*. The action opens the **Generate Sample XML Files** dialog box that allows you to configure 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 176)*. 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 242. 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

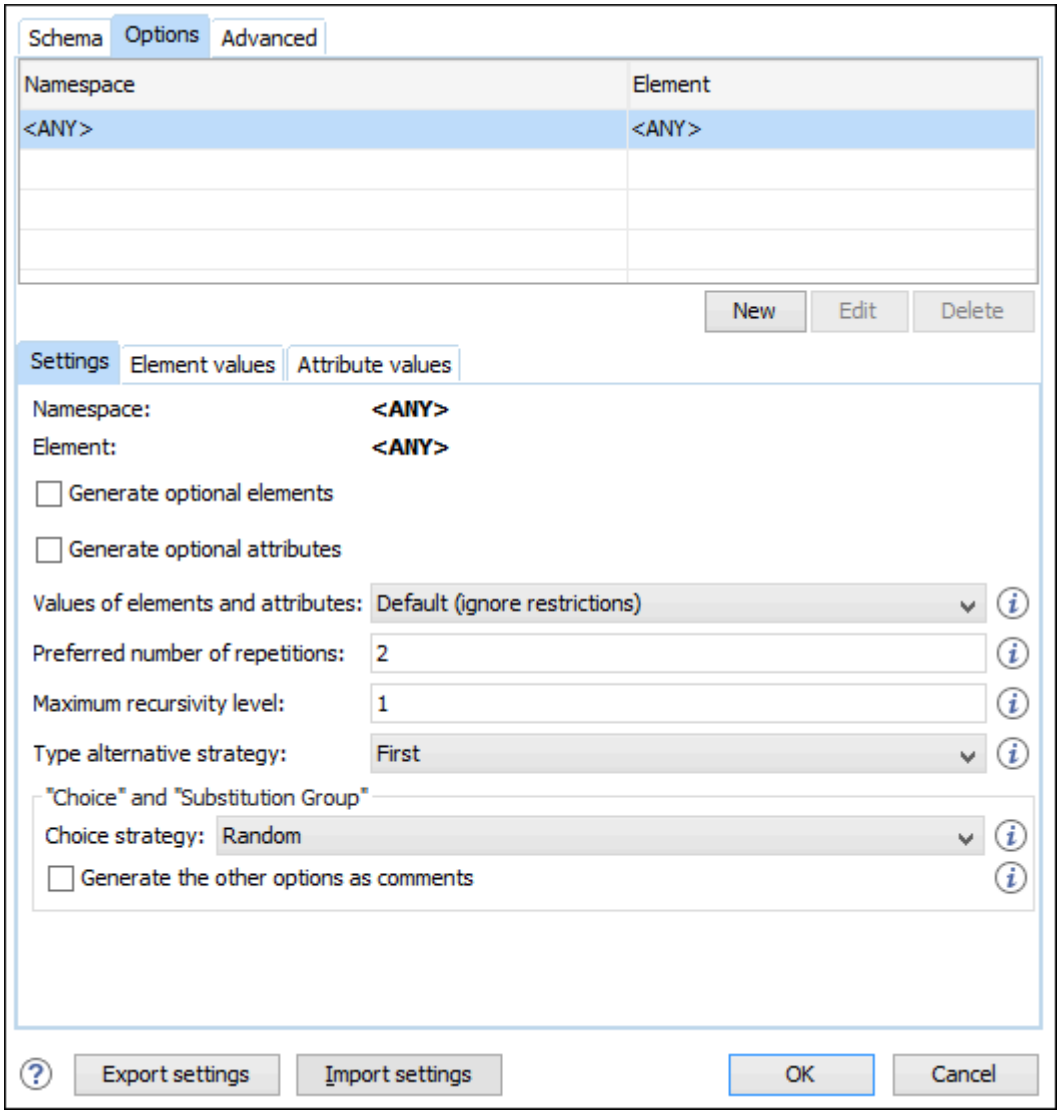
Use this button to load previously exported settings.

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 243. 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:
  - 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

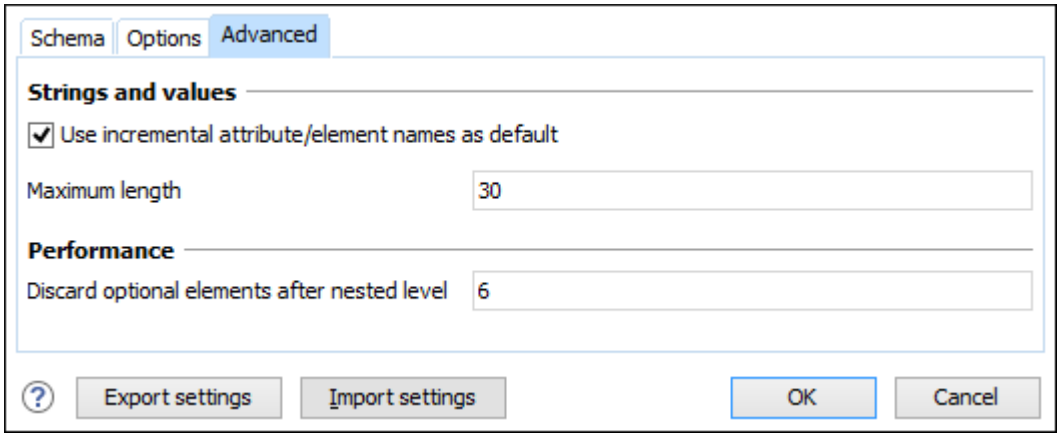
Use this button to load previously exported settings.

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 244. 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 2263)*.

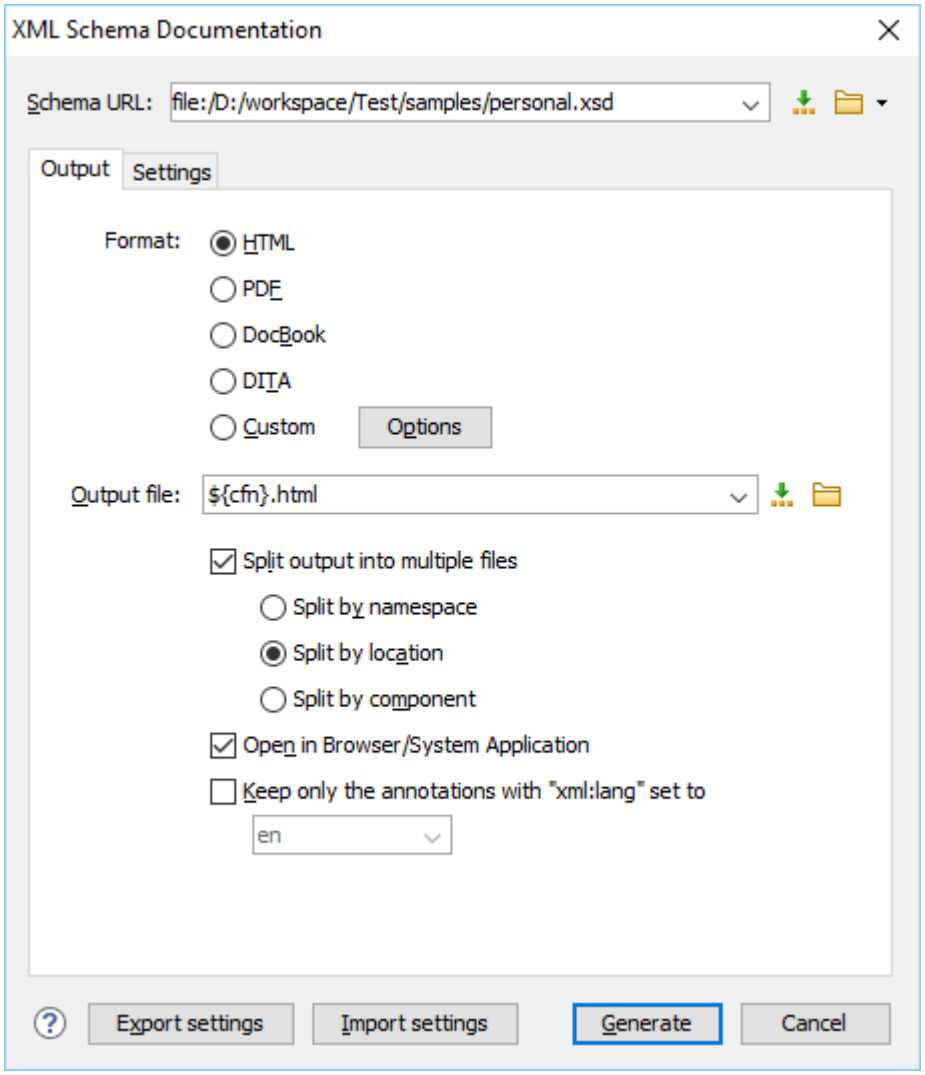
Generating Documentation for an XML Schema

Oxygen XML Developer 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 334)*. You can also open the tool by using the **Generate Documentation** toolbar button.

Figure 245. XML Schema Documentation Dialog Box



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 758)*.
  - **PDF** - The documentation is generated in PDF output format *(on page 761)*.
  - **DocBook** - The documentation is generated in DocBook output format *(on page 761)*.
  - **DITA** - The documentation is generated in DITA output format *(on page 761)*.
  - **Custom** - The documentation is generated in a custom output format *(on page 761)*, 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 Developer (in the current editor).

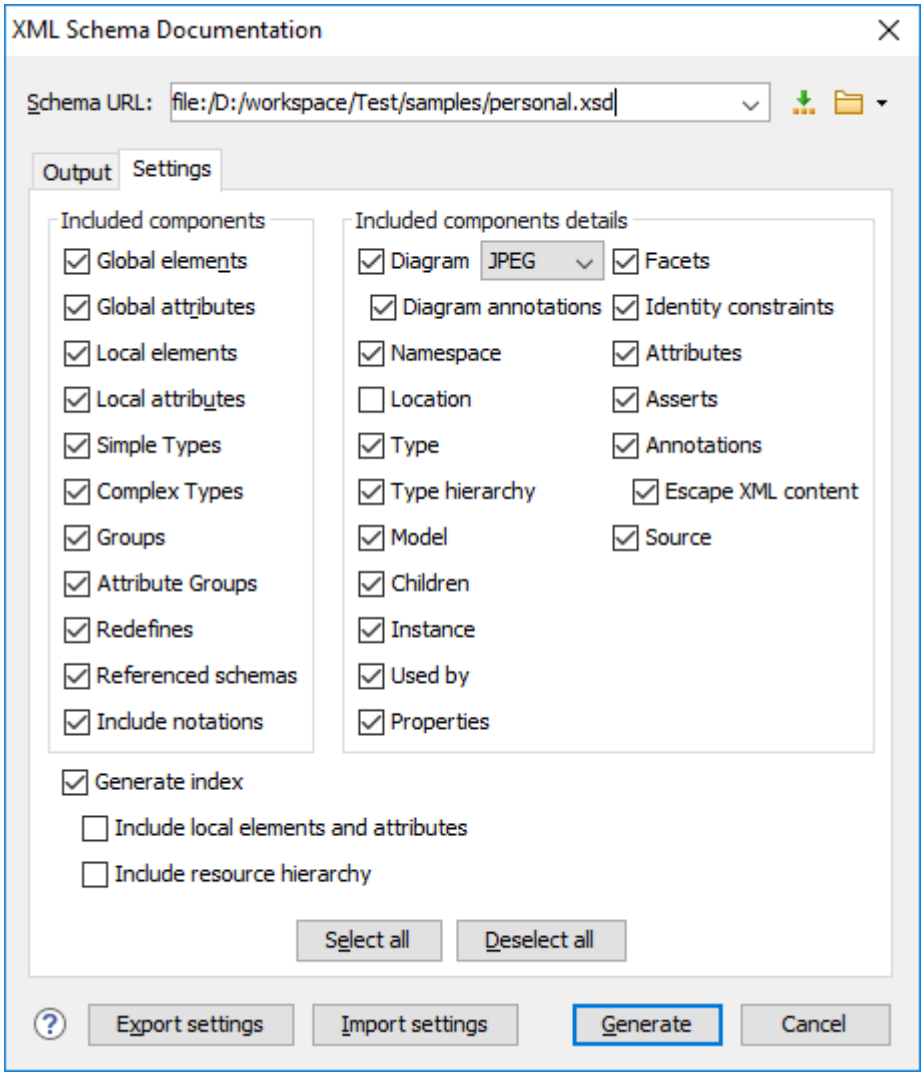
> **Note:**
> To set the browser or system application that will be used, open the **Preferences** dialog box **(Options > Preferences)** *(on page 79)*, go to **Global**, and set it in the **Default Internet browser** field. This will take precedence over the default system application settings.

- **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.

Figure 246. Settings Tab of the XML Schema Documentation Dialog Box



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 132)* 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 2263)*.

Related Information:
Customizing PDF or DocBook Output of Generated XML Schema Documentation *(on page 762)*

Output Formats for Generating XML Schema Documentation

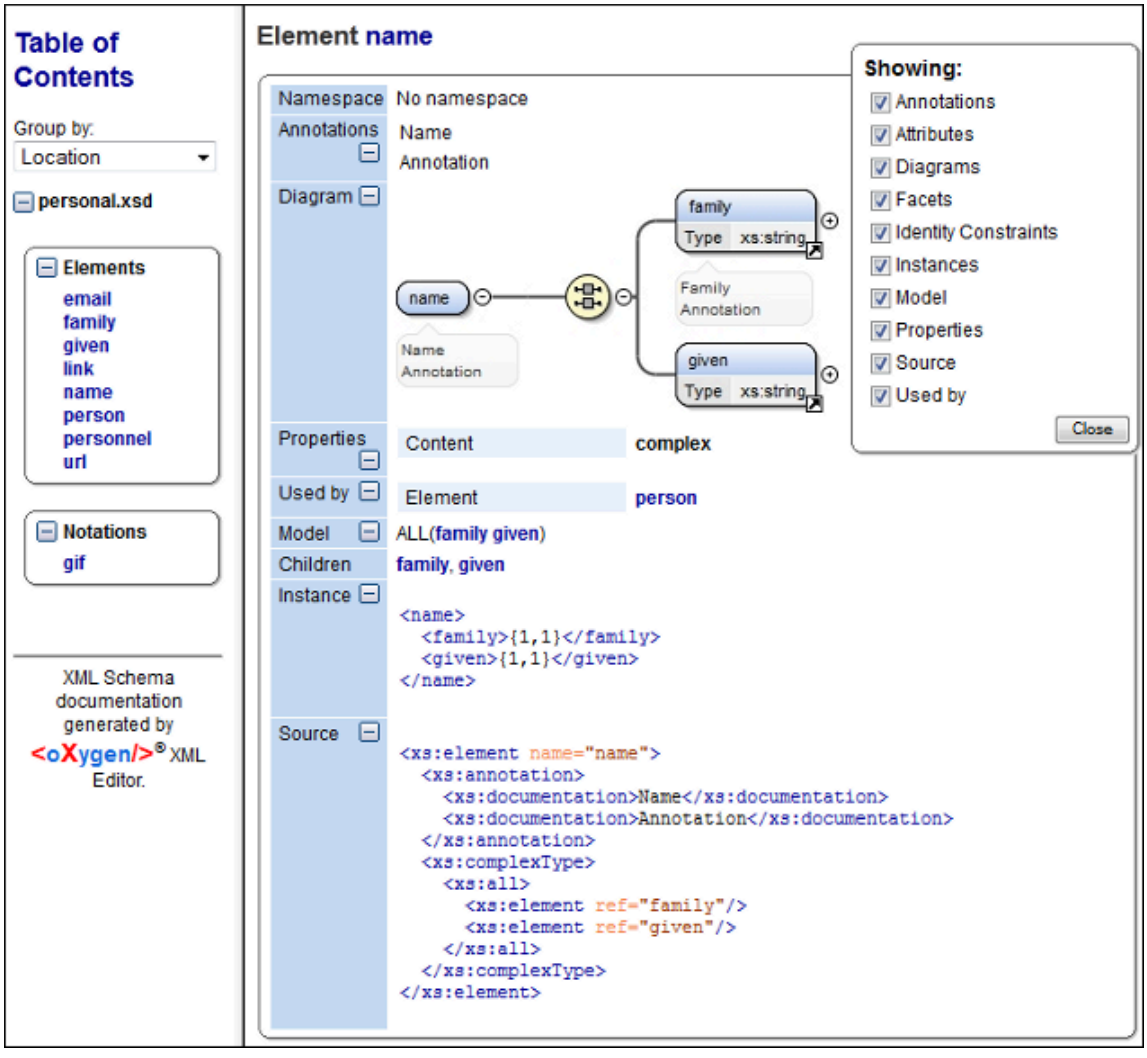
XML Schema documentation can be generated in HTML, PDF, DocBook, or a custom format. You can choose the format from the Schema Documentation *(on page 753)* dialog box. For the PDF and DocBook formats, the option to split the output in multiple files is not available.

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 280)* 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 `<xs:documentation>` 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.

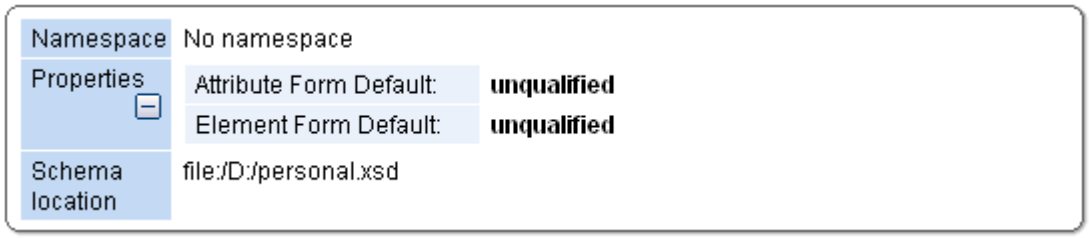
The generated images format is **PNG**. The image of an XML Schema component contains the graphical representation of that component as it is rendered in the schema diagram panel of the Oxygen XML Developer XSD editor panel. *(on page 281)*

Figure 247. 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.

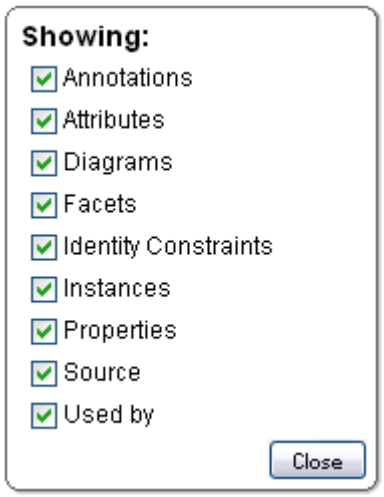
Figure 248. Information About a Schema



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.

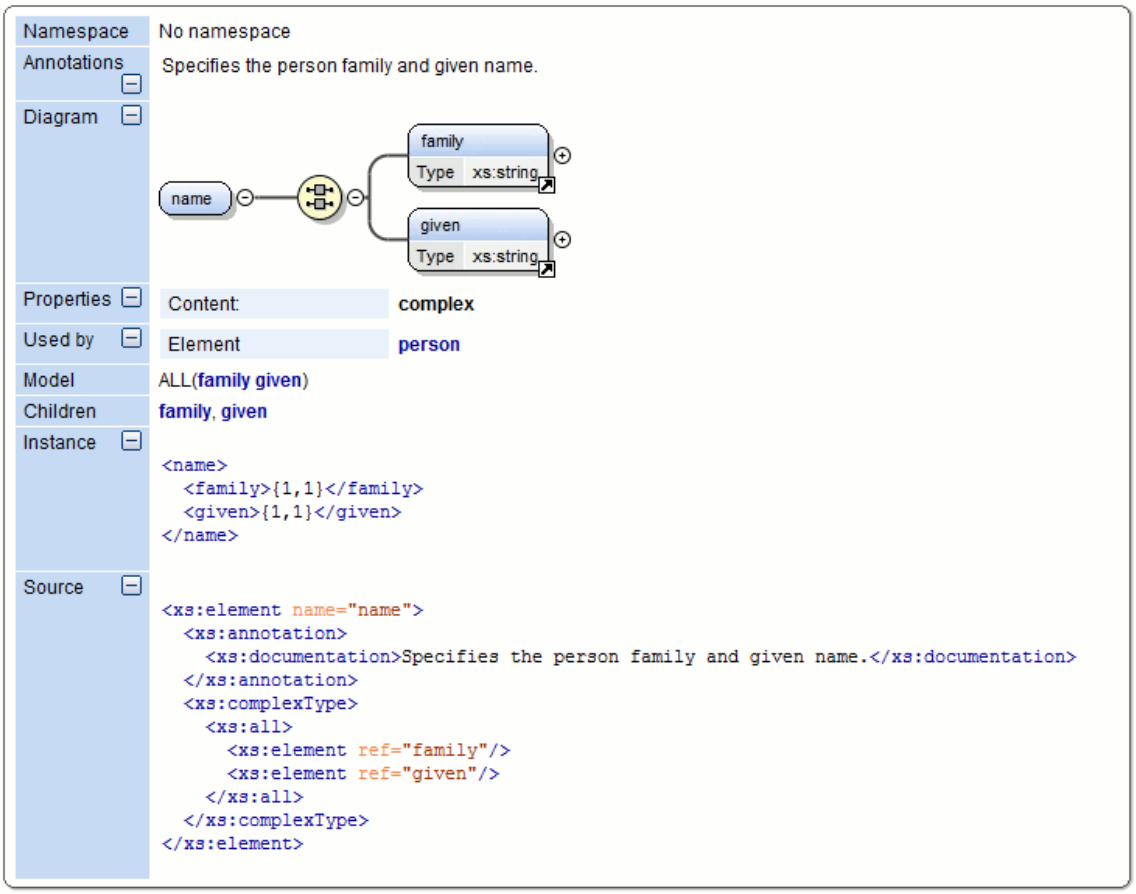
After the documentation is generated, you can collapse or expand details for some schema components by using the **Showing** options or the **Collapse** or **Expand** buttons.

Figure 249. Showing Options



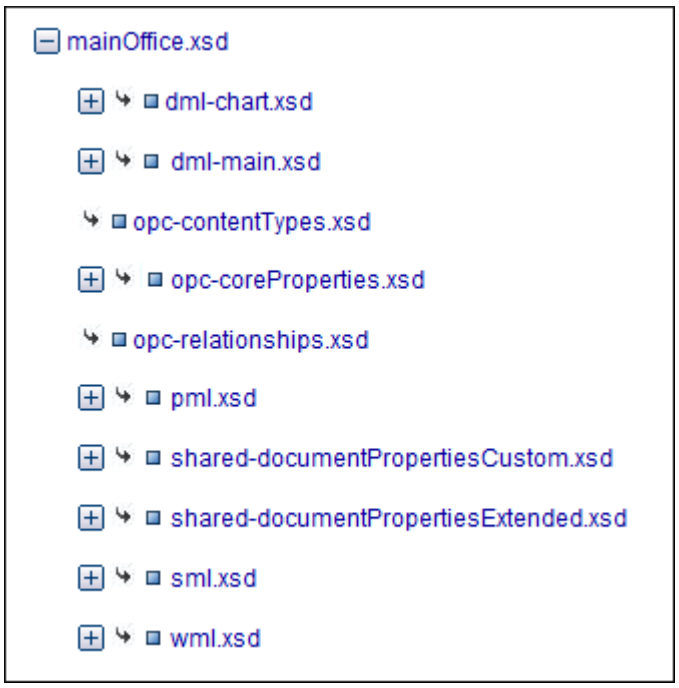
For each component included in the documentation, the section presents the component type follow by the component name. For local elements and attributes, the name of the component is specified as *parent name/component name*. You can easily go to the parent documentation by clicking the parent name.

Figure 250. Documentation for a Schema Component



If the schema contains imported or included modules, their dependencies tree is generated in the documentation.

Figure 251. Example: Generated 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 192)* preferences page.

For information about customizing the PDF output, see Customizing PDF or DocBook Output of Generated XML Schema Documentation *(on page 762)*.

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 1099)* (such as, *DocBook PDF* or *DocBook HTML*) on the output file, or configure your own transformation scenario *(on page 1101)* 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 762)*.

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 2304)* 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 1127)* 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 763)*.

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 755)* 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 *[OXYGEN_INSTALL_DIR]*`/frameworks/schema_documentation/xsl/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 `match="schemaDoc"` attribute between the `<info>` and `<xsl:apply-templates>` elements, as commented in the following example:

```
<info>
    <pubdate><xsl:value-of select="format-date(current-date(),
        '[Mn] [D], [Y]', 'en', (), ())"/></pubdate>
</info>
   <!-- Add the XSLT template content with match="schemaDoc" attribute here -->


<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 **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/xsdDocDocbook.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. When you return to the **XML Schema Documentation** dialog box, just click the **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  **Configure Transformation Scenario(s)** action from the toolbar or the **Document > Transformation** menu, click **New**, and select **XML transformation with XSLT**.
    b. In the **New Scenario** dialog box, go to the **XSL URL** field and choose the *[OXYGEN_INSTALL_DIR]*`/frameworks/docbook/oxygen/xsdDocDocbookCustomizationFO.xsl` file.
    c. Go to the **FO Processor** tab and select the **Perform FO Processing** and **XSLT result as input** options.
    d. Go to the **Output** tab and select the directory where you want to store the XML Schema documentation output and click **OK**.
    e. Click **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 2306)* 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 Developer 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:

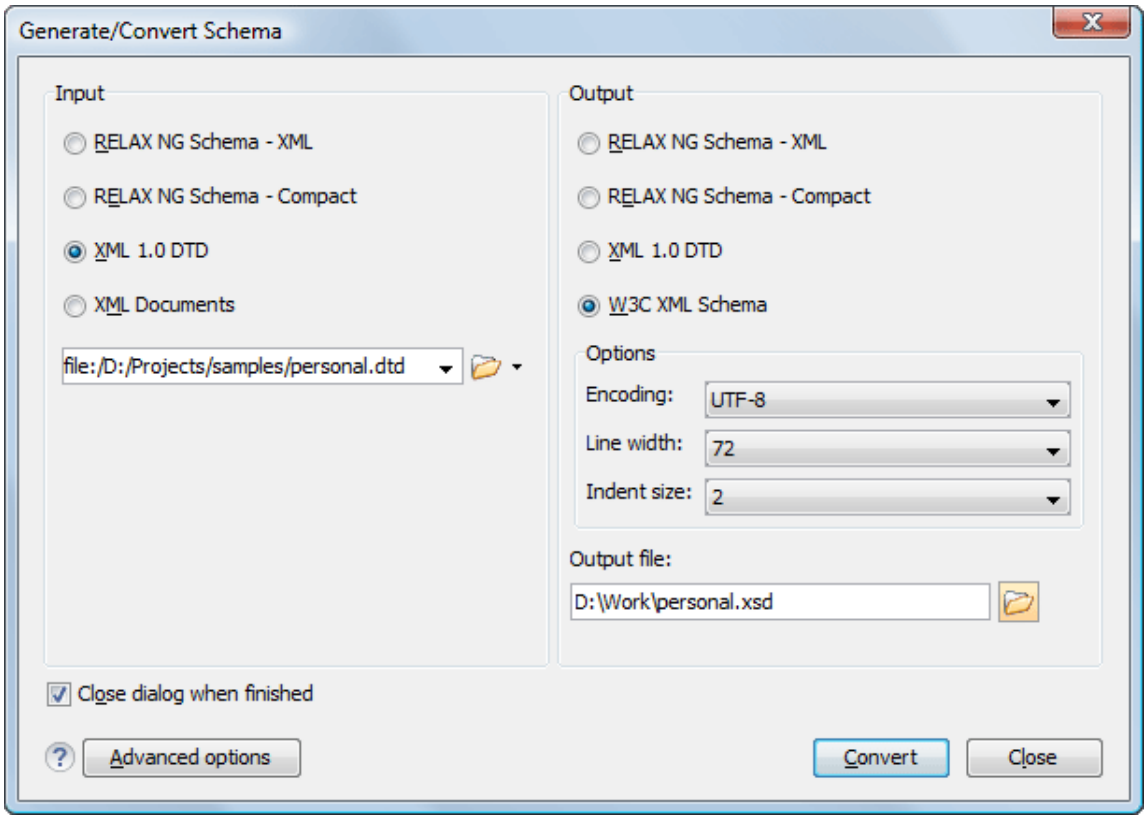
1. Customize the *`[OXYGEN_INSTALL_DIR]`*`/frameworks/schema_documentation/xsl/xsdDocDita.xsl` stylesheet to incorporate your desired changes.
2. Create an intermediary file that holds the content of your XML Schema documentation by following these steps:
    a. Go to **Tools > Generate Documentation > 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 755)* 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 Developer generates an approximation of the source schema. Oxygen XML Developer 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 + Option + 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 334)*. This action opens the **Generate/Convert Schema** dialog box that allows you to configure various options for conversion.

Figure 252. 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 `<anyAttribute>` elements. The default is skip (corresponding to RELAX NG semantics).

Converting Database to XML Schema

Oxygen XML Developer 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 205)* 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**.

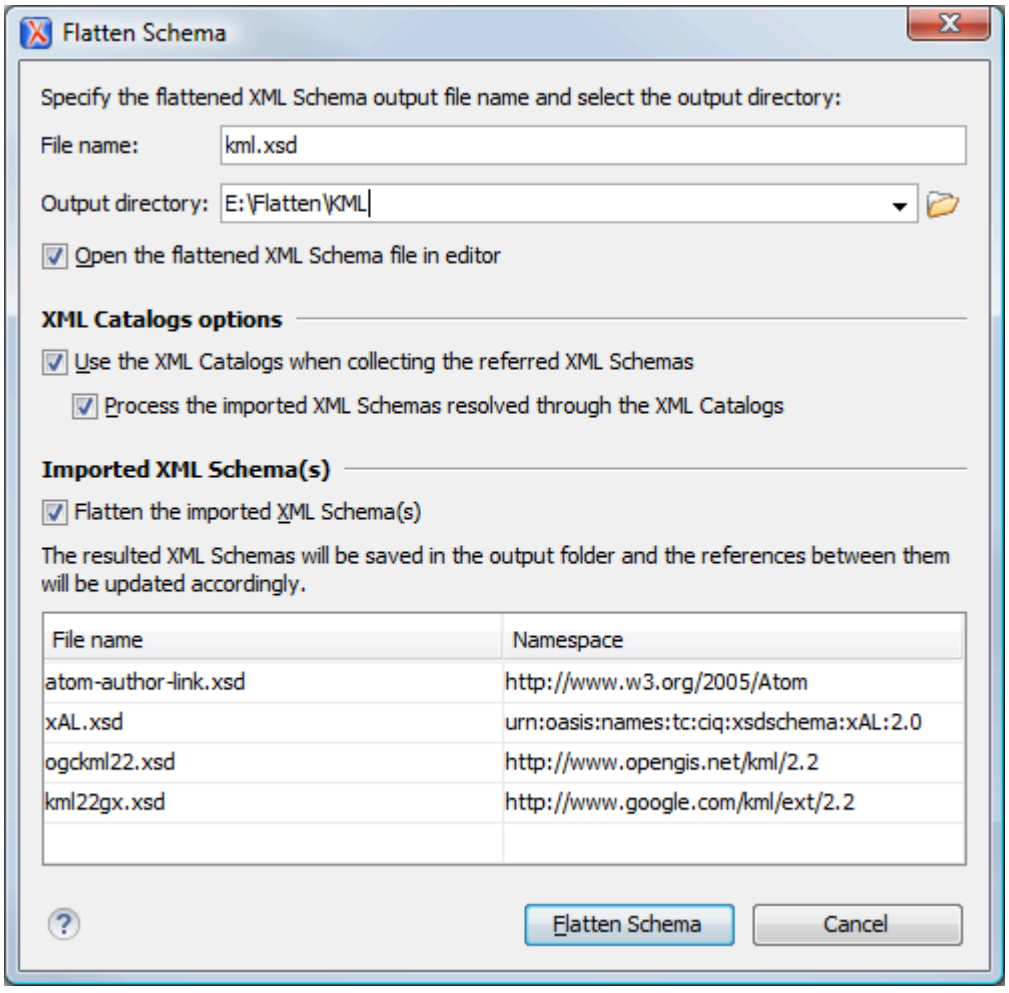
Result: The database structure is converted to an XML Schema and it is opened for viewing and editing.

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 Developer 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 253. 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 Developer 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 2309)*.

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 2309)* 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 2263)*.

Generating Java Classes from XML Schema

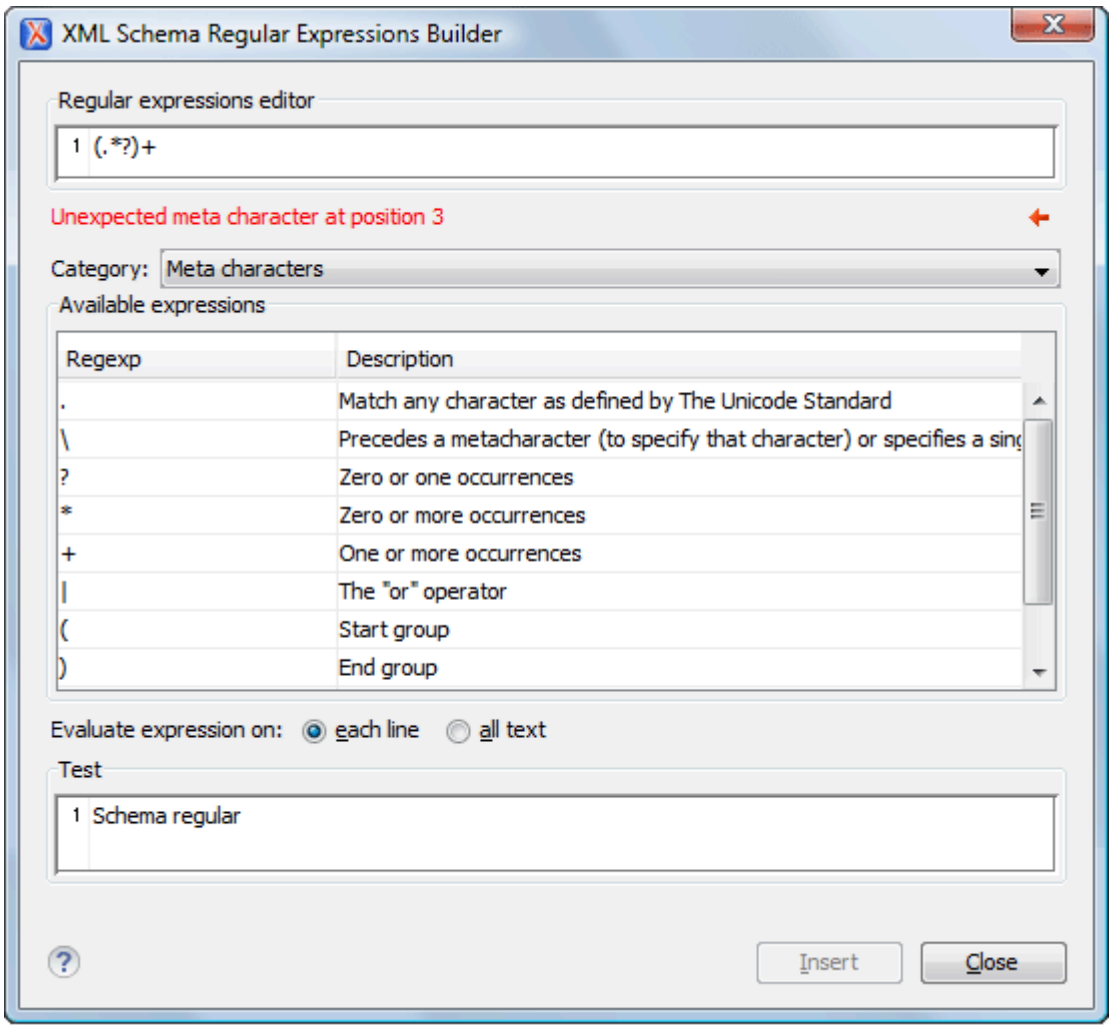
Oxygen XML Developer includes a tool for generating Java classes from an XML Schema (XSD) file. The **Generate Java classes from XML Schema (XSD)** action for invoking the tool can be found in the **Tools** menu. It requires an additional add-on to be installed, so the first time you invoke the action, Oxygen XML Developer will present a dialog box asking if you want to install it. Once installed, you need to restart Oxygen XML Developer and the action will invoke the Java class generator tool.

Add-on documentation and installation details.

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 254. 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**.

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 279)* or schema **Design** mode *(on page 280)*. 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 Developer offers full support for XML Schema 1.1, including:

- XML Documents Validation *(on page 514)* and Content Completion *(on page 457)* based on XML Schema 1.1.
- XML Schema 1.1 Validation *(on page 732)* and Content Completion *(on page 733)*.
- Editing XML Schema 1.1 files in the Schema **Design** mode *(on page 280)*.
- The Flatten Schema *(on page 767)* action.
- Referenced/Dependent Resources *(on page 739)* and Refactoring Actions *(on page 744)*.
- *Main files (on page 2306)*.
- Generating Documentation for XML Schema 1.1 *(on page 753)*.
- 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.



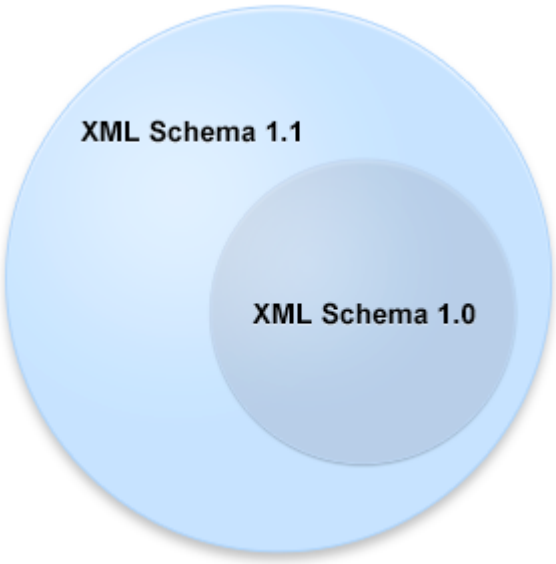
Note:
To enable XML Schema 1.1 validation in NVDL, you need to pass the following option to the validation engine to specify the schema version: `http://www.thaiopensource.com/validate/xsd-version` (the possible values are `1.0` or `1.1`.



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 255. 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 2303)* 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 79)* and go to **XML > XML Parser > XML Schema**.

Resources

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 772)*

Setting the XML Schema Version

Oxygen XML Developer 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 33. Using the *minVersion* and *maxVersion* Attributes to Set the XML Schema Version

Versioning Attributes	XML Schema Version
minVersion = "1.0" **maxVersion** = "1.1"	1.0
minVersion = "1.1"	1.1
minVersion = "1.0" **maxVersion** = greater than "1.1"	The XML Schema version defined in the **XML Schema** preferences page *(on page 172)*
Not set in the XML Schema document	The XML Schema version defined in the **XML Schema** preferences page *(on page 172)*

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 Developer to its default options.

Oxygen XML Developer 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 Developer takes into account the *minVersion* and *maxVersion* attributes defined in the XML Schema.

Related Information:
XML Schema 1.1 *(on page 771)*

Editing XQuery Documents

XQuery is the query language for XML and is officially defined by a W3C Recommendation document. Oxygen XML Developer provides support for XQuery 4.0 and 3.1 (which is also backwards compatible with XQuery 3.0 and 1.0).

The many benefits of XQuery include:

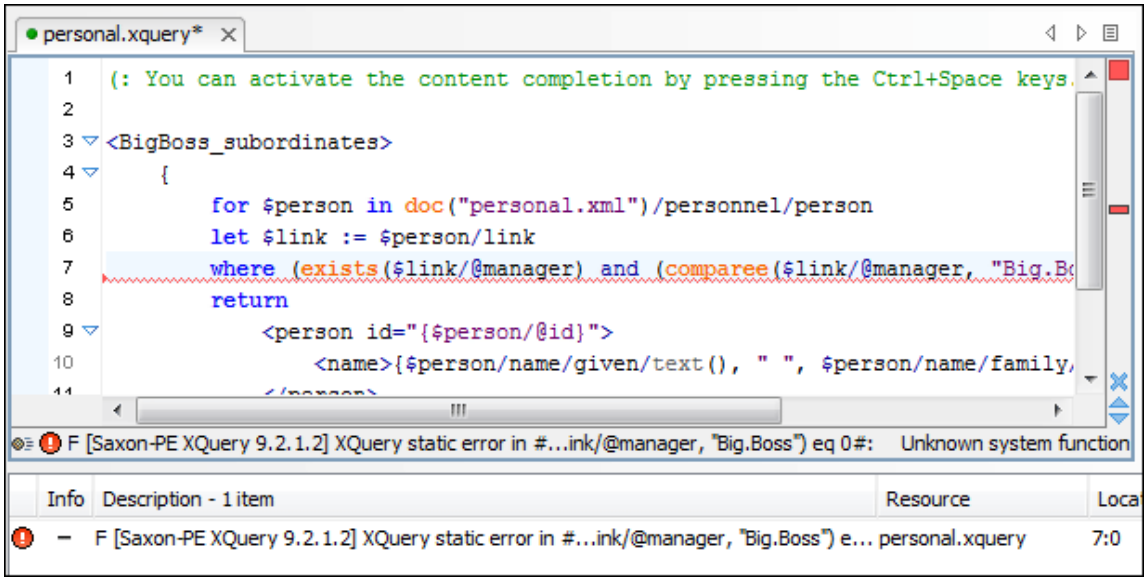
- XQuery allows you to work in one common model no matter what type of data you are working with: relational, XML, or object data.
- XQuery is ideal for queries that must represent results as XML, to query XML stored inside or outside the database, and to span relational and XML sources.
- XQuery allows 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 1795)*

XQuery Validation

With Oxygen XML Developer, you can validate your documents before using them in your transformation scenarios. The validation uses the Saxon 12.9 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 1784)* 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 256. XQuery Validation



Note:
If you choose a processor that does not support XQuery validation, Oxygen XML Developer 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 185)* in the Oxygen XML Developer preferences.

When you open an XQuery document from a connection that supports validation (for example, MarkLogic, or eXist), by default Oxygen XML Developer 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 Developer provides content completion for keywords and all known XQuery functions and operators. The *Content Completion Assistant (on page 2303)* 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 185)* page. For more information about the support for working with XQuery with regard to databases, see XQuery and Databases *(on page 1795)*.

The extension functions included in the Saxon engine are available on content completion if one of the following conditions are true:

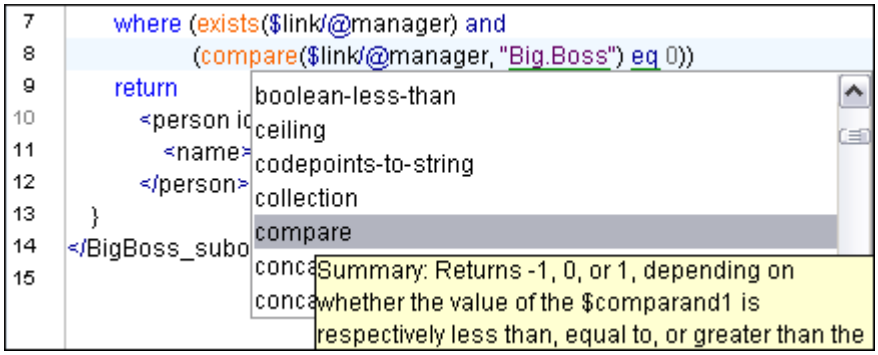
- The edited file has a transformation scenario associated that uses as transformation engine Saxon 12.9 PE or Saxon 12.9 EE.
- The edited file has a validation scenario associated that use as validation engine Saxon 12.9 PE or Saxon 12.9 EE.
- The validation engine specified in Preferences *(on page 185)* is Saxon 12.9 PE or Saxon 12.9 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.

Figure 257. XQuery Content Completion



Syntax Highlighting in XQuery

Oxygen XML Developer 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 79)*.
2. Go to **Editor > Syntax Highlight** *(on page 158)*.
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 158)*

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 Developer 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:

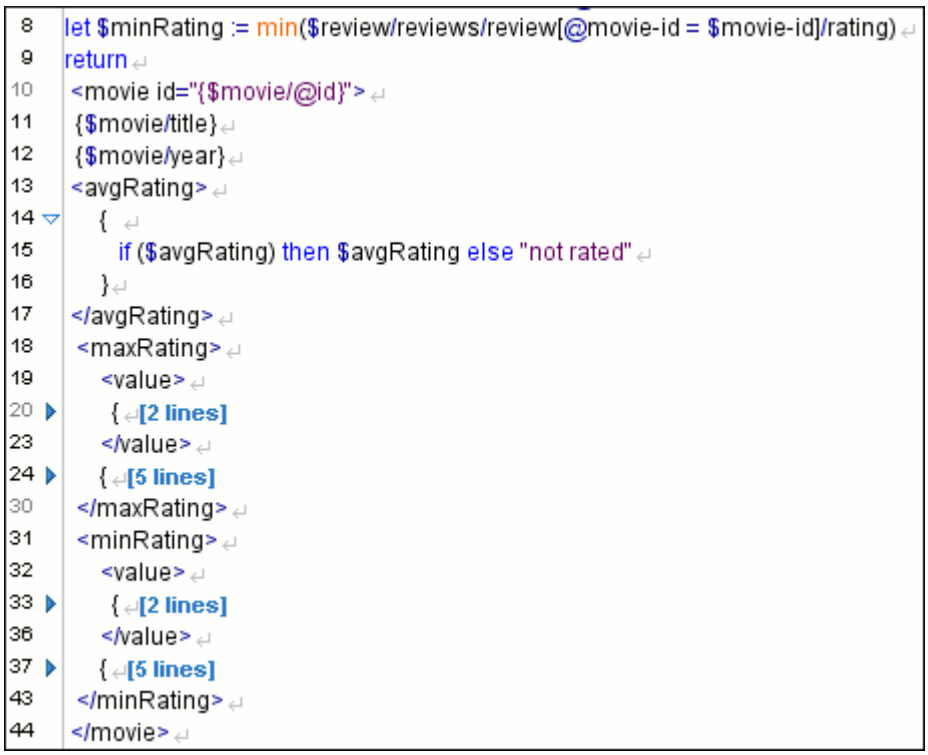
- Manages whitespaces, by collapsing or inserting space characters where needed.
- Formats complex expressions on multiple, more readable lines by properly indenting each of them. The amount of whitespaces that form an indent unit is controlled through one of the **Indent with tabs** and **Indent size** options from the **Format Preferences** page *(on page 136)*.

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 474)*.

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 454)* are also available in XQuery documents.

Figure 258. Folding in XQuery Documents

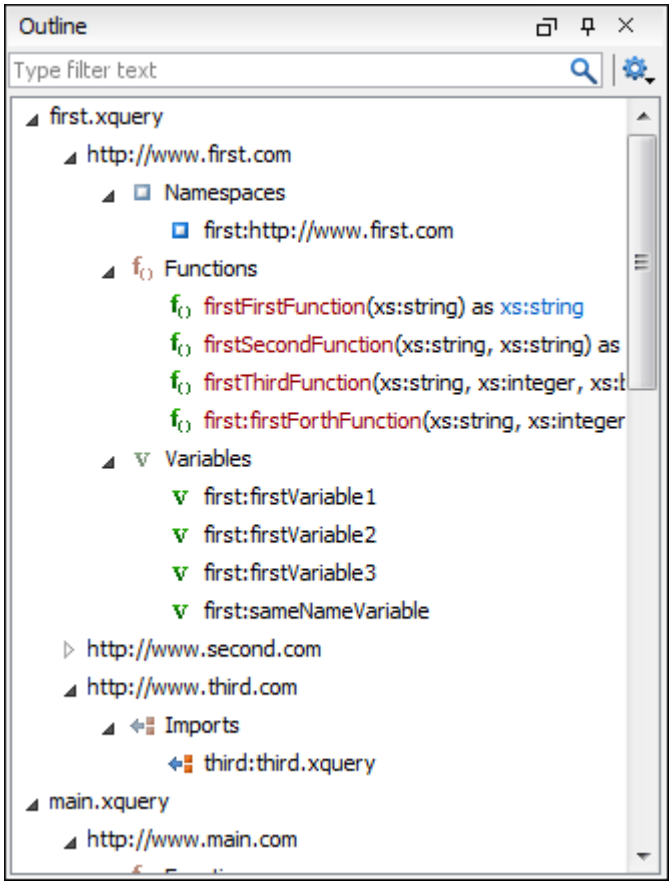


There is available the action **Go to Matching Bracket Ctrl + Shift + G (Command + Shift + G on macOS)** on contextual menu of XQuery editor for going to matching character when cursor is located at '{' character or '}' character. It helps for finding quickly matching character of current *folding element (on page 2305)*.

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.

Figure 259. XQuery Outline View



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 or JSON 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 Developer 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 Developer 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 Developer 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 at the bottom of the application.



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 Developer 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 397)* or when the scope is other than **Current file**.

Options

Opens the Preferences page of the currently selected processing engine.

XPath scope menu

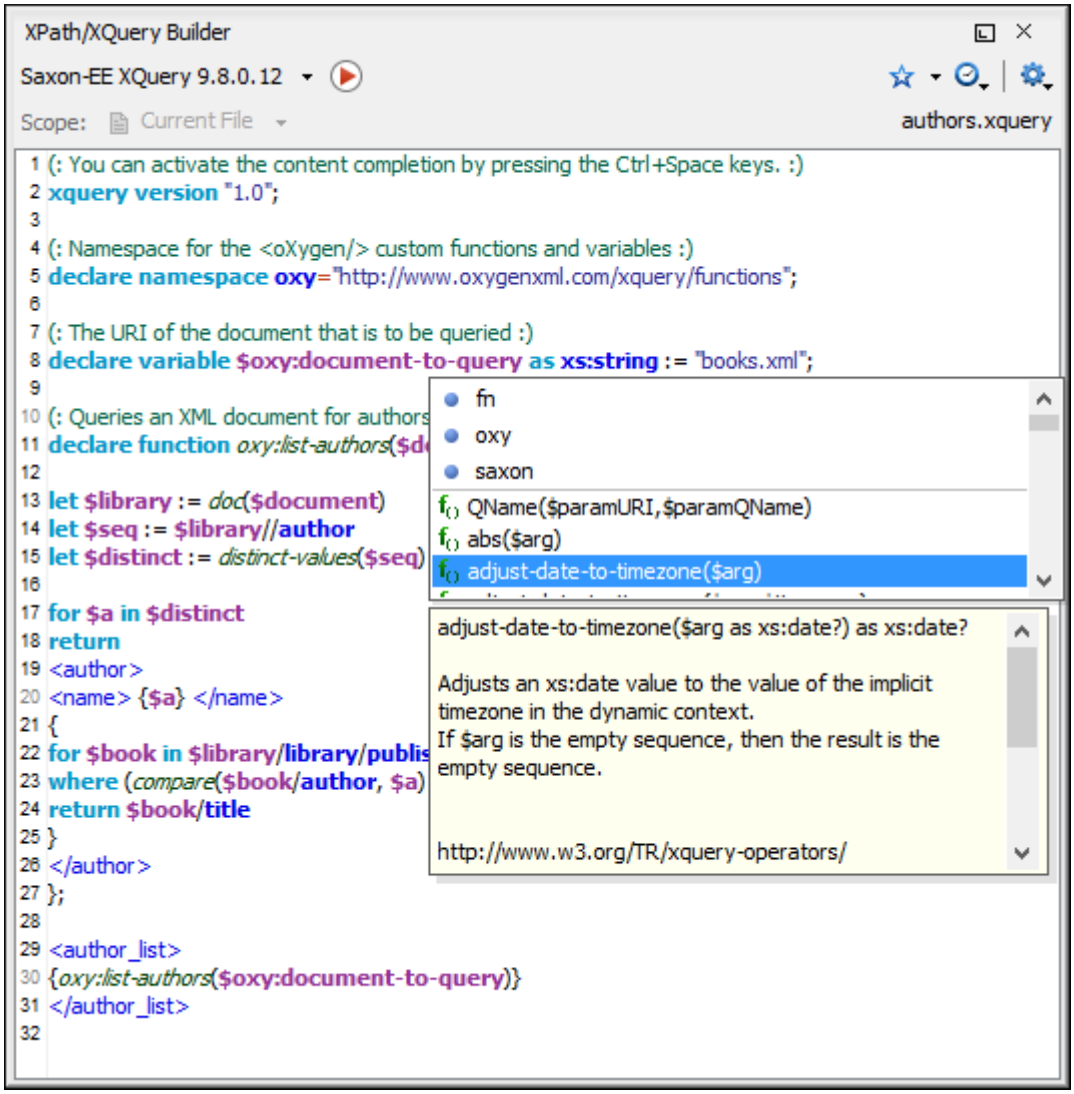
Oxygen XML Developer allows you to define a scope for the XPath operation to be executed. You can choose where the XPath expression will be executed:

- **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.
- **Opened archive** - Files that are opened in the **Archive Browser** view *(on page 1735)*.
- **Working sets** - The selected *working sets (on page 2309)*.

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 2309)*.
- **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 260. XPath/XQuery Builder View



While you edit an XPath or XQuery expression, Oxygen XML Developer assists you with the following features:

- *Content Completion Assistant (on page 2303)* - 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 79)* and go to **Editor > Syntax Highlight** *(on page 158)*.
- 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.

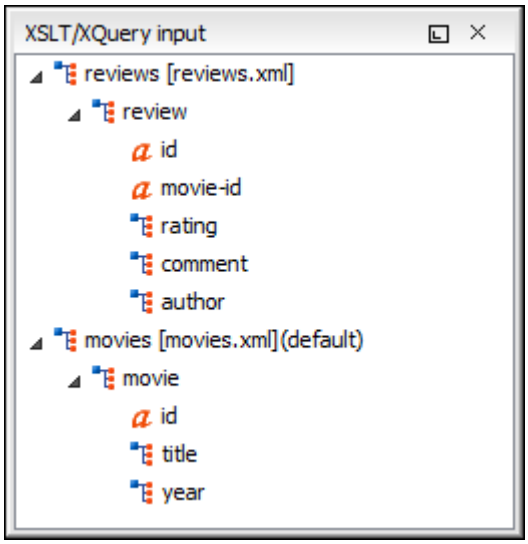
The usual edit actions (**Cut**, **Copy**, **Paste**, **Select All**, **Undo**, **Redo**) are available in the contextual menu of the top editable part of the view.

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.

Figure 261. XQuery Input View



Example:

For the following XML documents:

```
<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">
```

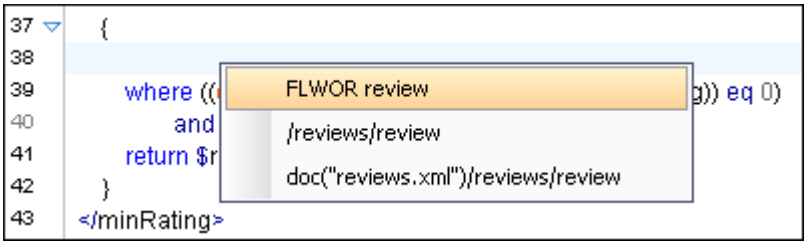
```
<rating>5</rating>
<comment>It is made after a great Stephen King book.
</comment>
<author>Paul</author>
</review>
<review id="101" movie-id="1">
<rating>3</rating>
<comment>Tom Hanks does a really nice acting.</comment>
<author>Beatrice</author>
</review>
<review id="104" movie-id="2">
<rating>4</rating>
<comment>Robert De Niro is my favorite actor.</comment>
<author>Maria</author>
</review>
</reviews>
```

and the following 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:



Select **FLWOR review**, the resulting document will look like this:

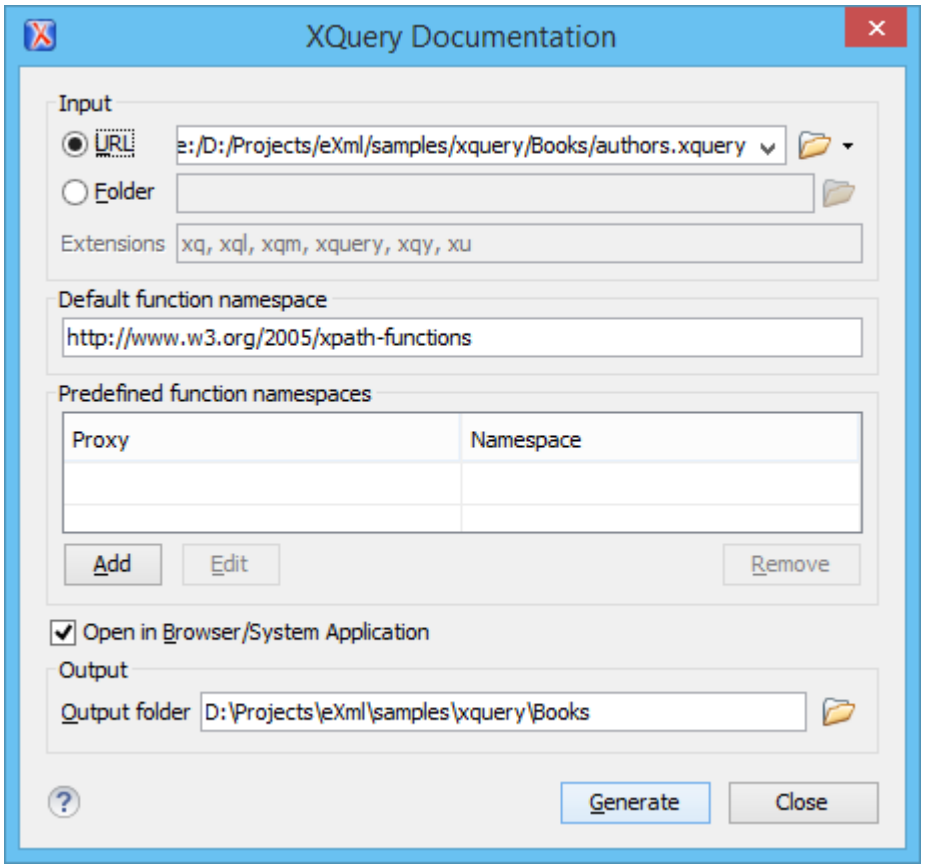
```
37      {
38          for $review in doc("reviews.xml")/reviews/review
39          return
40          where ((compare($rev/rating/text(), string($minRating)) eq 0)
41                  and ($rev/@movie-id = $movie/@id))
42          return $rev/author
43      }
```

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 334)*. 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 262. 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:
  - **URLFile** - 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 79)*, 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 1101)*. The result can be saved and opened in the associated application. You can even run a FO processor *(on page 1174)* on the output of an XQuery. The transformation scenarios may be shared between many XQuery files, are exported *(on page 251)* together with the XSLT scenarios and can be managed in the **Configure Transformation Scenario** dialog box *(on page 1210)*, or in the **Scenarios** view *(on page 1217)*. 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 1102)*. 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 12.9 HE, Saxon 12.9 PE, Saxon 12.9 EE processors, a database connection (details can be found in the Working with Databases *(on page 1742)* chapter - in the XQuery transformation *(on page 1796)* section) or any XQuery processor that provides an XQJ API implementation.

The Saxon 12.9 EE processor also supports XQuery 3.1 transformations.

Related Information:
XQuery and Databases *(on page 1795)*

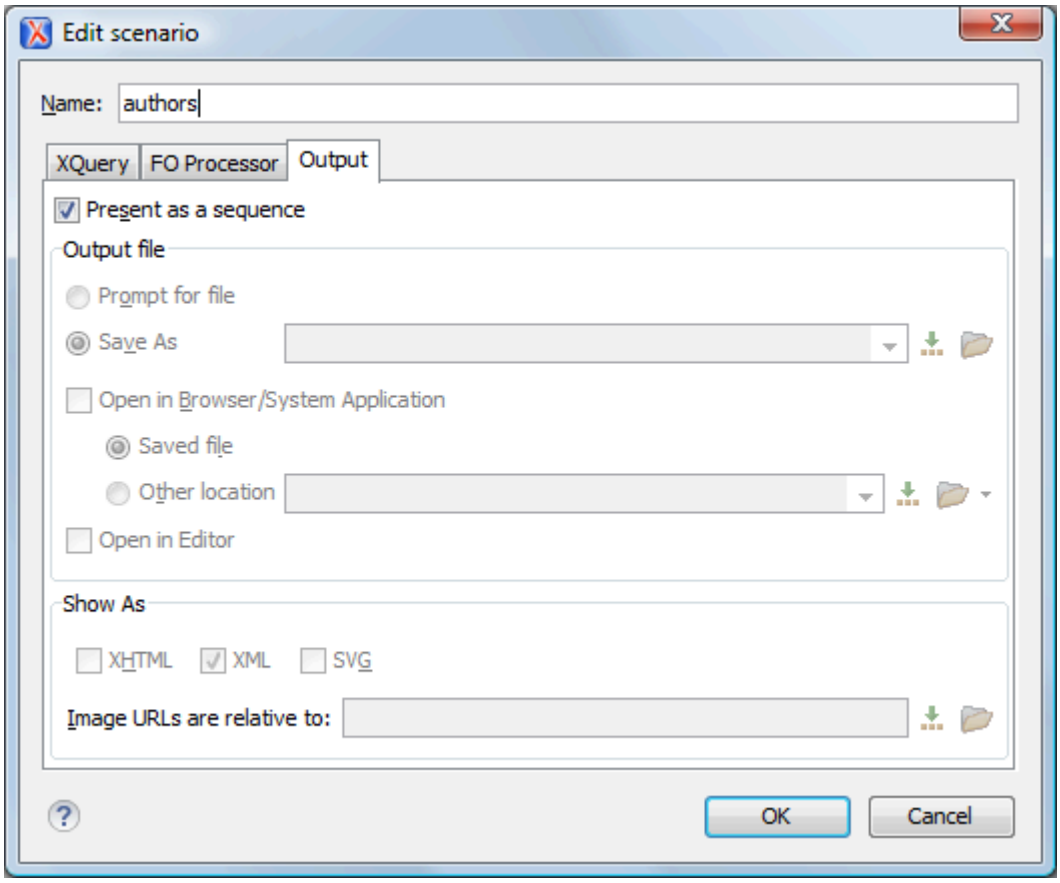
Display XQuery Result in Sequence View

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 1123)* 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.

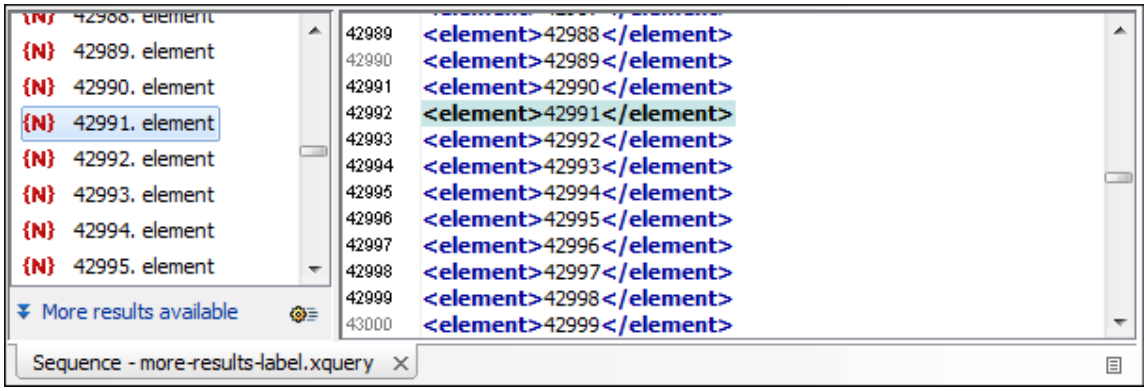
The size of a chunk can be set with the **Size limit of Sequence view** option *(on page 185)*. The **XQuery options** button from the **More results available** label provides a quick access to this option by opening the **XQuery** preferences page *(on page 185)* where the option can be modified.

Figure 263. XQuery transformation result displayed in Sequence view



A chunk of the XQuery transformation result is displayed in the **Sequence** view.

Figure 264. XQuery transformation result displayed in Sequence view



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 186)* 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 2305)* defined in the preferences page.

Saxon-HE/PE/EE Options

The advanced options for Saxon 12.9 Home Edition (HE), Professional Edition (PE), and Enterprise Edition (EE) are as follows:

Use a configuration file ("-config")

Sets a Saxon 12.9 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 1123)* is not set.

Saxon-PE/EE Options

The following advanced options are specific for Saxon 12.9 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 12.9 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 12.9 EE XQuery processor Oxygen XML Developer 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 12.9 EE as a transformer in the scenario associated with the XQuery files containing update statements and Oxygen XML Developer will notify you if the update was successful.

Example: Using XQuery Update to modify a tag name in an XML file

```
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  **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 2305)*.

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.

```
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.

```
<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 (Deprecated)

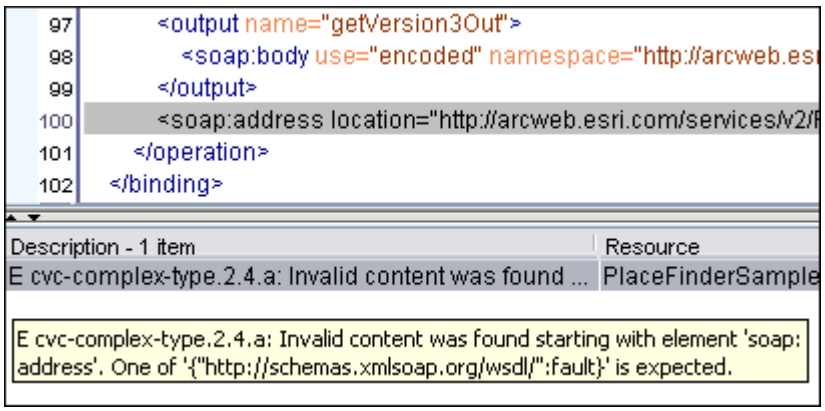
WSDL is an XML format for describing network services as a set of endpoints operating on messages containing either document-oriented or procedure-oriented information. The operations and messages are described abstractly, and then bound to a concrete network protocol and message format to define an endpoint. Related concrete endpoints are combined into abstract endpoints (services).

Oxygen XML Developer provides a special type of editor dedicated to WSDL documents. The WSDL editor offers support for validation, a specialized *Content Completion Assistant (on page 2303)*, a component oriented **Outline** view *(on page 795)*, 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 Developer 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.

Figure 265. Validating a WSDL file



Resources

For more information about the WSDL editing support in Oxygen XML Developer, watch our video demonstration:

https://www.youtube.com/embed/OS5Ucm9b8sY

Related Information:

Editing XML Documents in Text Mode *(on page 442)*

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 Developer 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 cat set a main WSDL document either using the *main files* support from the **Project** view *(on page 350)*, 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 Developer 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 2306)* include:

- Correct validation of a module in the context of a larger WSDL structure.
- *Content Completion Assistant (on page 2303)* displays all components valid in the current context.
- The **Outline** view *(on page 795)* 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.

Resources

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 79)*, go to **Editor > Document Checking**, and deselect the **Enable automatic validation** option *(on page 160)*.

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.

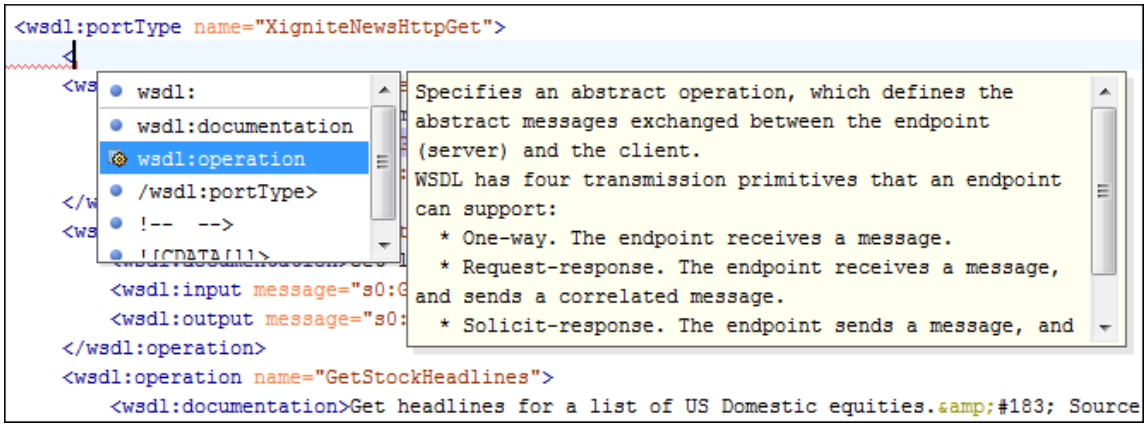
Content Completion Assistance in WSDL Documents

The *Content Completion Assistant (on page 2303)* 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** 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.

Figure 266. WSDL Content Completion Assistant



Note:
If you are using the concept of *main files (on page 2306)* 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 Developer, see Defining Main Files at Project Level *(on page 350)*.

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 267. Namespace Prefixes in the Content Completion Assistant



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 Developer provides an easy mode to declare them by proposing a prefix for these namespaces.

WSDL Syntax Highlighting

Oxygen XML Developer 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 79)*.
2. Go to **Editor > Syntax Highlight** *(on page 158)*.
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 Developer 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 159)*.

Related Information:
Customize Syntax Highlight colors *(on page 158)*

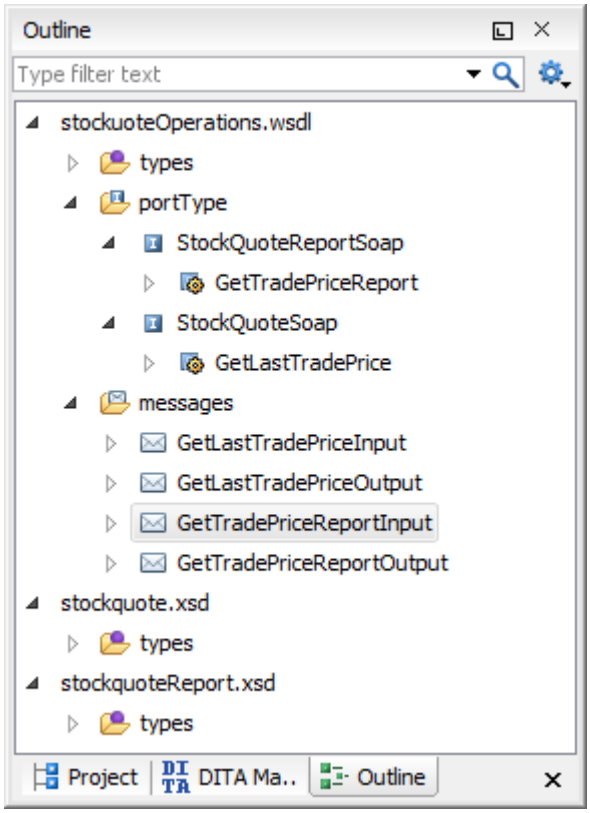
WSDL Outline View

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 350)*, 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 268. 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 235)*.

Configure displayed attributes

Displays the **XML Structured Outline** preferences page *(on page 235)*.

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 802)* that displays the dependencies of the currently selected component.

Show referenced resources

Opens the **Referenced/Dependent Resources** view *(on page 800)* that displays the references for the currently selected resource.

Show dependent resources

Opens the **Referenced/Dependent Resources** view *(on page 800)* 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 802)* 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.

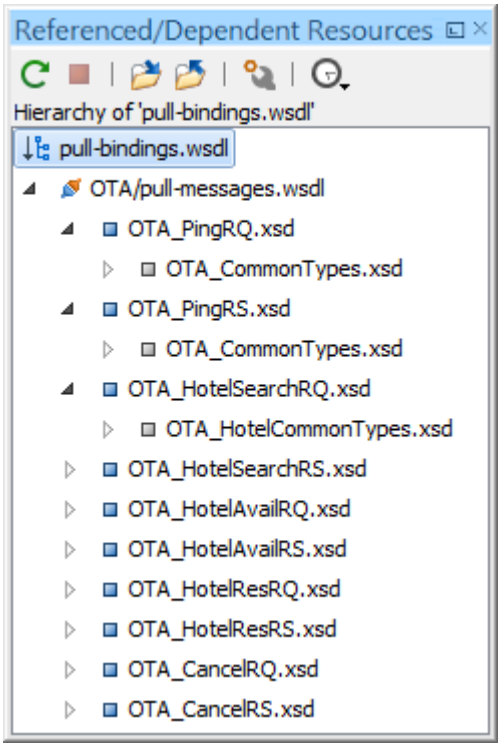
WSDL Referenced/Dependent Resources View in WSDL Documents

The **Referenced/Dependent Resources** 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 references or dependencies of a WSDL document, select the document in the **Project** view *(on page 334)* and choose **Show referenced resources** or **Show dependent resources** from the contextual menu.

Figure 269. Referenced/Dependent Resources View



The following actions are available on the toolbar of the **Referenced/Dependent Resources** view:

Refresh

Refreshes the resource structure.

Stop

Stops the computing.

Show hierarchy for

Computes the hierarchical structure of the references for a resource.

Show dependencies for

Computes the structure of the dependencies for a resource.

Configure dependencies search scope

Allows you to configure a scope to compute the dependencies. 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 **Referenced/Dependent Resources** view contains the following actions:

Open

Opens the resource. You can also double-click a resource within the hierarchical 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.

Show references resources

Shows the references for the selected resource.

Show dependent resources

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 hierarchical structure.

Collapse All

Collapses all children of the selected resource from the hierarchical structure.

Tip:
When a recursive reference is encountered in the 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 802)*.

Related Information:
Modular Contextual XML Editing Using 'Main Files' Support *(on page 567)*
Search and Refactor Operations Scope *(on page 569)*

Moving/Renaming WSDL Resources

You can move and rename a resource presented in the **Referenced/Dependent Resources** 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 **Referenced/Dependent Resources** 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 **Referenced/Dependent Resources** 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.

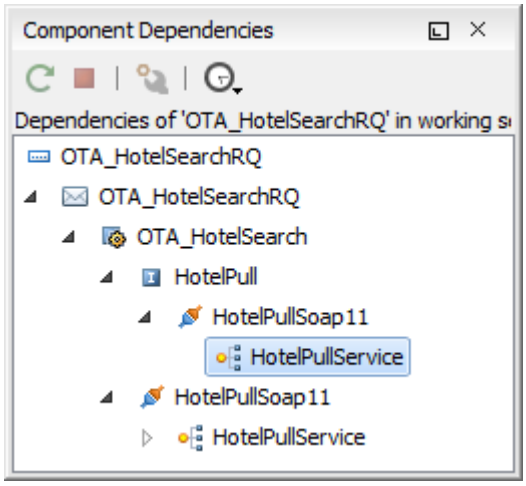
To see the dependencies of a WSDL component:

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 .

Figure 270. Component Dependencies View



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 806)*

Highlight Component Occurrences in WSDL Documents

When you position your mouse cursor over a component in a WSDL document, Oxygen XML Developer 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 79)* 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 474)*.

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:

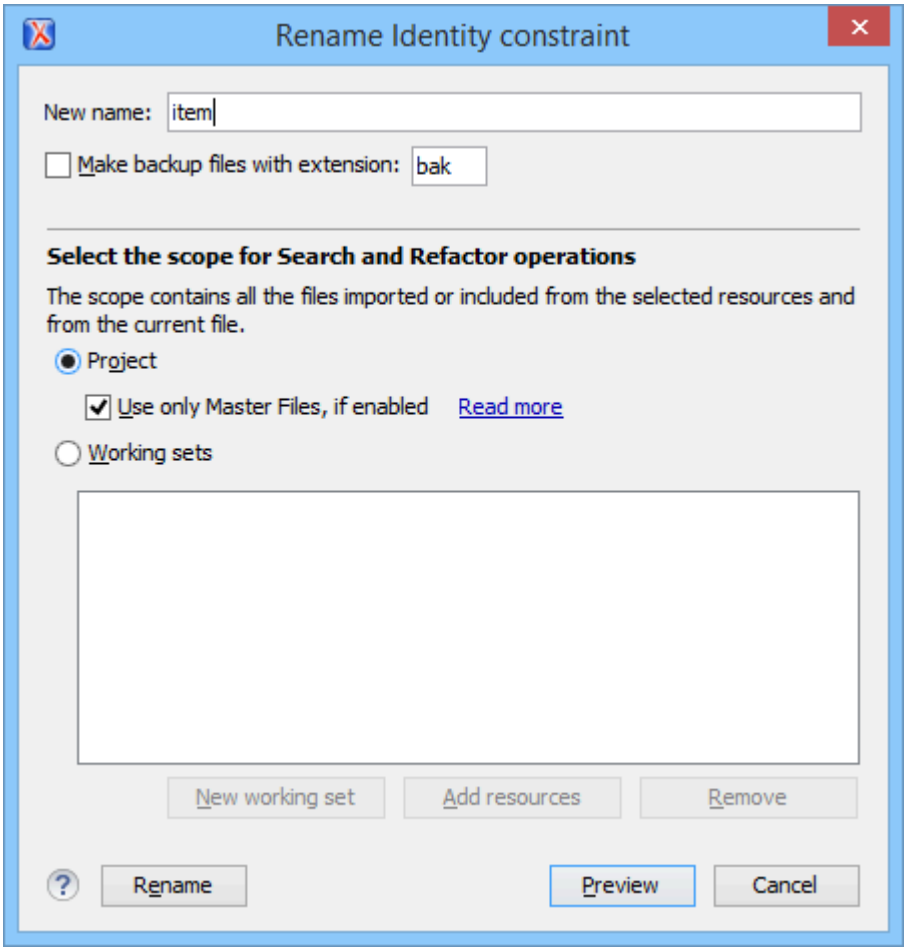
Rename Component

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.

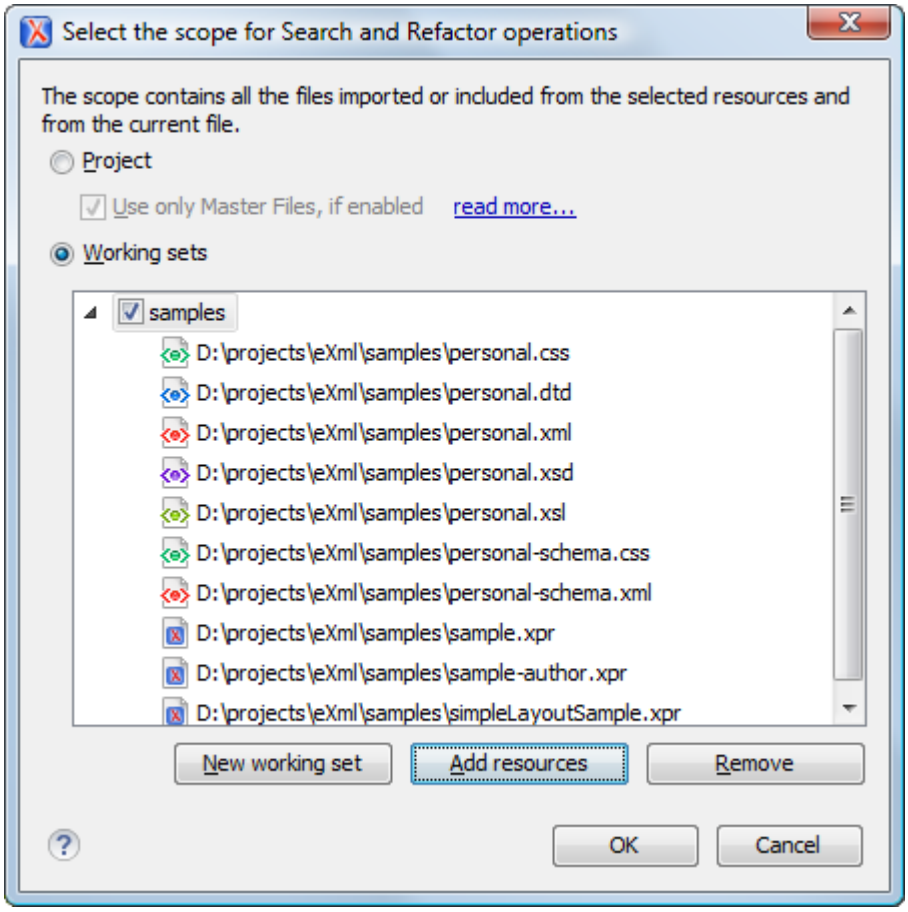
Figure 271. Rename Identity Constraint Dialog Box



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 **Referenced/ Dependent Resources** view's toolbar. You can restrict the scope to the current project or to one or multiple *working sets (on page 2309)*. 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 350)*.

Figure 272. Change Scope Dialog Box

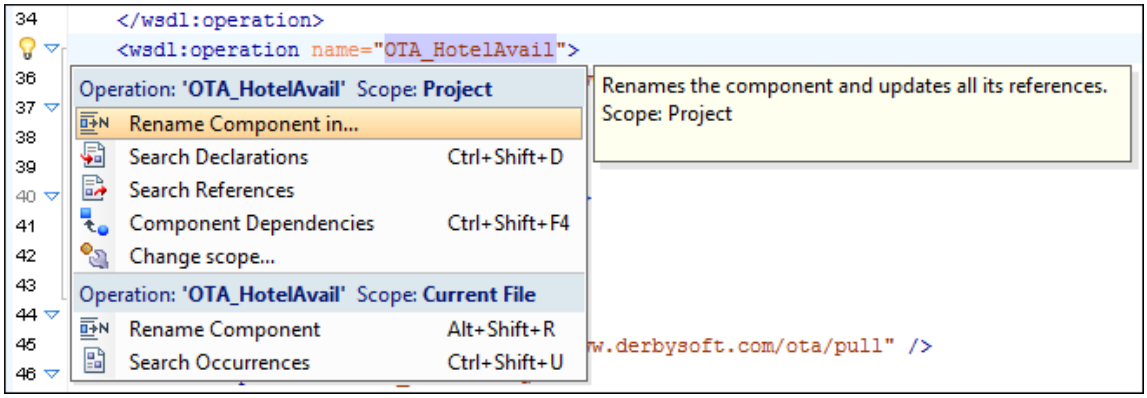


The scope you define is applied to all future search and refactor operations until you modify it. Contextual menu actions allow you to add or delete files, folders, and other resources to the *working set (on page 2309)* structure.

Quick Assist Support in WSDL Documents

The *Quick Assist* feature *(on page 2308)* 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.

Figure 273. WSDL Quick Assist Support



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.

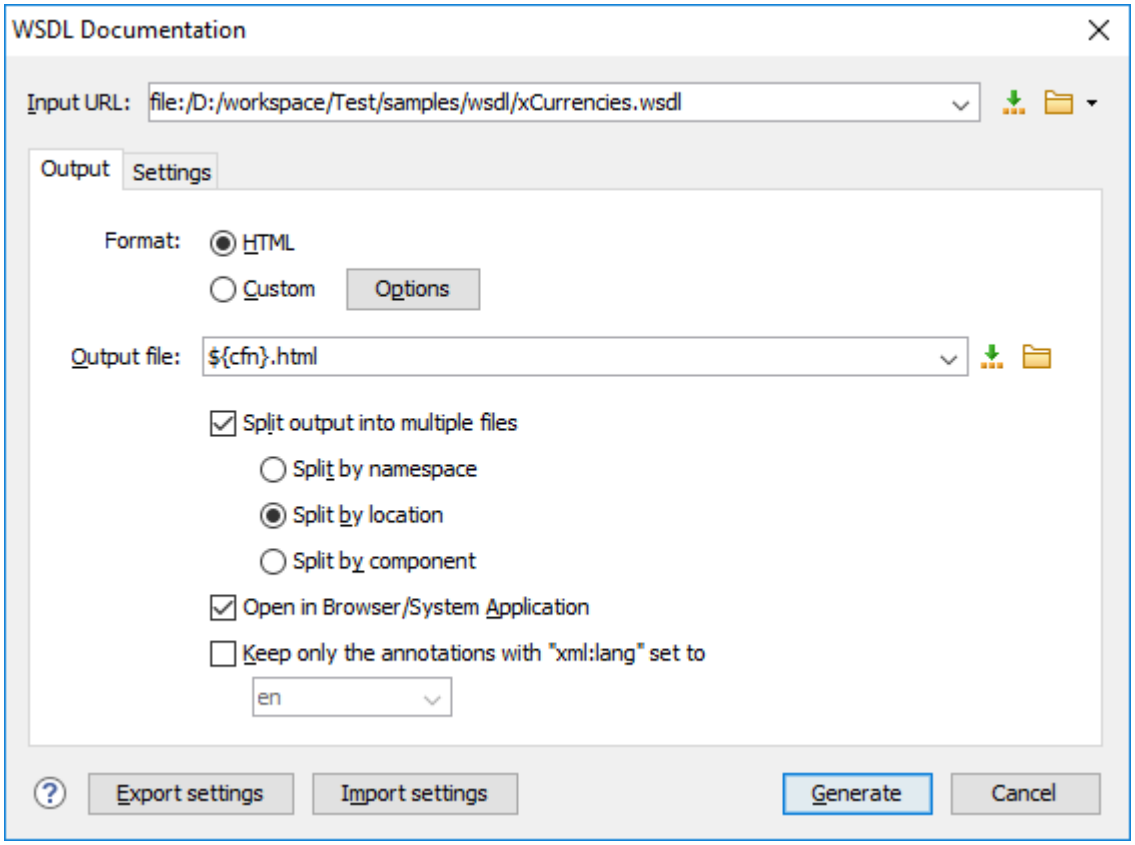
Generating Documentation for WSDL Documents (Deprecated)

You can use Oxygen XML Developer 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 813)* 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 334)*. You can also open the tool by using the **Generate Documentation** toolbar button.

Figure 274. 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 812)*.
  - **Custom** - The documentation is generated in a custom output format *(on page 813)*, 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.

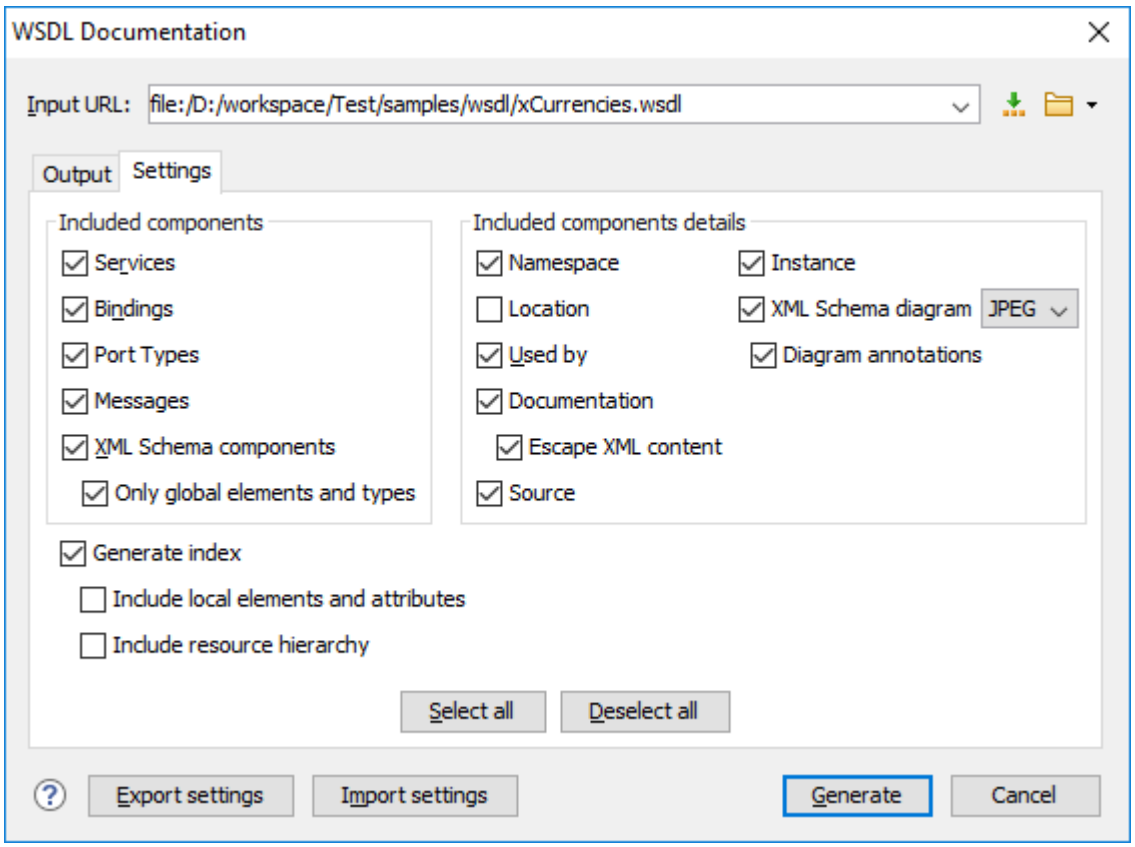
> **Note:**
> To set the browser or system application that will be used, open the **Preferences** dialog box **(Options > Preferences)** *(on page 79)*, go to **Global**, and set it in the **Default Internet browser** field. This will take precedence over the default system application settings.

- **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 275. 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.

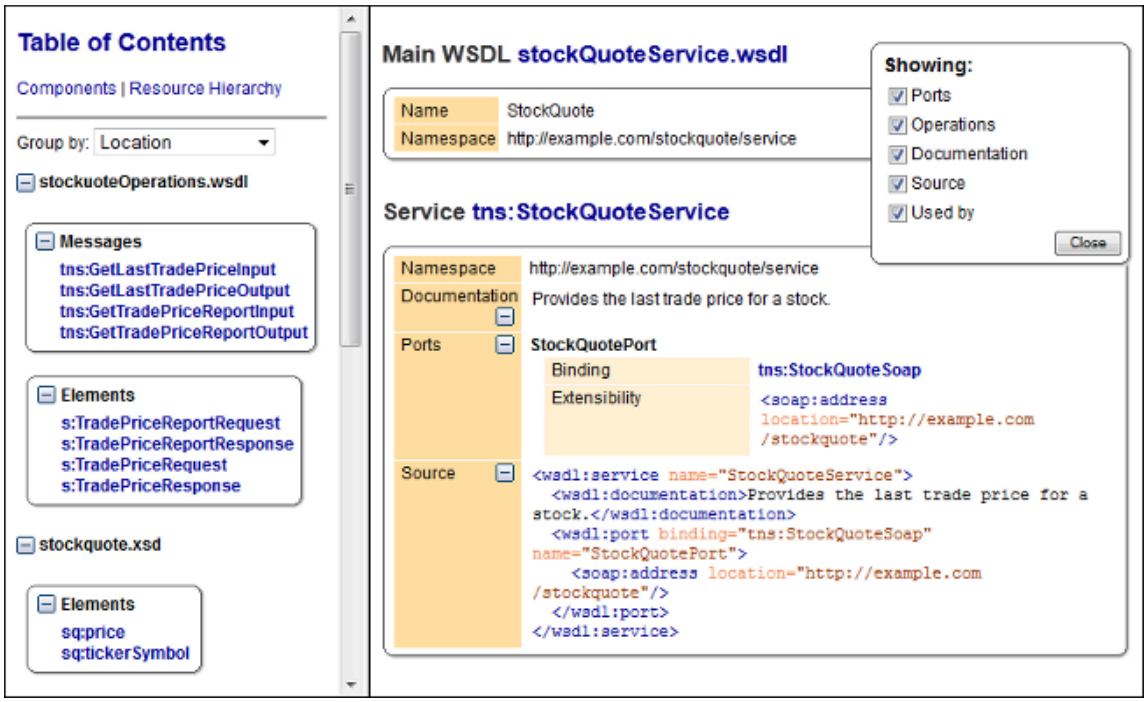
Generate - Use this button to generate the WSDL documentation.

Tip:
This function can be executed from an automated command-line script, for more details, see Scripting Oxygen *(on page 2263)*.

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 276. WSDL Documentation in HTML Format



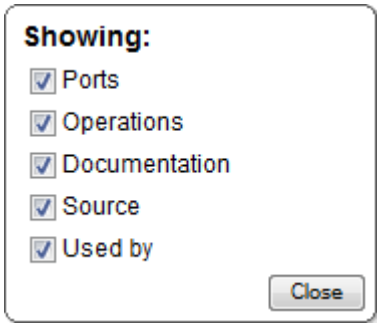
The documentation of each component is presented in a separate section. The title of the section is composed of the component type and the component name. The component information (namespace, documentation, etc.) is presented in a tabular form.

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.

Figure 277. Showing Options



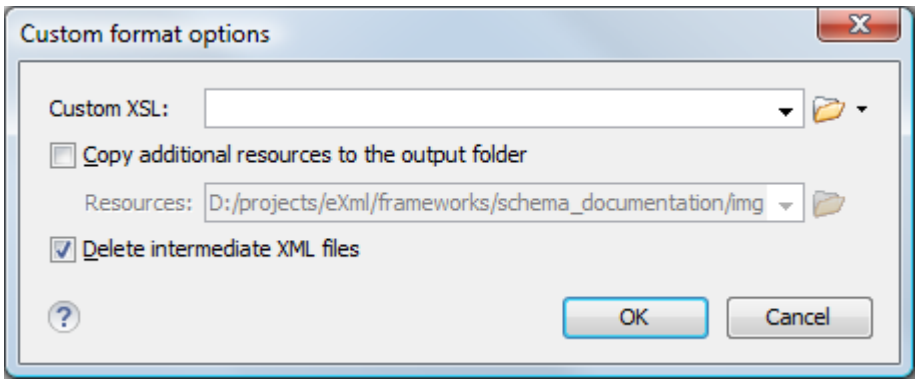
Generating WSDL Documentation in a Custom Format

To obtain the default HTML documentation output from a WSDL document, Oxygen XML Developer 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.

Figure 278. Custom Format Options Dialog Box



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 (Deprecated)

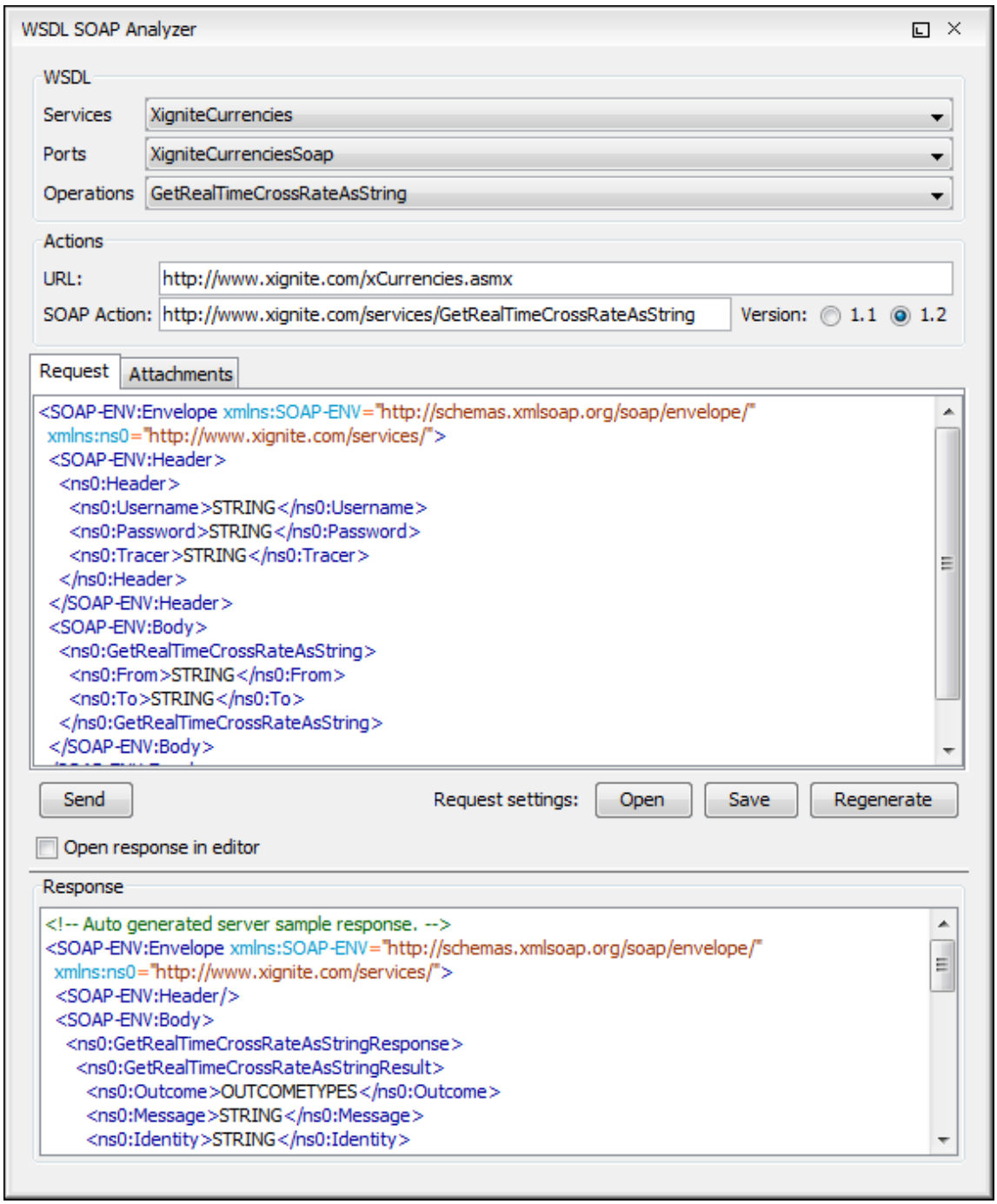
WSDL SOAP Analyzer is a tool that helps you test if the messages defined in a Web Service Descriptor (WSDL) are accepted by a Web Services server.

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 Developer 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 334)* view.

Figure 279. WSDL SOAP Analyzer Dialog Box



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 Developer 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 2303)* 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 Developer 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 Developer 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 Developer 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 816)* 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 813)*. 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 Developer 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 299)*.
- **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 Developer 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.

Validating CSS Stylesheets

Oxygen XML Developer includes a built-in *CSS Validator*, integrated with general validation support. This makes the usual validation features for presenting errors also available for CSS stylesheets.

When you edit a CSS document, you can access the CSS validator options *(on page 168)* 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 168)*. The **CSS 3 with Oxygen extensions** profile includes all the CSS 3 standard properties and the CSS extensions specific for **Oxygen**. That means all **Oxygen**-specific extensions are accepted in a CSS stylesheet by the built-in CSS validator *(on page 817)* 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 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">
```

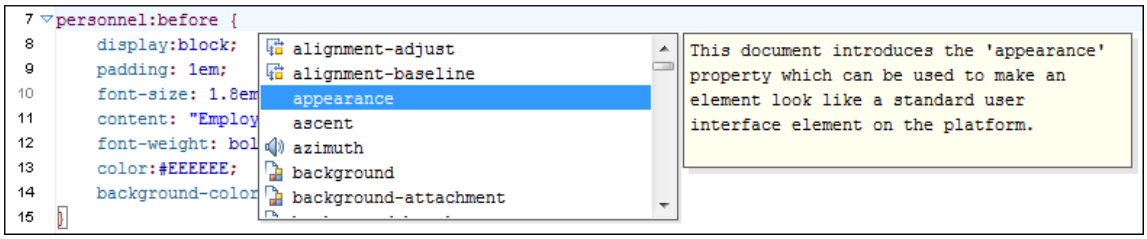
```
      <summary>Description for custom property.</summary>
        <value name="customValue"/>
        <value name="anotherCustomValue"/>
      </property>
</css_keywords>
```

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 2303)*, similar to the one available for XML documents *(on page 457)*, offers the CSS properties and the values available for each property. It can be manually activated with the **Ctrl + Space** 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 462)* into CSS stylesheets. The code templates that are proposed include form controls, actions, and **Author** mode operations.

Figure 280. Content Completion in CSS Stylesheets



The properties and values available are dependent on the CSS Profile selected in the **CSS** preferences *(on page 168)*. 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.

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 817)*

Syntax Highlighting in CSS Files

Oxygen XML Developer 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 79)*.
2. Go to **Editor > Syntax Highlight** *(on page 158)*.
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 158)*

CSS Outline View

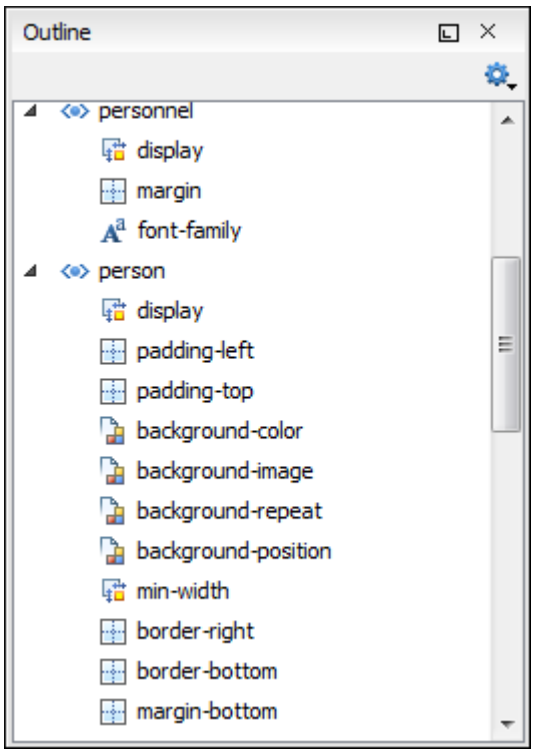
The **Outline** view for CSS stylesheets presents the import declarations for other CSS stylesheet files and all the selectors defined in the current CSS document. The selector entries can be presented as follows:

- 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 Developer 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 281. 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 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 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 Developer 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 Developer 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 Developer 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 Developer 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 299)*.
- **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 Developer 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 Developer includes a built-in *LESS CSS Validator*, integrated with general validation support. The usual validation features for presenting errors also available for LESS stylesheets.

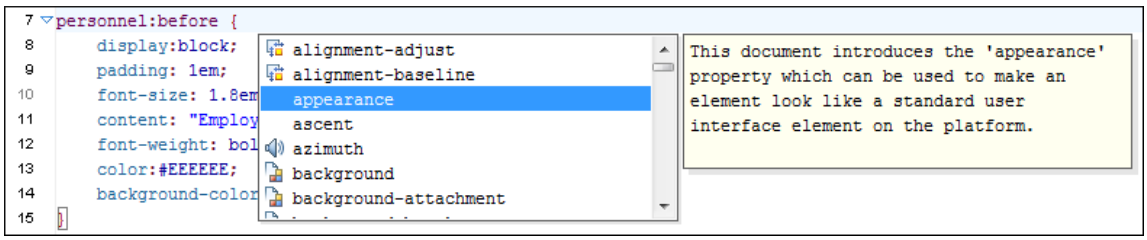
Oxygen XML Developer provides three validation methods:

- 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 2303)* offers the LESS properties and the values available for each property. It can be manually activated with the **Ctrl + Space** 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 462)* into LESS stylesheets. The code templates that are proposed include form controls, actions, and **Author** mode operations.

Figure 282. Content Completion in LESS Stylesheets



The properties and values available are dependent on the CSS Profile selected in the CSS preferences *(on page 168)*.

Syntax Highlighting in LESS Files

Oxygen XML Developer 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 79)*.
2. Go to **Editor > Syntax Highlight** *(on page 158)*.

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 158)*

Compiling LESS Stylesheets to CSS

When editing LESS files, you can compile the files into CSS. Oxygen XML Developer 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:

```
@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 135)* (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 Developer offers support for editing Relax NG schema files in the following editing modes:

- **Text editing mode** *(on page 731)* - 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 825)* and **Logical Model View** *(on page 826)*.
- **Grid editing mode** *(on page 279)* - Displays Relax NG schema files in a structured spreadsheet-like grid.

For information about applying and detecting schemas, see Associating a Schema to XML Documents *(on page 554)*.

Related Information:
Associating a Schema to XML Documents *(on page 554)*

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 Developer 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 cat set a main Relax NG document either using the *main files* support from the **Project** view *(on page 350)*, 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 Developer 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 2306)* 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 526)*
XML Schema Outline View *(on page 735)*

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 Developer 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.

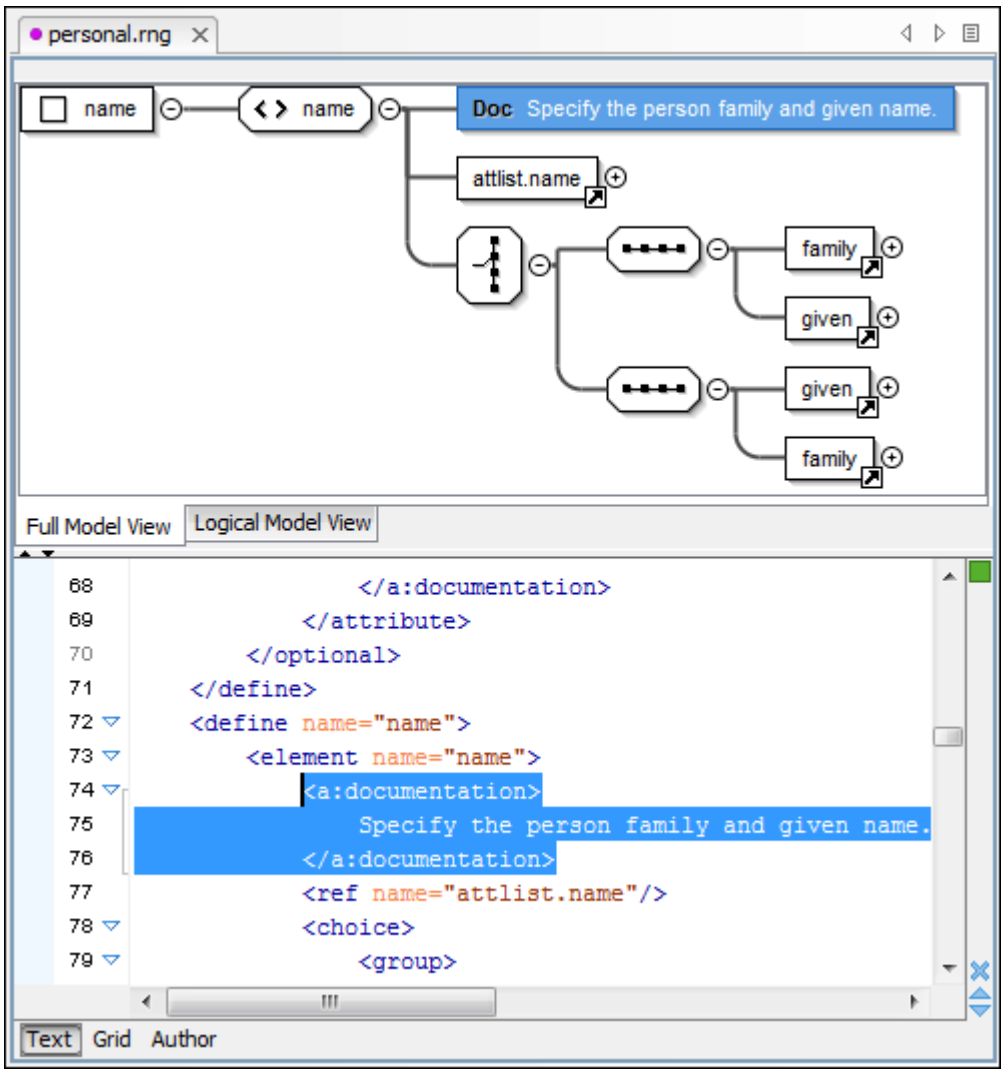
Oxygen XML Developer provides a side-by-side source and diagram presentation with real-time synchronization:

- 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 826)*.

Figure 283. Relax NG Schema Editor - Full Model View



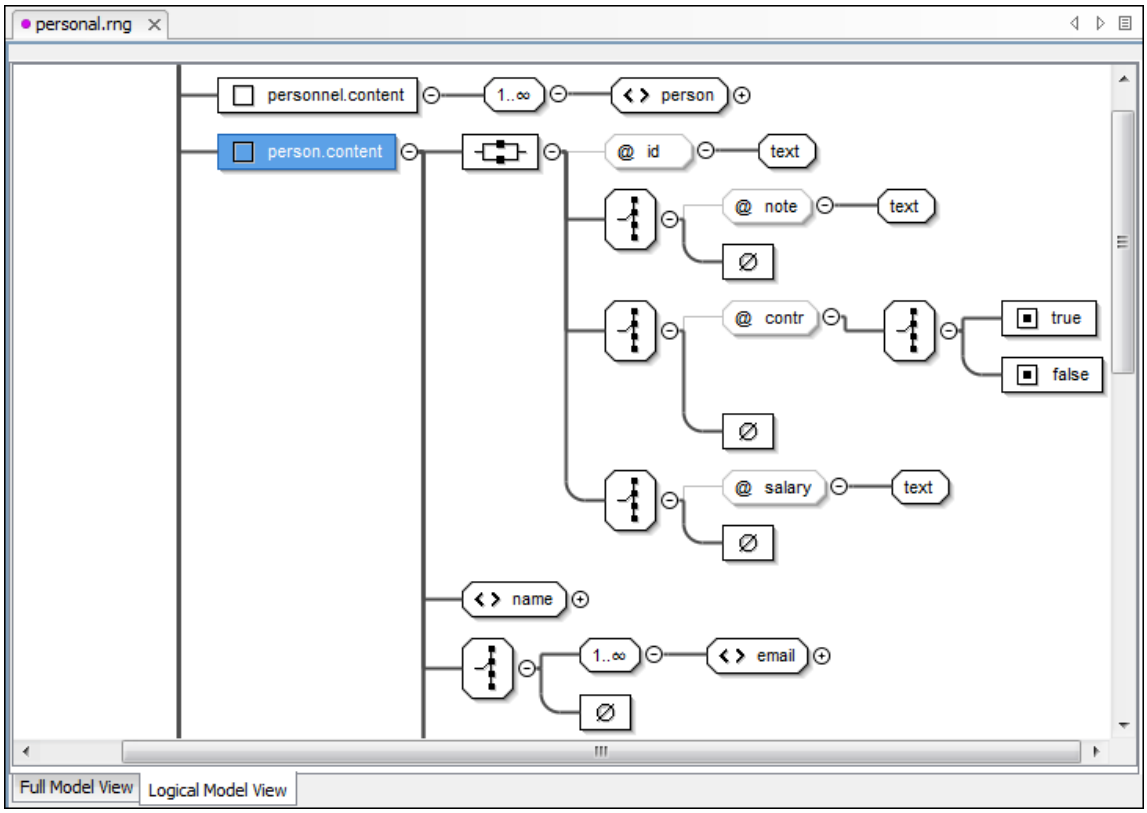
The following references can be expanded in place: patterns, includes, and external references. This expansion mechanism, coupled with the synchronization support, makes the schema navigation easy.

All the element and attribute names are editable by double-clicking the names.

Logical Model View

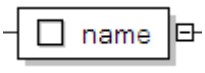
The **Logical Model View** presents the compiled schema in the form of a single pattern. The patterns that form the element content are defined as top-level patterns with generated names. These names are generated depending of the elements name class.

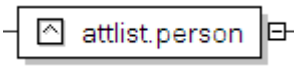
Figure 284. Logical Model View for a Relax NG Schema



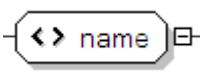
Symbols Used in the Schema Diagram

The views in the schema diagram editor renders all the Relax NG schema patterns with the following intuitive symbols:

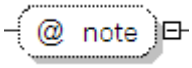
- 

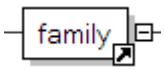
- `define` pattern with the `@name` attribute set to the value shown inside the rectangle (in this example `name`).
- 

- `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`).
- 

- `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`).
- 

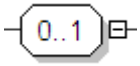
- `element` pattern with the `@name` attribute set to the value shown inside the rectangle (in this example `name`).

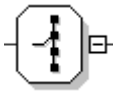
- 

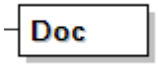
- `attribute` pattern with the `@name` attribute set to the value shown inside de rectangle (in this case `note`).
- 

- `ref` pattern with the `@name` attribute set to the value shown inside the rectangle (in this case `family`).
- 

- `oneOrMore` pattern.
- 

- `zeroOrMore` pattern.
- 

- `optional` pattern.
- 
- `choice` pattern.
- 

- `value` pattern (for example, used inside a `choice` pattern).
- 
- `group` pattern.
- 

- 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` pattern.
- 

- `empty` pattern.

Actions Available in the Schema Diagram Editor

When editing Relax NG schemas in **Full Model View** *(on page 825)*, 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 826)* 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 79)*, go to **Editor > Document Checking**, and deselect the **Enable automatic validation** option *(on page 160)*.

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 Developer 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.

Related Information:

Validating XML Documents Against a Schema *(on page 516)*
Embedding Schematron Rules in XML Schema or RELAX NG *(on page 967)*
Validation Scenario *(on page 525)*
Associating a Schema to XML Documents *(on page 554)*
Presenting Validation Errors in Text Mode *(on page 519)*

Content Completion in Relax NG Schemas

The intelligent *Content Completion Assistant (on page 2303)* 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 79)*, go to **Editor > Content Completion**, and deselect the **Enable content completion** option *(on page 144)*.

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** 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.

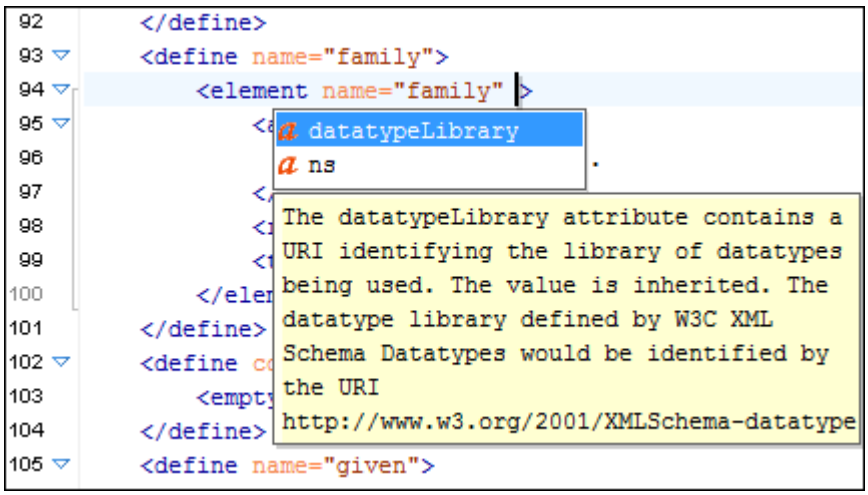
To insert the selected proposal in **Text** mode, do one of the following:

- 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 2306)* 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 Developer, see Defining Main Files at Project Level *(on page 350)*.

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 285. Relax NG Content Completion Assistant



Syntax Highlighting in Relax NG Schemas

Oxygen XML Developer 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 79)*.
2. Go to **Editor > Syntax Highlight** *(on page 158)*.
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 Developer 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 159)*.

Related Information:
Syntax Highlight Preferences *(on page 158)*

Quick Fixes for DTD, XSD, and Relax NG Errors

Oxygen XML Developer offers *Quick Fixes (on page 2308)* 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 .

Oxygen XML Developer provides *Quick Fixes* for numerous types of problems, including the following:

Problem Type	Available Quick Fixes
A specific element is required in the current context	Insert the required element
An element is invalid in the current context	Remove the invalid element
The content of the element should be empty	Remove the element content
An element is not allowed to have child elements	Remove all child elements
Text is not allowed in the current element	Remove the text content
A required attribute is missing	Insert the required attribute
An attribute is not allowed to be set for the current element	Remove the attribute
The attribute value is invalid	Propose the correct attribute values
ID value is already defined	Generate a unique ID value
References to an invalid ID	Change the reference to an already defined ID

Related Information:
Schematron Quick Fixes (SQF) *(on page 553)*
Ignoring/Unignoring Validation Problems *(on page 549)*

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 825)* and **Logical Model View** *(on page 826)* 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.

Figure 286. Relax NG Outline View



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 235)*.

Configure displayed attributes

Displays the **XML Structured Outline** preferences page *(on page 235)*.

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 837)* 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.

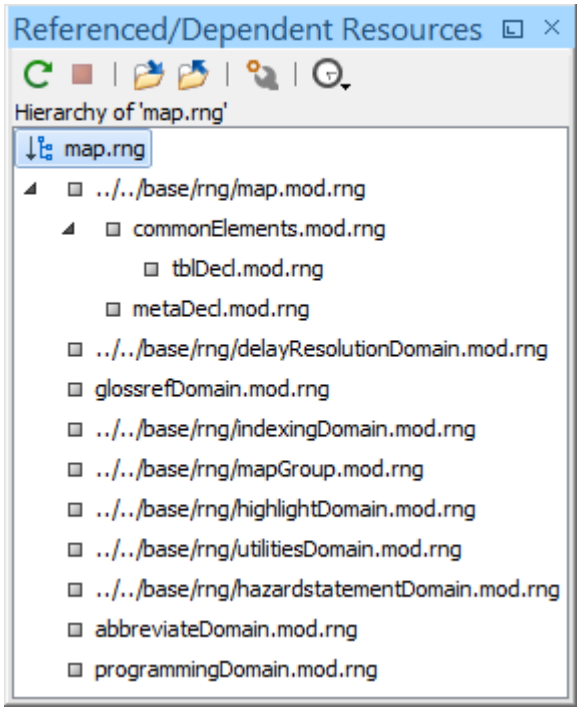
The upper part of the **Outline** view contains a filter box that allows you to focus on the relevant components. Type a text fragment in the filter box and only the components that match it are presented. For advanced usage you can use wildcard characters (such as `*` or `?`) and separate multiple patterns with commas.

RNG Referenced/Dependent Resources View

The **Referenced/Dependent Resources** view displays the references 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 334)* and choose **Show referenced resources** or **Show dependent resources** from the contextual menu.

Figure 287. Referenced View



The following actions are available on the toolbar of the **Referenced/Dependent Resources** view:

Refresh

Refreshes the resource structure.

Stop

Stops the computing.

Show hierarchy for

Computes the hierarchical structure of the references for a resource.

Show dependencies for

Computes the structure of the dependencies for a resource.

Configure dependencies search scope

Allows you to configure a scope to compute the dependencies. 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 **Referenced/Dependent Resources** view contains the following actions:

Open

Opens the resource. You can also double-click a resource within the hierarchical 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.

Show references resources

Shows the references for the selected resource.

Show dependent resources

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 hierarchical structure.

Collapse All

Collapses all children of the selected resource from the hierarchical structure.



Tip:
When a recursive reference is encountered in the 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 802)*.

Related Information:
Modular Contextual XML Editing Using 'Main Files' Support *(on page 567)*
Search and Refactor Operations Scope *(on page 569)*

Moving/Renaming RNG Resources

You can move and rename a resource presented in the **Referenced/Dependent Resources** 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 **Referenced/Dependent Resources** 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 **Referenced/Dependent Resources** 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.

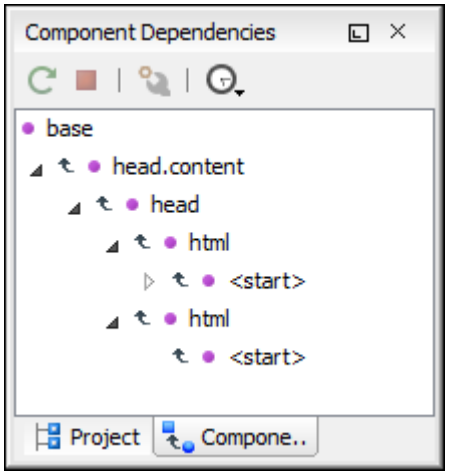
To see the dependencies of a Relax NG component:

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  .

Figure 288. Component Dependencies View



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 569)*

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

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 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:

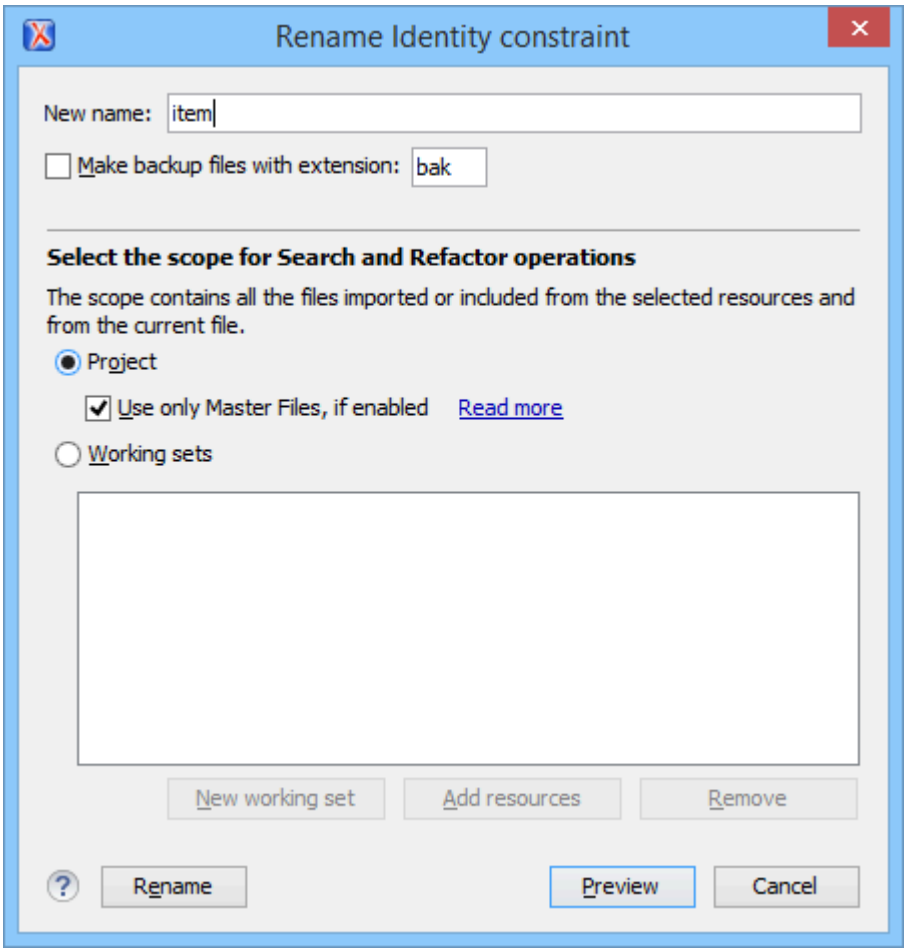
Rename Component

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 289. Rename Identity Constraint Dialog Box

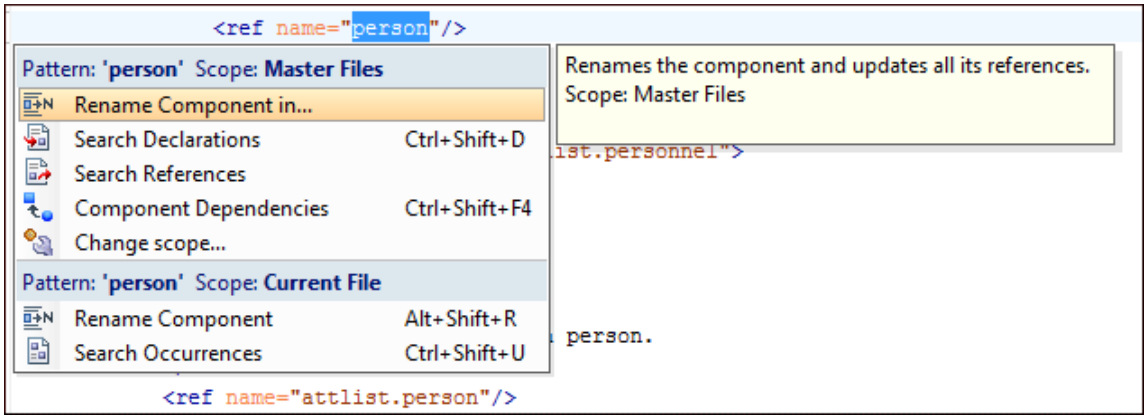


RNG Quick Assist Support

The *Quick Assist* support *(on page 2308)* 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 2308)* 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.

Figure 290. RNG Quick Assist Support



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 837)*
Referenced/Dependent Resources View *(on page 834)*
Searching and Refactoring Actions *(on page 838)*
Search and Refactor Operations Scope *(on page 569)*

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 pluggable datatypes specification.

The *JAR (on page 2306)* 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 Developer.

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 Developer offers support for editing NVDL schema files in the following editing modes:

- **Text editing mode *(on page 731)*** - 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 843)* and **Logical Model View** *(on page 844)*.
- **Grid editing mode *(on page 279)*** - Displays NVDL schema files in a structured spreadsheet-like grid.

For information about applying and detecting schemas, see Associating a Schema to XML Documents *(on page 554)*.

Related Information:
Associating a Schema to XML Documents *(on page 554)*

NVDL Schema Diagram

This section explains how to use the graphical diagram of a NVDL schema.

Introduction to NVDL Schema Diagram Editor

Oxygen XML Developer 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.

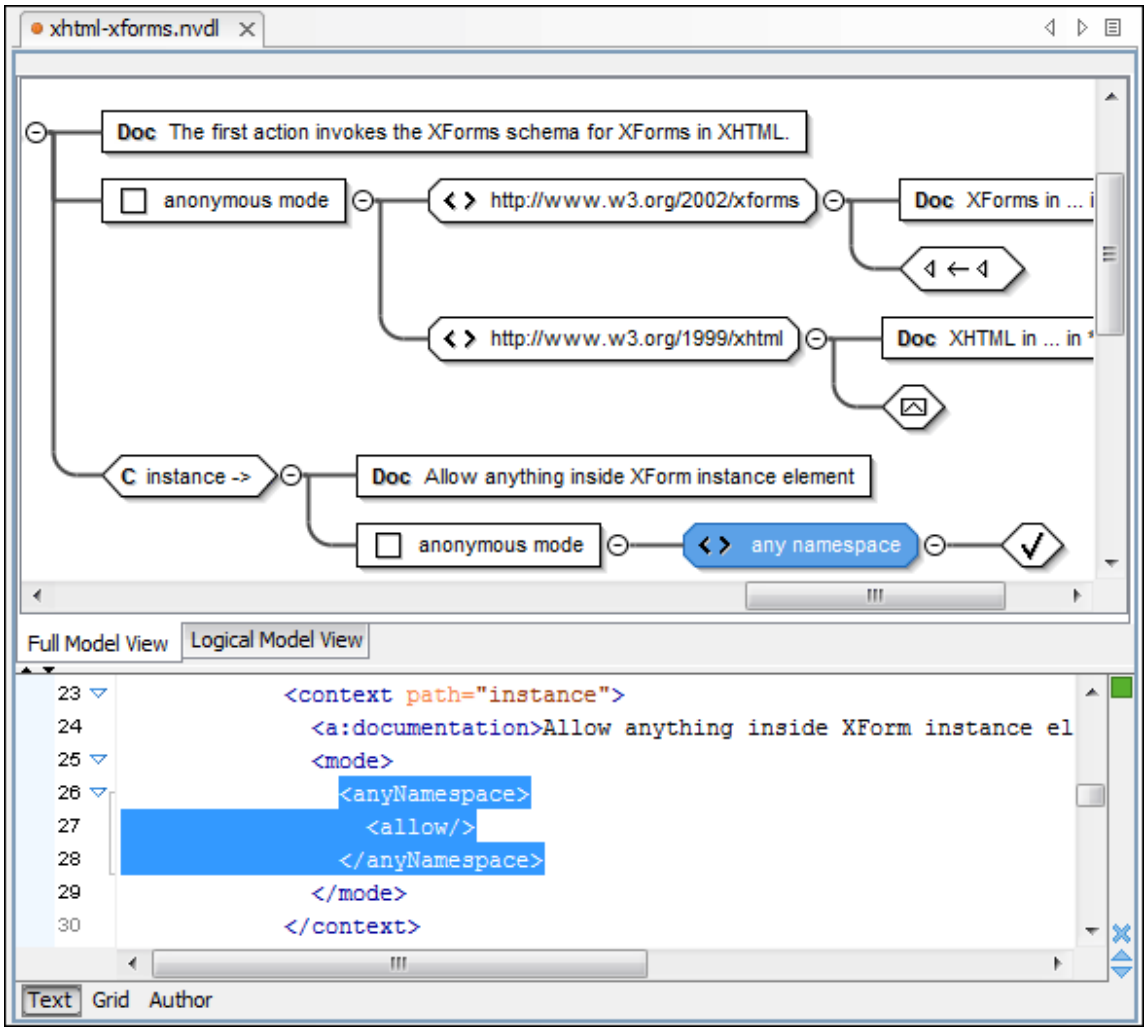
Oxygen XML Developer provides a side-by-side source and diagram presentation with real-time synchronization:

- 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 844)*.

Figure 291. NVDL Schema Editor - Full Model View



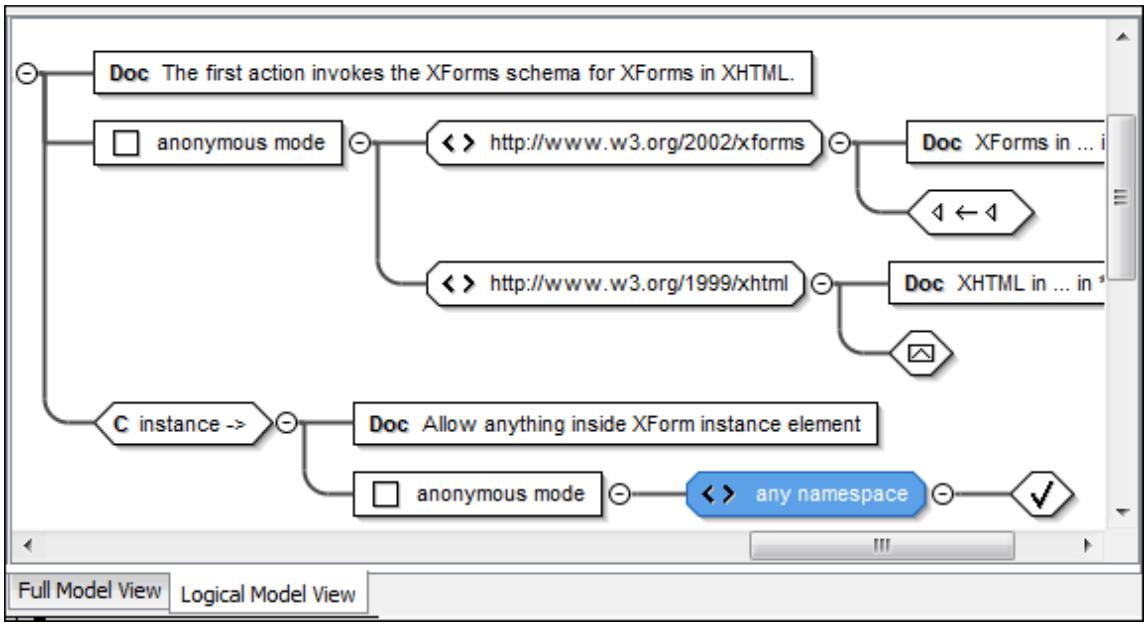
The **Full Model View** renders all the NVDL elements with intuitive icons. This representation coupled with the synchronization support makes the schema navigation easy.

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 of the elements name class.

Figure 292. 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 844)* 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 79)*, go to **Editor > Document Checking**, and deselect the **Enable automatic validation** option *(on page 160)*.

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 Developer 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 516)*
Presenting Validation Errors in Text Mode *(on page 519)*

Content Completion in NVDL Schemas

The intelligent *Content Completion Assistant (on page 2303)* 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 79)*, go to **Editor > Content Completion**, and deselect the **Enable content completion** option *(on page 144)*.

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** 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.

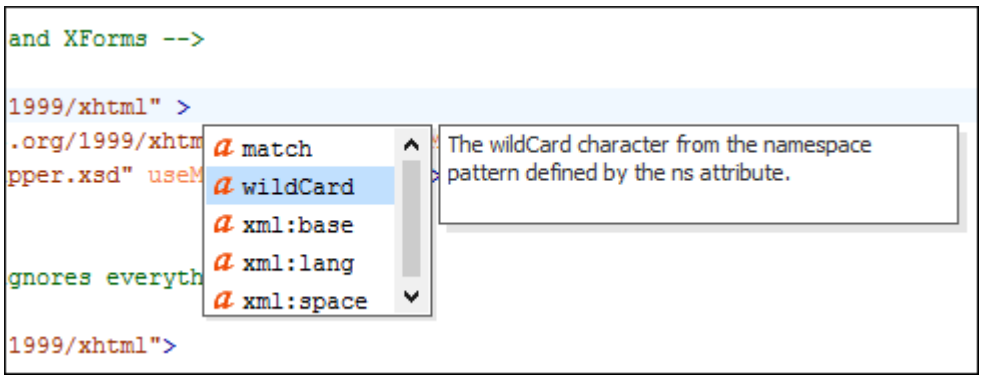
To insert the selected proposal in **Text** mode, do one of the following:

- 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 2306)* 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 Developer, see Defining Main Files at Project Level *(on page 350)*.

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 293. NVDL Content Completion Assistant



Syntax Highlighting in NVDL Schemas

Oxygen XML Developer 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 79)*.
2. Go to **Editor > Syntax Highlight** *(on page 158)*.
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 Developer 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 159)*.

Related Information:
Syntax Highlight Preferences *(on page 158)*

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.

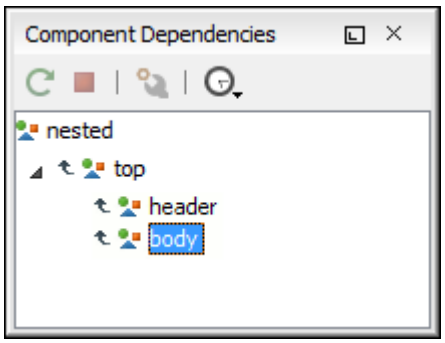
To see the dependencies of an NVDL component:

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 .

Figure 294. Component Dependencies View



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

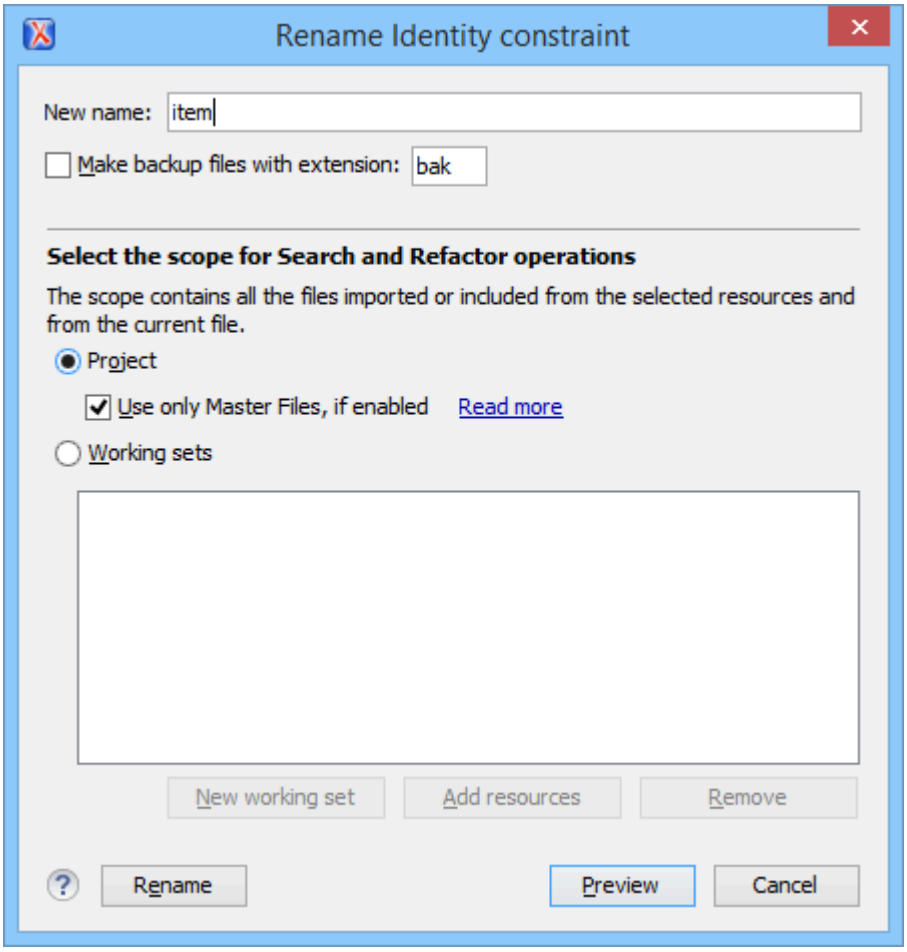
Allows you to rename the current component (in-place). The component and all its references in the document are highlighted with a thin border and the changes you make to the component at the cursor position are updated in real time to all occurrences of the component. To exit the in-place editing, press 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.

Figure 295. Rename Identity Constraint Dialog Box



Editing JSON Documents

This section explains the features of the Oxygen XML Developer and how to use them.

Resources

For more information about the JSON support in Oxygen XML Developer, see the following resources:

- **Video: JSON Editing**
- **Video: JSON Tools in Oxygen**
- **Webinar: JSON and JSON Schema Support in Oxygen**
- **Webinar: Introducing Oxygen JSON Editor - The Ultimate Solution for JSON Editing**

JSON Editor

Oxygen XML Developer 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 299)*.

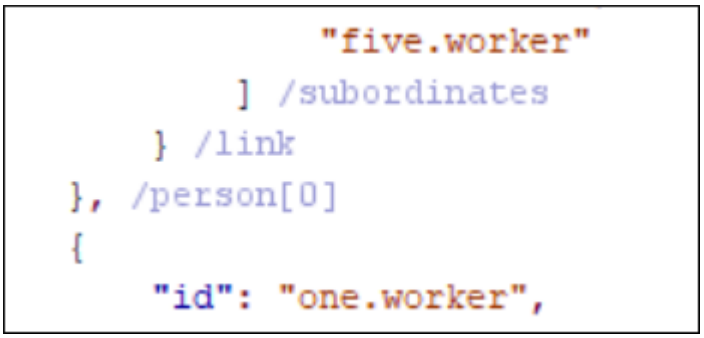
Tip:
You can experiment with a sample of a JSON file available at: `[OXYGEN-INSTALL-DIR]/samples/json/personal.json`.

Text Mode Editor

When editing JSON documents in the **Text** editing mode, the usual text editing actions are available, along with other editor-specific actions, including:

- Search and Find/Replace *(on page 354)*
- Drag and Drop
- Validation *(on page 855)*
- Format and Indent (Pretty Print)

The JSON Text mode editor also offers unique features. For example, the property name is displayed after the ending bracket:



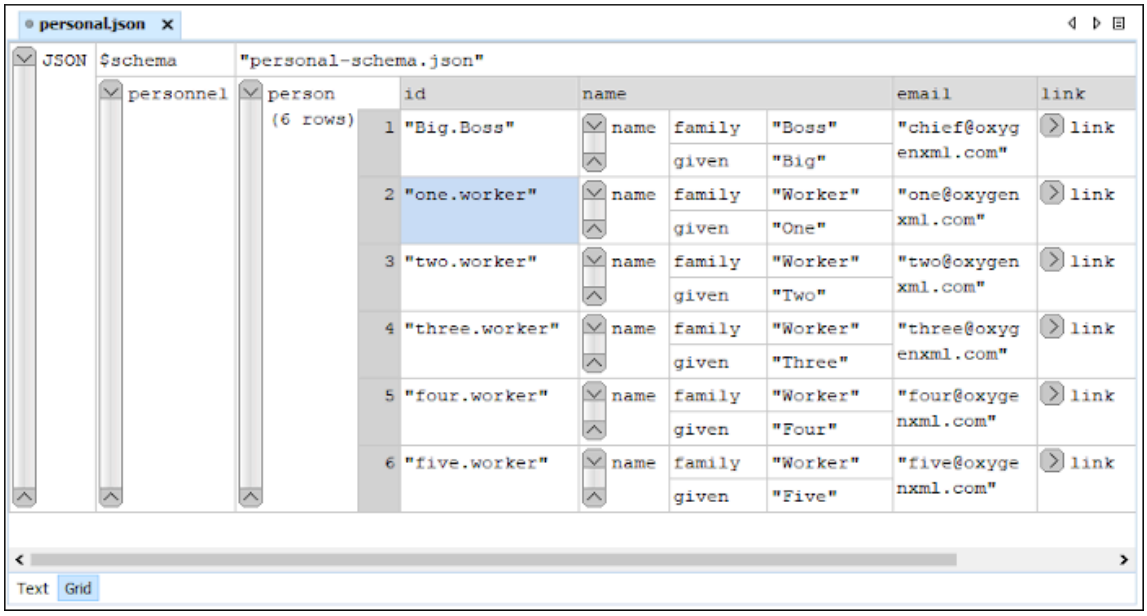
This default behavior of displaying the property name after the ending bracket can be disabled by deselecting the **Show property name after ending bracket** option *(on page 128)* in the **Editor > Edit Modes > Text** preferences page. The option also has a slider that you can use to adjust the transparency level.

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 Developer allows you to view and edit the JSON documents in the **Grid** mode. 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 296. 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 130)* 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.

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* reference or hyperlink will changes the reference to a clickable link.

JSON Schema References

Referencing allows you to create modular, maintainable, and reusable schemas. It is particularly useful when you have multiple instances of a similar structure in your data and want to enforce consistency in validation rules.

A schema can reference another schema using the `$ref` (or `$dynamicRef`) keyword. Its value is a URI-reference that is resolved against the schema's Base URI. When evaluating a `$ref`, an implementation uses the resolved identifier to retrieve the referenced schema and applies that schema to the instance. For more details about structuring JSON schemas, see Understanding JSON Schema: Structuring a complex schema.

- **Defining Reusable Schemas** - You can define reusable schema components using the `definitions` (or `$defs` for latest drafts) keyword. These definitions act as named schemas that you can reference using `$ref` (or `$dynamicRef`).

```
{
  "$defs": {
    "person": {
      "type": "object",
      "properties": {
        "name": { "type": "string" },
        "age": { "type": "integer" }
      }
    }
  }
}
(my_schema.json)
```

- **Referencing a Schema** - To reference a schema defined in a definitions block, use the `$ref` keyword followed by the JSON Pointer of the definition.

```
{
  ...
  "type": "object",
  "properties": {
    "person1": { "$ref": "#/$defs/person" },
    "person2": { "$ref": "#/$defs/person" }
  }
}
```

  In this example, both `"person1"` and `"person2"` properties refer to the `"person"` schema defined in `$defs`.

- **External References** - You can also reference schemas from external files using their URI. This can be useful when you have multiple schema files.

```
{
  "$ref": "my_schema.json#/$defs/person"
}
```

  Here, the schema at the specified URI is being referenced. You can also use `$id` to uniquely identify a schema resource and then reference it from another file using the same mechanism. The following example sets a custom id for the `"person"` schema:

```
{
  "$defs": {
    "person": {
      "$id": "#person_info"
      "type": "object",
      "properties": {
        "name": { "type": "string" },
        "age": { "type": "integer" }
      }
    }
  }
}
```

You can now reference this definition by its `$id`, not by its JSON Pointer:

```
{
  "$ref": "my_schema.json/#person_info"
}
```



Note:
Oxygen XML Developer allows `$id` only as an URI fragment, not as a relative pointer.

- **Combining References with Other Keywords** - You can use `$ref` in combination with other keywords. For instance, you might reference a schema within the `items` keyword to define an array of a specific type:

```
{
  ...
  "type": "array",
  "items": { "$ref": "#/$defs/person" }
}
```

Starting with latest drafts, you can use `$ref` in conjunction with other type-specific keywords for stricter validation. For example, the previous schema can be modified to not allow extraneous properties for a "person" object:

```
{
  ...
  "type": "array",
  "items": {
      "$ref": "#/$defs/person",
      "additionalProperties": false
  }
}
```

Validating JSON Documents

Oxygen XML Developer 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.

Resources

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 Developer 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 334)*.

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

```
{"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 Developer, how validation errors are presented, and information about built-in validation scenarios.

Oxygen XML Developer 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 857)*.

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 JSON Schema Through a Validation Scenario *(on page 865)*.

Automatic Validation

By default, Oxygen XML Developer is configured to automatically mark validation errors in the JSON document as you are editing. The **Enable automatic validation** option *(on page 160)* in the **Document Checking** preferences page *(on page 160)* 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 161)* 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 857)*
Presenting Validation Errors in JSON Documents *(on page 857)*

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 Developer.

Manual Validation Actions

To manually validate the currently edited document, use one of the following actions:

Validate

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 334)*.

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 schema for validating the current document *(on page 866)*.

> **Note:**
> The **Validate with** action does not work for files loaded through a custom protocol plugin *(on page 1873)* developed independently and added to Oxygen XML Developer 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 334)*. 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 160)*.

Related Information:

Automatic Validation *(on page 856)*

Presenting Validation Errors in JSON Documents *(on page 857)*

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 160)*.

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 79)*, go to **Editor > Document checking**, and specify the desired limit in the **Maximum number of validation highlights** option *(on page 160)*.

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 160)*.

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. 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 160)* 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 474)* view, 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 865)*. Oxygen XML Developer 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:

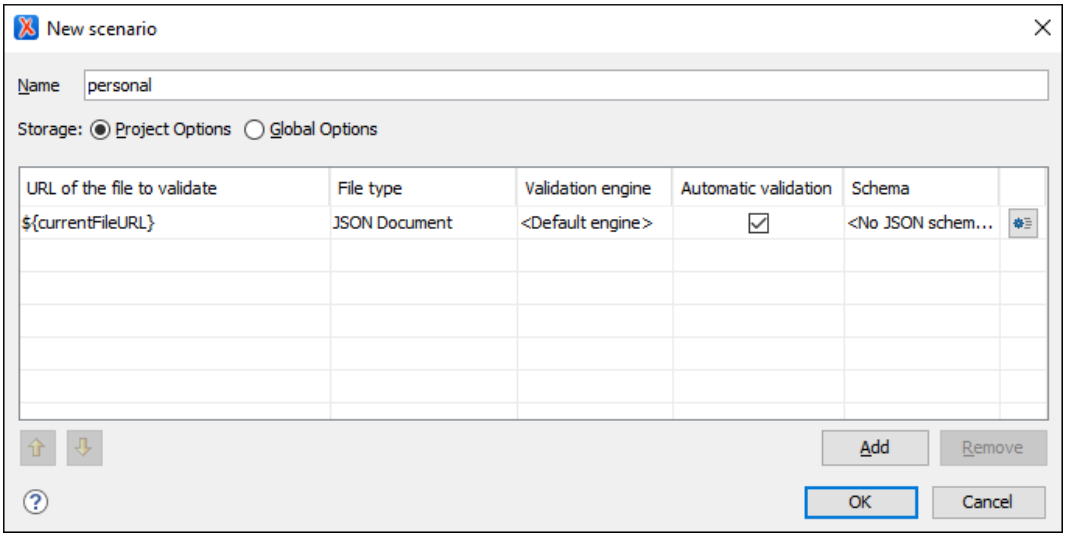
1. Select the  **Configure Validation Scenario(s)** action in one of the following ways:
   - From the  **Validation** toolbar drop-down menu.
   - From the **Document > Validate** menu.
   - From the **Validate** submenu, when invoking the contextual menu on a file in the **Project** view *(on page 334)*.

   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.

Figure 297. 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

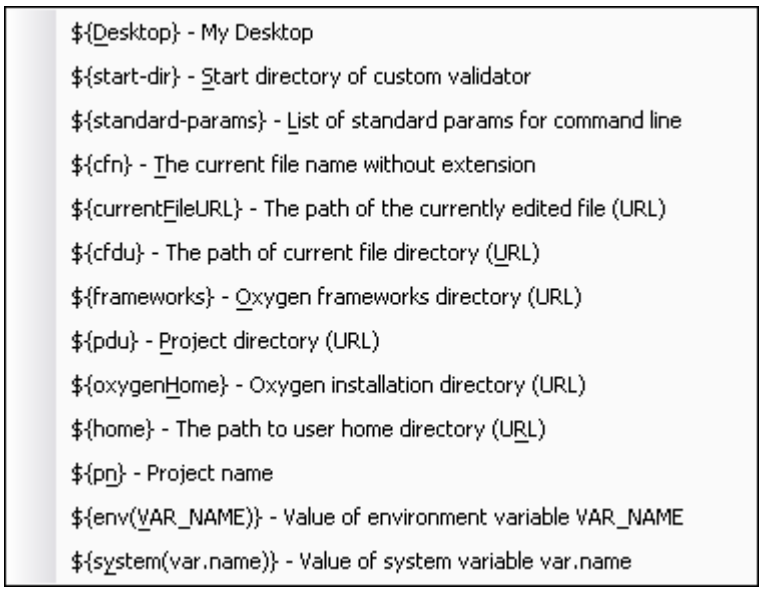
You can choose between storing the scenario in the **Project Options** *(on page 2308)* or **Global Options** *(on page 2305)*.

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 252)* or a custom editor variable *(on page 261)*.

Figure 298. Insert an Editor Variable



File type

The type of the document that is validated in the current validation unit. Oxygen XML Developer 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 validation 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:

```
<sch:schema xmlns:sch="http://purl.oclc.org/dsdl/schematron" queryBinding="xslt2">
```

Automatic validation

If this option is selected, the validation operation defined by this row is also applied by the automatic validation feature. If the **Automatic validation** feature is disabled in the

Document Checking preferences page *(on page 160)*, 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

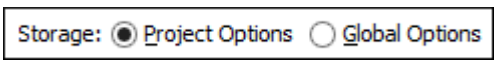
Removes an existing validation unit from the list.

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 2308)* or by exporting them to a specialized scenarios file *(on page 251)* 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 2308)* or **Global Options** *(on page 2305)*.



Selecting **Project Options** *(on page 2308)* 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 2305)* 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 2309)*.

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

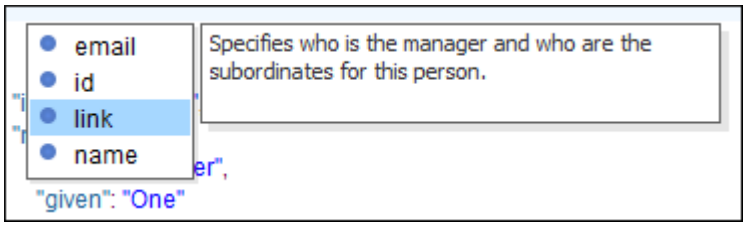
Working with XML Catalogs *(on page 564)*

Content Completion Assistant in JSON

Oxygen XML Developer includes an intelligent *Content Completion Assistant (on page 2303)* 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 79)*, go to **Editor > Content Completion**, and deselect the **Enable content completion** option *(on page 144)*.

Figure 299. Content Completion Assistant in JSON



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 2303)* 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 865)* section.

If a JSON document does not have a schema associated, Oxygen XML Developer automatically learns the document structure as editing events occur and will offer content completion proposals accordingly. Note that this feature is disabled for documents that contain more than one million characters.

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** or **Alt + ForwardSlash (Command + Option + ForwardSlash on macOS)**.

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 864)*. 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 Developer 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 462)*.

Schema Annotations in JSON Content Completion

A schema annotation is a documentation snippet that appears in the *Content Completion Assistant (on page 2303)* offering more information about the current proposal.

This feature is enabled by default, but you can disable it by deselecting the **Show annotations in Content Completion Assistant** *(on page 150)* 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:

```
"idType": {
  "title": "The 'id' property",
  "description": "Specifies a required ID for this person.",
  "type": "string",
  "maxLength": 20
}
```

Associating a Schema to JSON Documents

To provide as-you-type validation and to compute valid proposals for the *Content Completion Assistant (on page 2303)*, Oxygen XML Developer 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 Developer 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 865)* 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 868)*.

> **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 JSON Schema Through a Validation Scenario

Oxygen XML Developer 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 Developer 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 859)*. 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 334)*).
2. Click the **New** button to create a new validation scenario *(on page 859)* 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 859)*.
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 300. 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) or SFTP (or a custom protocol *(on page 1890)*). You can specify the URL by using the text field, the history drop-down, the **Insert Editor Variables** *(on page 252)* 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 **xslt2** after the Schematron namespace declaration:

```
<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 301. 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), SFTP or a custom protocol *(on page 1890)*. You can specify the URL by using the text field, the history drop-down, the **Insert Editor Variables** *(on page 252)* 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 **xslt2** after the Schematron namespace declaration:
>
> ```
> <sch:schema xmlns:sch="http://purl.oclc.org/dsdl/schematron" queryBinding="xslt2">
> ```

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 JSON Schema Directly in JSON Documents

Associate Schema Action

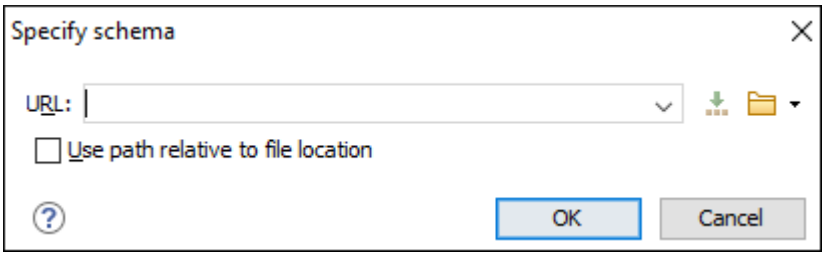
The schema used by the *Content Completion Assistant (on page 2303)* and document validation engine can be associated with the current document by using the **Associate Schema** action. The association can specify a relative file path or a URL of the schema.

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:

   Figure 302. Associate Schema Dialog Box

   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) or SFTP (or a custom protocol *(on page 1890)*).
   - **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.

2. Select the JSON Schema that will be associated with the JSON 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 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).

Associating a JSON Schema in a Framework (Document Type) Configuration

The JSON schema used to compute valid proposals in the *Content Completion Assistant (on page 2303)* and by the document validation engine to report errors and warnings can be defined in each particular *framework (on page 2305)* (document type). This schema will be used only if one is not detected in the current JSON file *(on page 868)*.

To associate a JSON schema in a particular *framework* (document type), follow these steps:

1. Open the **Preferences** dialog box **(Options > Preferences)** *(on page 79)* and go to **Document Type Association**.
2. Select your particular document type and click the **Edit** *(on page 93)*, **Extend** *(on page 94)*, or **Duplicate** *(on page 94)* 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 95)*.
3. Go to the **Schema** tab *(on page 99)*.
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 JSON document.

Learn Document Structure When JSON Schema is not Detected

When there is no JSON schema associated with a JSON document, Oxygen XML Developer can learn the document structure by parsing the document internally to provide an initialization source for content completion and validation.

This feature is controlled by the **Learn on open document** option *(on page 144)* that is available in the **Editor > Content Completion** preferences page. This feature enabled by default. If you want to disable it, deselect the **Learn on open document** option.

You can choose to create a schema from the learned structure and save it as a file using the **Learn Structure** and **Save Structure** actions.

Creating a JSON Schema from Learned Document Structure

To create a JSON schema from the learned structure of a JSON document, follow these steps:

1. Open the JSON document that will be used to create the schema.
2. Go to **Document > JSON Document > Learn Structure** (**Ctrl + Shift + L (Command + Shift + L on macOS)**). The **Learn Structure** action reads the mark-up structure of the current document. A **Learn completed** message is displayed in the application status bar when the action is finished.
3. Go to **Document > JSON Document > Save Structure** and enter the file path for the JSON schema.
4. Click the **Save** button.

Syntax Highlighting in JSON Documents

Oxygen XML Developer 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 79)*.
2. Go to **Editor > Syntax Highlight** *(on page 158)*.
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 158)*

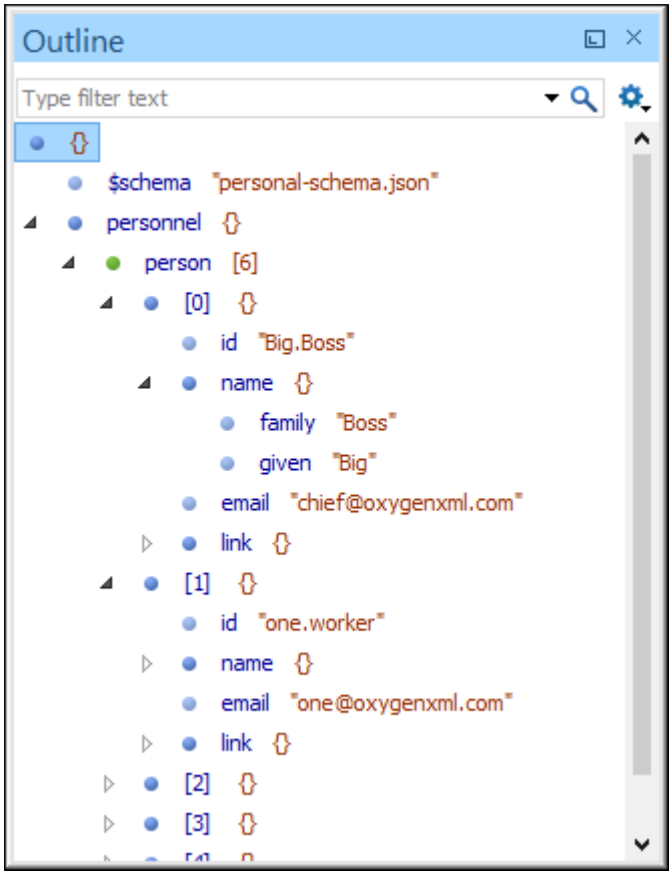
Folding in JSON

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 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.

Figure 303. JSON Outline View



Outline View Features

Some of the features provided by the **Outline** view include:

- You can quickly navigate through the document by selecting nodes in the **Outline** tree.
- You can move elements by dragging them to a new position in the tree structure.
- 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.
- Document errors 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.
- Arrays and array elements have the number of their children elements displayed in their label and each child is prefixed with the index in the array.

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 within 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

Discover the Oxygen JSON Editor!
Tailored for Working with JSON and JSON Schema Documents

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

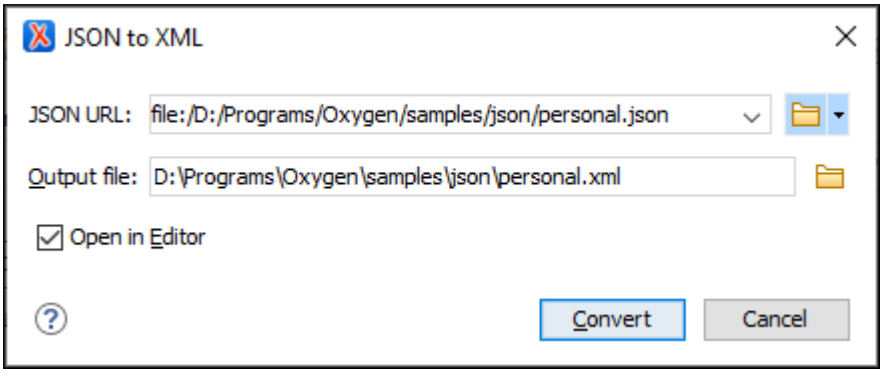
Oxygen XML Developer includes a useful and simple tool for converting JSON files to XML. The **JSON to XML** action for invoking the tool can be found in the **Tools > JSON Tools** menu.

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:

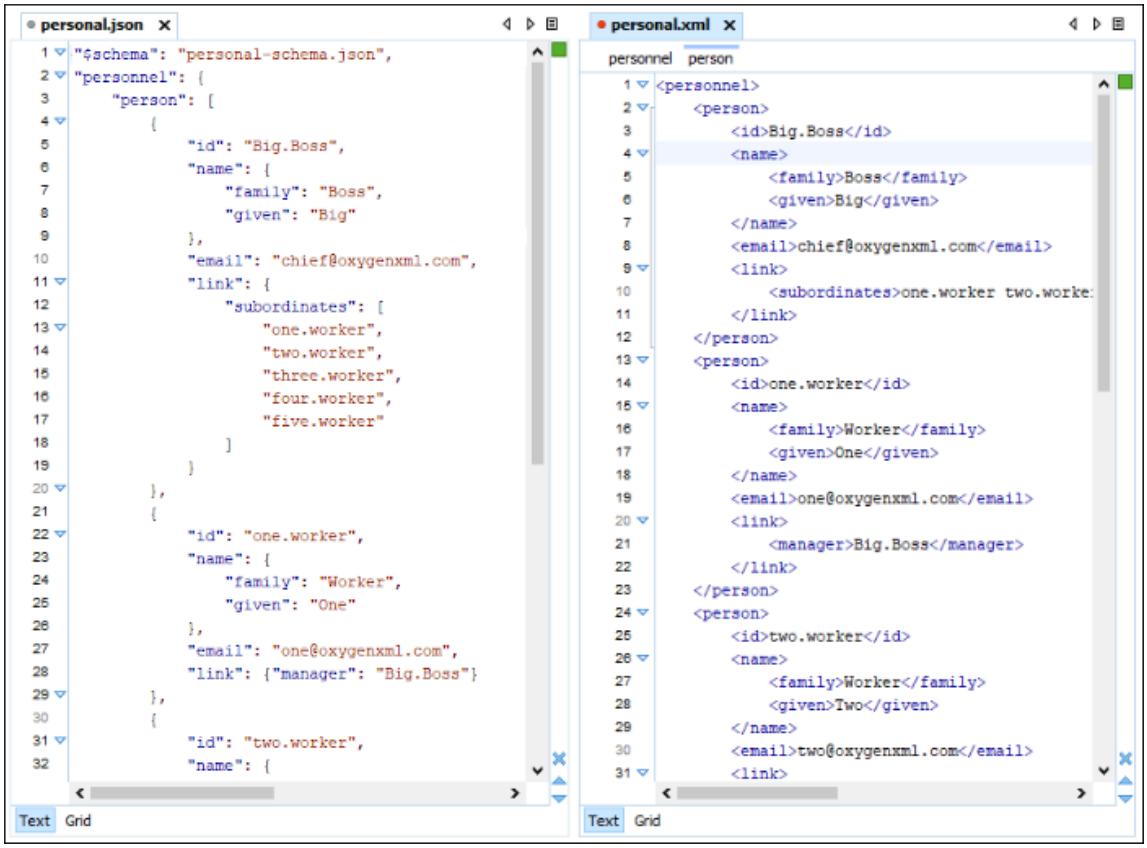
   Figure 304. JSON to XML Dialog Box

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.

Figure 305. Example: XML to JSON Operation Result



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:

```
{
   "personnel": {
    "person": [
      {"name": "Boss"},
      {"name": "Worker"}
    ]
  },
  "id":"personnel-id"
}
```

  it is converted to:

```
<?xml version="1.0" encoding="UTF-8"?>
<JSON>
  <personnel>
```

```
    <person>
      <name>Boss</name>
    </person>
    <person>
      <name>Worker</name>
    </person>
  </personnel>
  <id>personnel-id</id>
</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.

```
[
  {"name": "Boss"},
  {"name": "Worker"}
]
```

it is converted to:

```
<?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.

```
{"$id": "personnel-id"}
```

is converted to:

```
<_X24_id>personnel-id</_X24_id>
```

Related Information:

XML to JSON Converter *(on page 876)*

XML to JSON Converter

Discover the Oxygen JSON Editor!
Tailored for Working with JSON and JSON Schema Documents

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

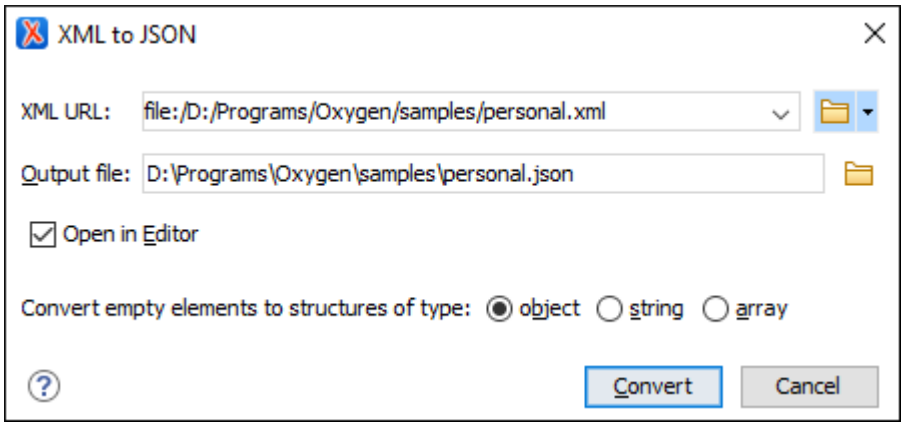
Oxygen XML Developer includes a useful and simple tool for converting XML files to JSON. The **XML to JSON** action for invoking the tool can be found in the **Tools > JSON Tools** menu.

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:

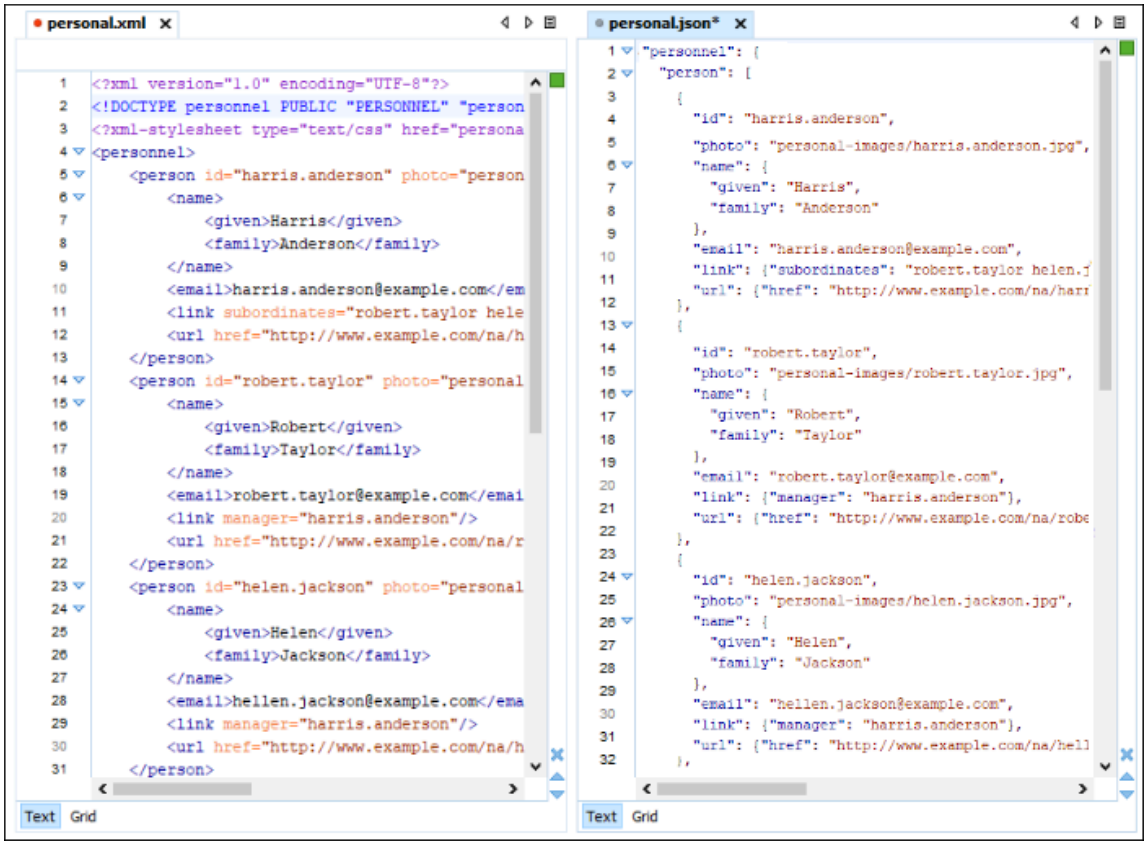
   Figure 306. XML to JSON Dialog Box

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.

Figure 307. Example: XML to JSON Operation Result



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:

```
{
  "personnel": {
    "person": [
      {
```

```
        "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).

  ```
  <p>This <b>is</b> an <b>example</b>!</p>
  ```

  is converted to:

  ```
  {
    "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 873)*), it will be converted to the normal character value in the converted JSON document.

  ```
  <_X24_id>personnel-id</_X24_id>
  ```

  is converted to:

  ```
  {"$id": "personnel-id"}
  ```

Related Information:

JSON to XML Converter *(on page 873)*

JSON to YAML Converter

Converting JSON to YAML in Oxygen

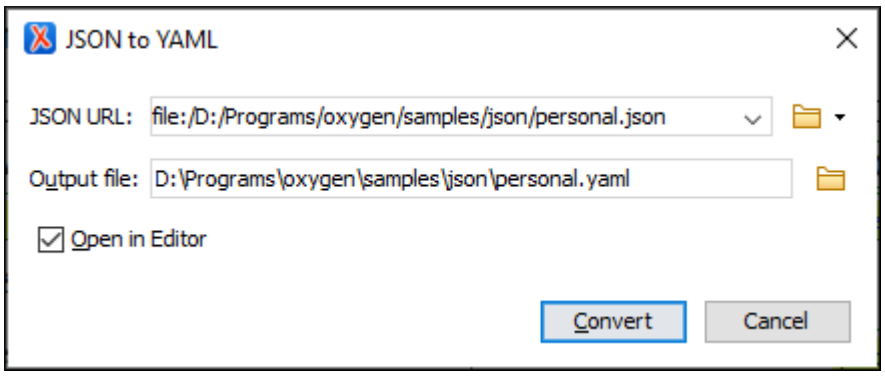
Oxygen XML Developer includes a useful and simple tool for converting JSON files to YAML. The **JSON to YAML** action for invoking the tool can be found in the **Tools > JSON Tools** menu.

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:

   Figure 308. 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 879)*

YAML to JSON Converter

Converting YAML to JSON in Oxygen

Oxygen XML Developer 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.

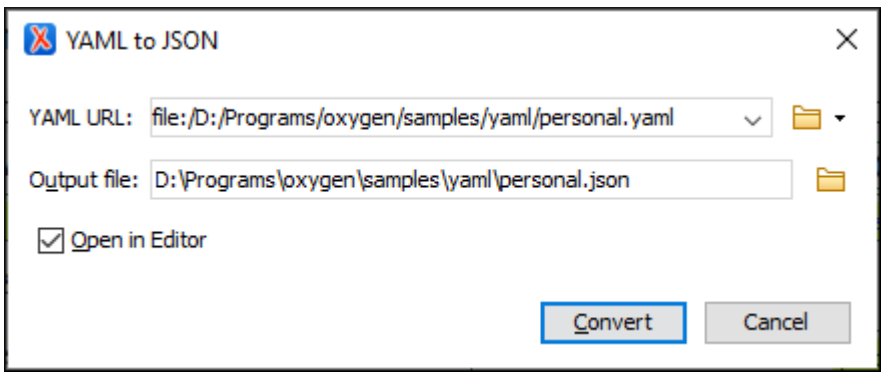
The **YAML to JSON** action for invoking the tool can be found in the **Tools > JSON Tools** menu.

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:

   Figure 309. 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 879)*

Contextual Menu Actions in JSON Documents

When editing JSON documents, Oxygen XML Developer 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.

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.

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 391)*. 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 `ɩ`
- `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 394)*.

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 124)* 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 236)*.

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 124)* 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 236)*.

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 124)* 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 236)*.

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 124)* 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 236)*.

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 124)* 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 236)*.

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 124)* 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 236)*.

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 + Option + 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.

Indent selection (Ctrl + I (Command + I on macOS))

Corrects the indentation of the selected block of lines.

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.

Flatten Schema

Available only if the JSON document is a JSON schema, it flattens the entire hierarchy of the JSON schema. For more details, see Flatten JSON Schema *(on page 922)*.

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 397)*.

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 402)*.

Transforming and Querying JSON Documents

Oxygen XML Developer 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.

Resources

For more information about transforming and querying in JSON, watch our video demonstrations:

- Video: Transform JSON Documents with XSLT/XQuery
- Video: Support for Querying and Transforming JSON Documents

Transforming JSON Documents with XSLT

There are several methods that can be used to transform JSON documents through XSLT processing.

Transforming a JSON Document Directly with XSLT

1. Create a new transformation scenario *(on page 1101)* using one of the following types of transformations:
   - **JSON Transformation with XSLT** *(on page 1144)* - This scenario is useful if you want to develop a JSON document and the XSLT document is in its final form.
   - **XSLT Transformation on JSON** *(on page 1178)* - This scenario is useful if you want to develop an XSLT document and the JSON document is in its final form.
2. Configure the transformation scenario to suit your needs. In the **XSLT** tab, make sure you select the JSON file in the **JSON URL** field and the XSL file in the **XSL URL** field.
3. Run the transformation.

Transforming Multiple JSON Documents at Once

It is also possible to transform multiple JSON documents at once with XSLT processing. To achieve this, select the JSON documents in the **Project** view, right-click, and apply or configure a transformation scenario.

Transforming a JSON Document Using XSLT and XPath Functions

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

There are several methods that can be used to transform JSON documents through XQuery processing.

Transforming a JSON Document Directly with XQuery

1. Create a new transformation scenario *(on page 1101)* using one of the following types of transformations:
   - **JSON Transformation with XQuery** *(on page 1151)***JSON Transformation with XQuery** - This scenario is useful if you want to develop a JSON document and the XQuery document is in its final form.
   - **XQuery Transformation on JSON** *(on page 1198)***XQuery Transformation on JSON** - This scenario is useful if you want to develop an XQuery document and the JSON document is in its final form.
2. Configure the transformation scenario to suit your needs. In the **XQuery** tab, make sure you select the JSON file in the **JSON URL** field and the XQuery file in the **XQuery URL** field.
3. Run the transformation.

Transforming Multiple JSON Documents at Once

It is also possible to transform multiple JSON documents at once with XQuery processing. To achieve this, select the JSON documents in the **Project** view, right-click, and apply or configure a transformation scenario.

Transforming a JSON Document Using XQuery

It is possible to transform JSON documents through XQuery processing.

1. Create an XQuery file.
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.

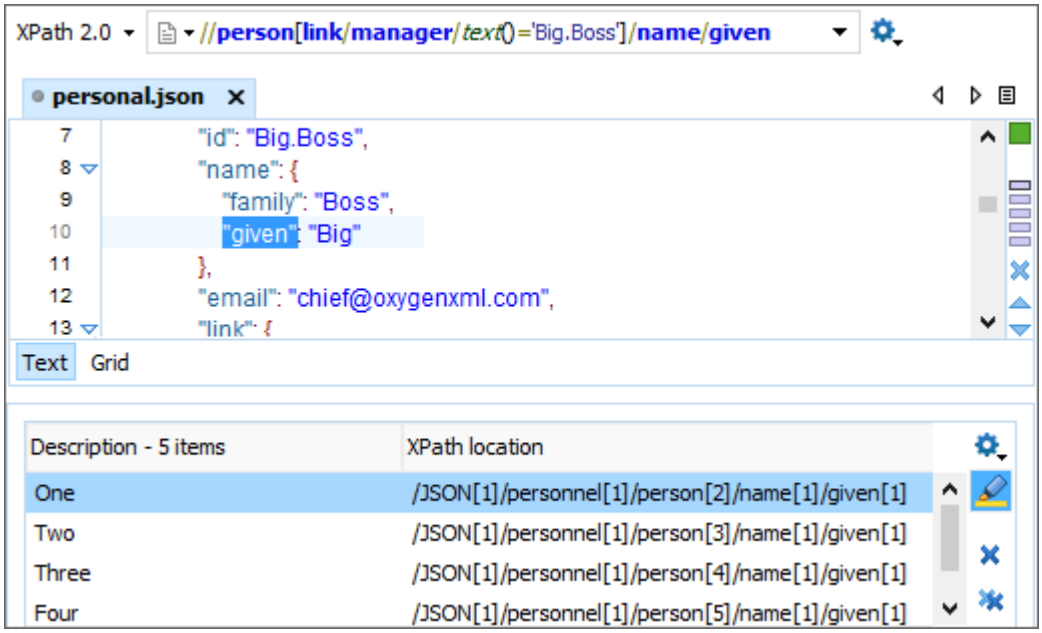
Querying JSON Documents with XPath or XQuery

Oxygen XML Developer 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** 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 1727)*.

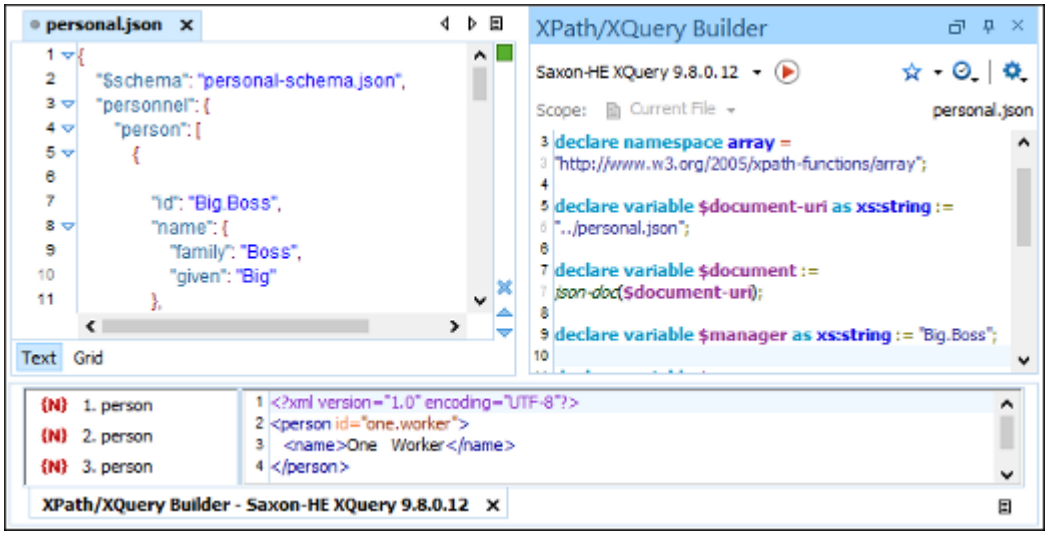
Figure 310. XPath Toolbar for JSON



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 1729)*.

Figure 311. XPath/XQuery Builder View for JSON



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 Developer uses the built-in JSON to XML Converter tool *(on page 873)*. The results are mapped back to the original JSON document.

For example, if you have the following JSON document:

```
{
   "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 874)*.

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.

XQuery Builder View

The **XPath/XQuery Builder** view allows you to compose complex XQuery expressions and execute them over the currently edited XML or JSON 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 Developer 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 Developer 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 Developer 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 at the bottom of the application.

 **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 Developer 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 397)* or when the scope is other than **Current file**.

Options

Opens the Preferences page of the currently selected processing engine.

XPath scope menu

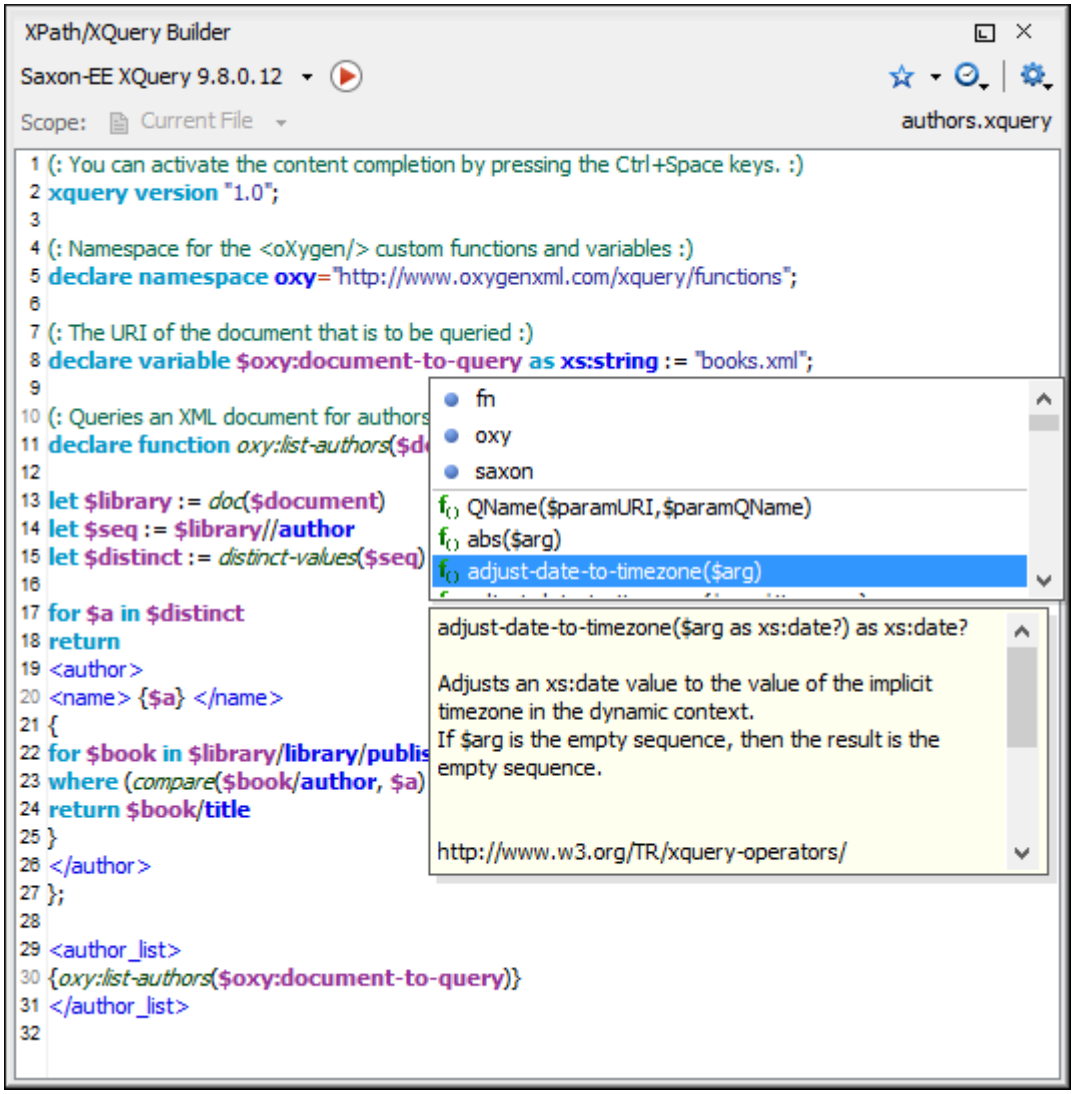
Oxygen XML Developer allows you to define a scope for the XPath operation to be executed. You can choose where the XPath expression will be executed:

- **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.
- **Opened archive** - Files that are opened in the **Archive Browser** view *(on page 1735)*.
- **Working sets** - The selected *working sets (on page 2309)*.

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 2309)*.
- **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 312. XPath/XQuery Builder View



While you edit an XPath or XQuery expression, Oxygen XML Developer assists you with the following features:

- *Content Completion Assistant (on page 2303)* - 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 79)* and go to **Editor > Syntax Highlight** *(on page 158)*.
- 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.

Editing JSON Schema Documents

Oxygen XML Developer offers powerful tools that allow you to design, develop, and edit JSON Schemas. These tools include:

- **Text** editing mode *(on page 894)* (packed full of editing helpers)
- Schema **Design** mode *(on page 894)* (a visual, intuitive schema diagram editor)
- **Grid** mode *(on page 894)* (a compact layout of nested tables)
- JSON Schema documentation tool *(on page 917)* for producing high quality output
- JSON Schema instance generator *(on page 915)* for producing JSON Schema documents from a JSON file
- XSD to JSON Schema converter tool *(on page 919)* for producing a JSON Schema from an XML schema

This section describes the features included in Oxygen XML Developer for editing JSON Schema documents and how to use them.

Resources

For more information about the JSON support in Oxygen XML Developer, see the following resources:

- Video: **Introducing JSON Schema Design Mode**
- Video: **JSON Editing**
- Video: **JSON Tools in Oxygen**
- Video: **JSON Schema Search and Refactoring Actions in Oxygen**
- Video: **JSON Schema Version 2020-12 Support in Oxygen**
- Webinar: **JSON and JSON Schema Support in Oxygen**
- Webinar: **Creating and Designing JSON Schemas**

JSON Schema Editor

Oxygen XML Developer includes a specialized JSON Schema editor with various editing features for files that have the `jschema` file extension, or for files that have `json` file extension and includes a meta-schema URL *(on page 920)* 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 Developer 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 299)*. You can also customize your own JSON Schema templates *(on page 309)* and specify other versions (Draft 04, 06, 07, 2019-09, or 2020-12).

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 are available, along with various other actions, including:

- JSON Outline View *(on page 870)*
- JSON-specific Syntax Highlighting *(on page 921)*
- Search and Find/Replace *(on page 354)*
- Drag and Drop
- Validation *(on page 855)*
- Format and Indent (Pretty Print)

Grid Mode Editor

Oxygen XML Developer allows you to view and edit the JSON Schema documents in the **Grid** mode. 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 851)*.

Design Mode Editor

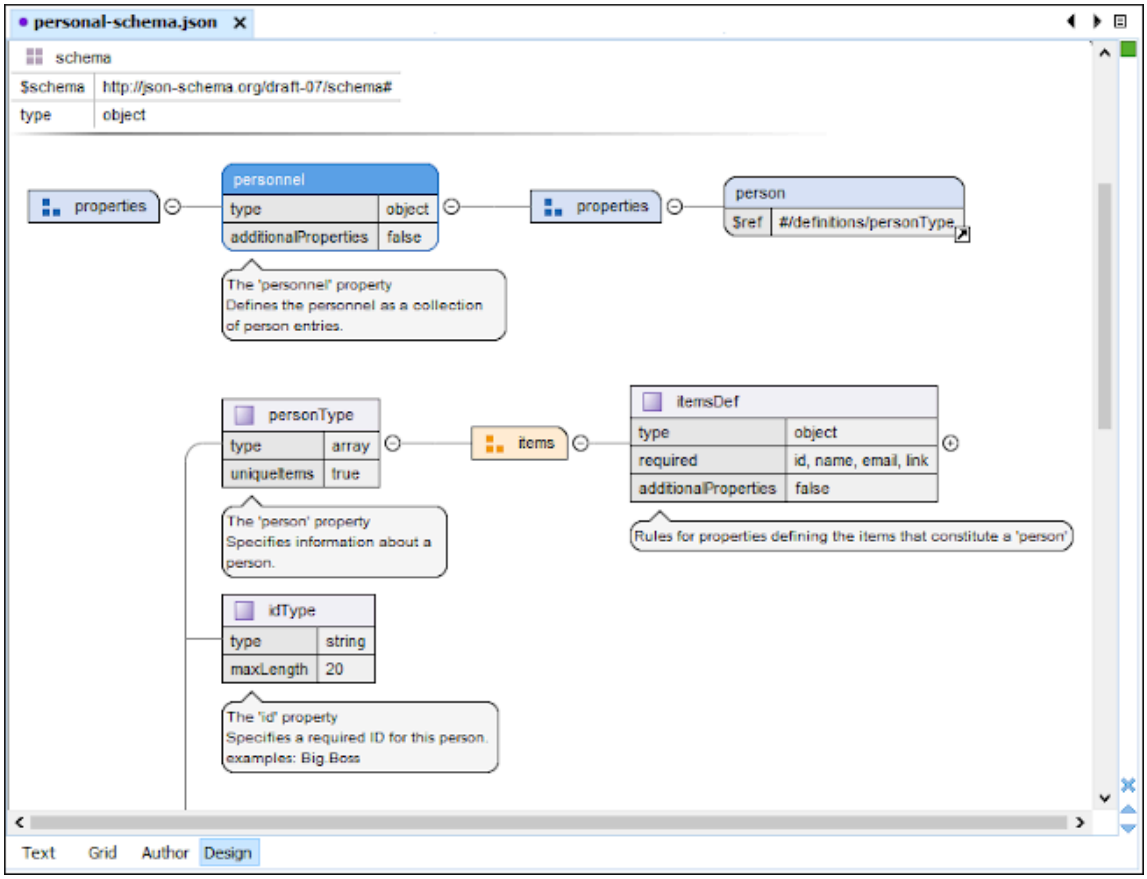
Oxygen XML Developer provides a powerful, expressive visual schema diagram editor (**Design** mode) for editing JSON Schemas. It is helpful for both content authors who want to visualize or understand a schema and schema designers who develop complex schemas. For all the details, see JSON Schema Design Mode *(on page 894)*.

JSON Schema Design Mode (JSON Schema Diagram Editor)

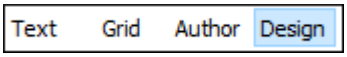
Oxygen XML Developer provides a powerful, expressive visual schema diagram editor (**Design** mode) for editing JSON Schemas. The structure of the diagram editor is designed to be intuitive and easy to use. The **Design** mode was created to help both content authors who want to visualize or understand a schema and schema designers who develop complex schemas.

The JSON Schema **Design** mode includes various navigation features *(on page 896)*, contextual menu actions *(on page 899)*, automatic validation *(on page 914)*, and you can move and edit components directly within the diagram *(on page 898)*.

Figure 313. JSON Schema Design Mode



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 Developer 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 131)*. The same zoom factor is applied for the print and save actions.

Resources

For more information about the JSON Schema Design mode, see the following resources:

- Video: **Introducing JSON Schema Design Mode**
- Video: **JSON Schema Search and Refactoring Actions in Oxygen**
- Video: **JSON Schema Version 2020-12 Support in Oxygen**
- Webinar: **The New JSON Schema Diagram Editor**
- Webinar: **Create JSON Schema in Design Mode**
- Webinar: **Creating and Designing JSON Schemas**

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 using the following toolbar actions:
  - **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.
- The dedicated **Outline** view displays all of the properties, pattern properties, and definitions found in the document and you can click on any property/definition to navigate to it within the document.
- Search in the diagram using the Find/Replace dialog box *(on page 364)* or the Quick find toolbar *(on page 378)*. 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 314. JSON Palette View (for schema versions draft-04, draft-06, draft-07)

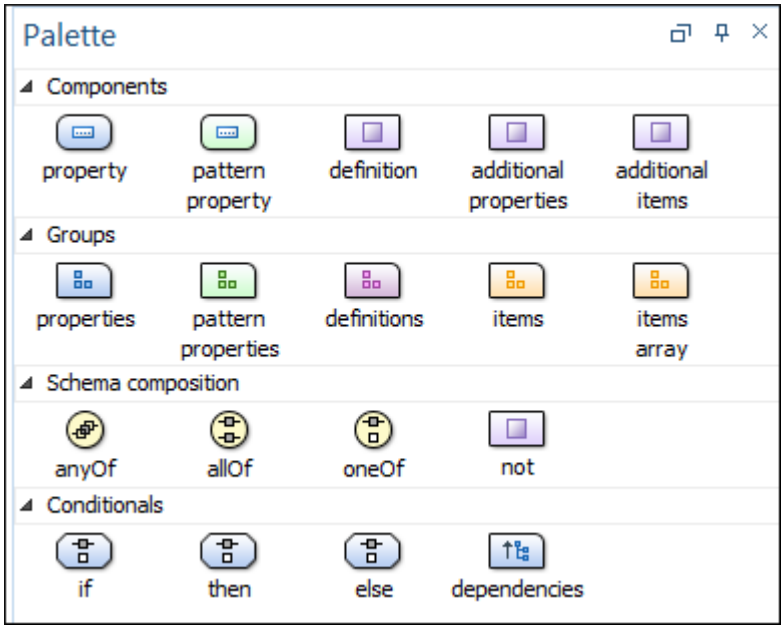
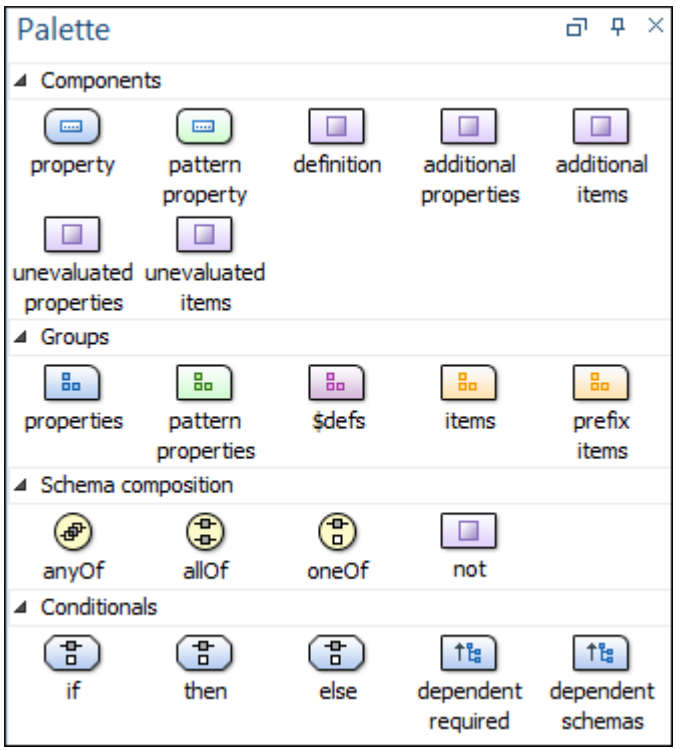


Figure 315. JSON Palette View (for schema versions 2019-09, 2020-12)



Components are organized functionally into 4 collapsible categories:

- **Components:** *property*, *patternProperty*, *definition*, *additionalProperties*, *additionalItems*, *unevaluatedProperties* (for 2019-09 or 2020-12 schemas), *unevaluatedItems* (for 2019-09 or 2020-12 schemas).
- **Groups:** *properties*, *patternProperties*, *definitions*/*$defs* (for draft 2019-09 or 2020-12 schemas), *items*, *itemsArray*, *prefixItems* (for 2019-09 or 2020-12 schemas).

> **Note:**
> Two of the group palette components presented above (*itemsArray* and *patternProperty*), 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.

- **Schema composition:** *anyOf*, *allOf*, *oneOf*, *not*.
- **Conditionals:** *if*, *then*, *else*, *dependencies* (for draft-04, draft-06, or draft-07 schemas), *dependentRequired* (for 2019-09 or 2020-12 schemas), *dependentSchemas* (for 2019-09 or 2020-12 schemas).

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 160)* (default color is red).

> **Tip:**
> When dragging and dropping a property/definition or pattern property on a component, if it does not have the corresponding group component (*properties*, *definitions*, *patternProperties*), it will automatically be created.

Resources

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 899)*.
- 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 132)*.

- 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.
- You can drag properties/definitions from the **Outline** view and drop them into appropriate location in the diagram editor.

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).

Edit Dependencies

Available for `dependencies` and `dependentRequired` components, this action allows you to add, rename, delete, and edit the values for dependencies.

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 in another file

Extracts a definition to a new file. If the file does not already exist, the action will create a new file and a document preset will be used to match the current schema specification. If the file does exist, the action will find a corresponding group (or create one) to append the extracted definition. This action can also be used on a selection of multiple definitions.

Refactoring > Extract definition in current file

Extracts a definition as a global definition and references 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. This action can also be used on a selection of multiple definitions. Note that this action is not available for global definitions.

Refactoring > Convert type to 'any' type

Converts the `type` for the selected property, definition, or conditional 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.

Refactoring > Convert 'any' type to standard type

Converts the `any type` for the selected property, definition, or conditional into a standard `type`.

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.



Undo [Ctrl + Z (Command + Z on macOS)]

Reverses the last editing action.



Redo [Ctrl + Y (Command + Shift + Z on macOS, Ctrl + Shift + Z on Linux/Unix)]

Recreates the last editing action that was reversed by the **Undo** function.

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. These files are identified by searching the references of the selected component in the scope provided.



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 key]

Removes the selected component(s).

Move Up [Alt + UpArrow (Option + UpArrow on macOS)]

Moves a component up in its parent.

Move Down [Alt + DownArrow (Option + DownArrow on macOS)]

Moves a component down in its parent.

Search > Search References

Available on components that have a *Definition* type in the diagram, it searches all references of the selected definition in a scope determined by the schemas referenced in the file and the schemas declared in the validation scenarios associated with them. You can also use it on the *Schema* type component (the root of the schema diagram) to search for all references to the schema name.

Search > Search References in

Available on components that have a *Definition* type in the diagram, it is an extension of the **Search References** action, where the search for references is additionally done in the file(s) specified when defining a scope in the resulting dialog box. You can also use it on the *Schema* type component (the root of the schema diagram) to search for references to the schema name.

Search > Search Occurrences in File [Ctrl + Shift + U (Command + Shift + U on macOS)]

Available on all components that have a *Definition* type in the diagram, it searches all occurrences of the currently selected definition in the current file. You can also use it on the *Schema* type component (the root of the schema diagram) to search for all occurrences of the schema name in the current file.

Flatten Schema

Flattens the entire hierarchy of JSON schemas. For more details, see Flatten JSON Schema *(on page 922)*.

 **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 132)* 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).

Figure 316. Example: Required Component

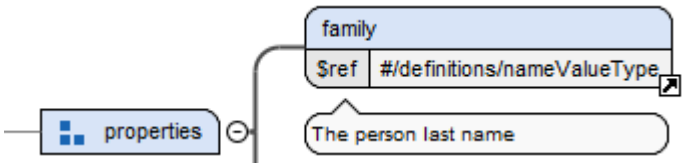
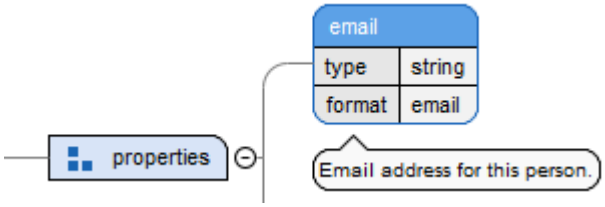


Figure 317. Example: Optional Component

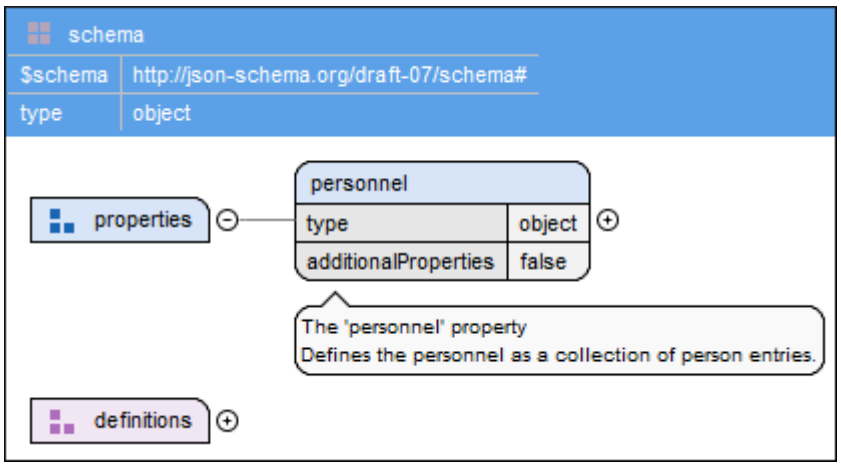


This section provide details about the available components and their corresponding graphics, along with details about component properties.

JSON Schema Components

Schema

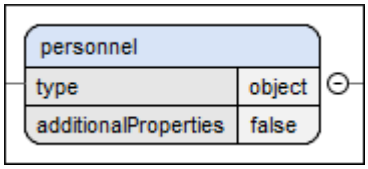
Figure 318. The schema Component



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.

Property

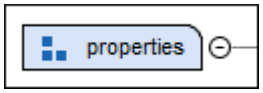
Figure 319. Example of a property Component



Description: Each `key:value` pair is known as a **property** of the object. The value can be any JSON data type.

Properties

Figure 320. The properties Component

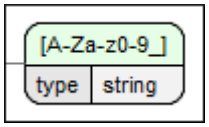


Description: An object is valid against the **properties** keyword if every property that is present in both the object and the value of this keyword validates against the corresponding schema. The value must be an object, where properties must contain valid JSON schemas (object or boolean). Only the 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.

Pattern Property

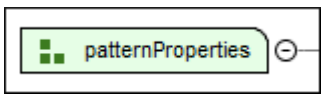
Figure 321. Example of a patternProperty Component



Description: Maps regular expressions to schemas. If a property name matches the given regular expression, the property value must validate against the corresponding schema.

Pattern Properties

Figure 322. The patternProperties Component

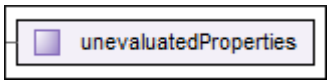


Description: An object is valid against the **patternProperties** keyword if every property where a property name (key) matches a regular expression from the value of this keyword is also valid against the corresponding

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).

Unevaluated Properties (for 2019-09 or 2020-12 schemas)

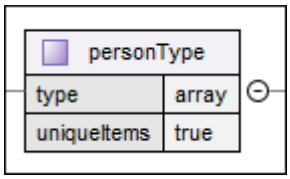
Figure 323. The unevaluatedProperties Component



Description: An object is valid against the **unevaluatedProperties** keyword if every unevaluated property is valid against the schema defined by the value of this keyword. Unevaluated properties are the properties that were not evaluated anywhere in the current schema. This keyword can see through adjacent keywords, such as **allOf**.

Definition

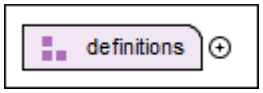
Figure 324. Example of a definition Component



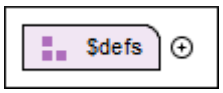
Description: The definition of a component of a schema. It can be referenced from a property to define its specification.

Definitions

Figure 325. The definitions Component



or

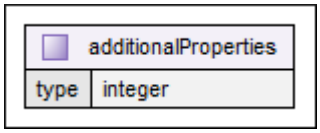


Description: The optional **definitions** keyword (or **$defs** for *draft 2019-09* or *2020-12* schemas) does not directly validate data, but it contains a map of validation schemas. The value can be anything.

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.

Additional Properties

Figure 326. The additionalProperties Component



Description:

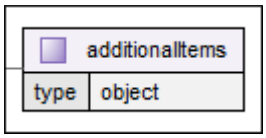
An object is valid against the **additionalProperties** keyword if all *unchecked* properties are valid against the schema defined by the value of this keyword. *Unchecked* properties are the properties not checked by the **properties** and **patternProperties** keywords (if a property name is not present 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.

Additional Items

Figure 327. The additionalItems Component



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 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).

Items

Figure 328. The items Component

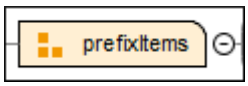


Description: An array is valid against the **items** keyword if the items are valid against the corresponding schemas provided by the keyword value. The value can be:

- 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) are considered valid unless the **additionalItems** keyword is present, which will decide the validity.

Prefix Items (for 2020-12 schemas)

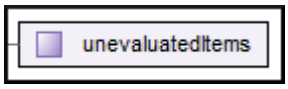
Figure 329. The prefixItems Component



Description: An array is valid against the **prefixItems** keyword if items are valid against the corresponding schemas provided by the keyword value. The value of this keyword must be an array of valid JSON schemas and each item must be valid against the schema defined at the same position (index). Items that do not have a corresponding position (an array contains 5 items and this keyword only has 3) will be considered valid, unless the **items** keyword is present (which will decide the validity).

Unevaluated Items (for 2019-09 or 2020-12 schemas)

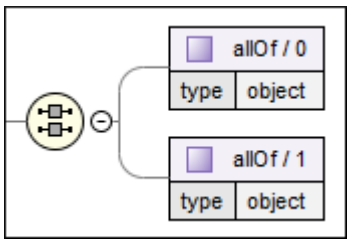
Figure 330. The unevaluatedItems Component



Description: An object is valid against the **unevaluatedItems** keyword if every unevaluated item is valid against the schema defined by the value of this keyword. Unevaluated items are the items that were not evaluated anywhere in the current schema. This keyword can see through adjacent keywords, such as **allOf**.

AllOf

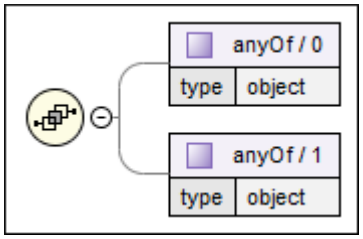
Figure 331. Example of an allOf Component



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

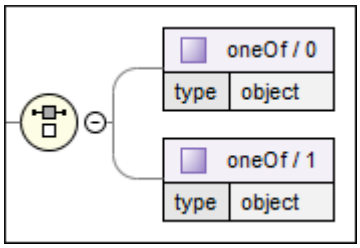
Figure 332. Example of an anyOf Component



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

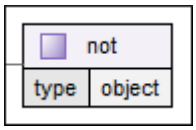
Figure 333. Example of an oneOf Component



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

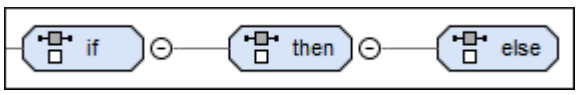
Figure 334. Example of a not Component



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 335. Examples of an if, then, and else Components



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.

Dependencies

Figure 336. The dependencies Component



Description: An object is valid against the **dependencies** keyword if it meets all dependencies specified by this keyword value. Only property names (from this keyword value) that are also present in the object are checked. The value of this keyword must be an object, where property values can be:

- Objects representing valid JSON schemas and the whole object must match the entire schema.

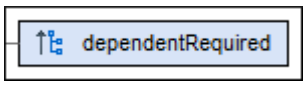
> **Important:**
> For *draft 2019-09* and *2020-12* schemas, you should use the **dependentSchemas** keyword *(on page 909)* instead.

- Arrays of strings representing property names, then the object must contain all property names.

> **Important:**
> For *draft 2019-09* and *2020-12* schemas, you should use the **dependentRequired** keyword *(on page 909)* instead.

Dependent Required (for 2019-09 or 2020-12 schemas)

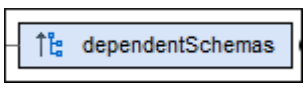
Figure 337. The dependentRequired Component



Description: An object is valid against the **dependentRequired** keyword if it meets all dependencies specified by this keyword value. The value of this keyword must be an object where property values must be arrays of strings representing property names, and the object must contain all property names. Only property names (from this keyword value) that are also present in the object are checked.

Dependent Schemas (for 2019-09 or 2020-12 schemas)

Figure 338. The dependentSchemas Component



Description: An object is valid against the **dependentSchemas** keyword if it meets all dependencies specified by this keyword value. The value of this keyword must be an object where property values must be objects

representing valid JSON schemas, and the whole object must match the entire schema. Only property names (from this keyword value) that are also present in the object are checked.

Related information

JSON Schema Component Properties *(on page 910)*

JSON Schema Component Properties

Table 34. JSON Schema Diagram Component Properties

Properties Group	Property Name	Description
Common Properties	**type**	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.
	$ref	An instance is valid against this keyword if it is valid against the schema that points to the location indicated in its value. The value must be a string representing a URI, URI reference, URI template, or JSON pointer. **Note:** A schema version *2019-09* or *2020-12* that contains a `$ref` in conjunction with other type-specific keywords (such as *properties* or *items*) is processed as a combined schema, under the *allOf* criterion. This means that for an instance to be valid, it has to validate against both the referenced schema and the schema defined in-place by those keywords.
	$id	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.
	enum	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).
	const	An instance validates against this keyword if its value equals the value of this keyword. The value can be anything.
	$comment	Contains an observation about the schema. The value must be a string.

Table 34. JSON Schema Diagram Component Properties (continued)

Properties Group	Property Name	Description
	readOnly	Used to mark specific properties as *read-only*.
	writeOnly	Used to mark specific properties as *write-only*.
	deprecated	Used to indicate that the instance value is deprecated and should not be used since it might be removed in the future.
Object Properties	**additionalProperties**	An object is valid against this keyword if all *unchecked* properties are valid against the schema defined by its value. *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 we have *unchecked* properties: • If the value of this 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.
	unevaluatedProperties	Similar to the **additionalProperties** keyword except that this one can recognize properties that declared in subschemas.
	maxProperties	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).
	minProperties	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.
	patternProperties	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).

Table 34. JSON Schema Diagram Component Properties (continued)

Properties Group	Property Name	Description
	propertyNames	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.
	dependencies	An object is valid against this keyword if it meets all dependencies specified by its value. **Note:** For *2019-09* and *2020-12* schemas, you should use the **dependentRequired** and **dependentSchemas** keywords instead.
	dependentRequired	An object is valid against this keyword if it meets all dependencies specified by its value. The value must be an object where property values must be arrays of strings representing property names, and the object must contain all property names.
	dependentSchemas	An object is valid against this keyword if it meets all dependencies specified by its value. The value must be an object where property values must be objects representing valid JSON schemas, and the whole object must match the entire schema.
Array Properties	**additionalItems**	An array is valid against this keyword if all *unchecked* items are valid against the schema defined by its value.
	unevaluatedItems	An array is valid against this keyword if the value is a valid JSON schema that will be applied to all array items that were not evaluated by other keywords for items (`prefixItems`, `items`, `contains`).
	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).
	maxContains	An array is valid against this keyword if the number of *contains* is lower than or equal to its value. The value must be a non-negative integer.
	minContains	An array is valid against this keyword if the number of *contains* is greater than or equal to its value. The value must be a non-negative integer.

Table 34. JSON Schema Diagram Component Properties (continued)

Properties Group	Property Name	Description
	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: • 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.
	maxItems	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.
	minItems	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.
	prefixItems	An array is valid against this keyword if each item is a schema that corresponds to each index of the document's array (where the first element validates the first element of the input array, the second element validates the second element of the input array, and so on).
	uniqueItems	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.
Number/Integer Properties	**exclusiveMaximum**	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.
	exclusiveMinimum	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.
	maximum	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).

Table 34. JSON Schema Diagram Component Properties (continued)

Properties Group	Property Name	Description
	minimum	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).
	multipleOf	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).
String Properties	**contentEncoding**	A string is valid against this keyword if it is encoded using the method indicated by its value. The value must be a string.
	contentMediaType	A string is valid against this keyword if its content has the media type (MIME type) indicated by its value. If the **contentEncoding** keyword is also specified, the decoded content must have the indicated media type. The value must be a string.
	format	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 **format** 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.
	maxLength	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.
	minLength	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.
	pattern	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.

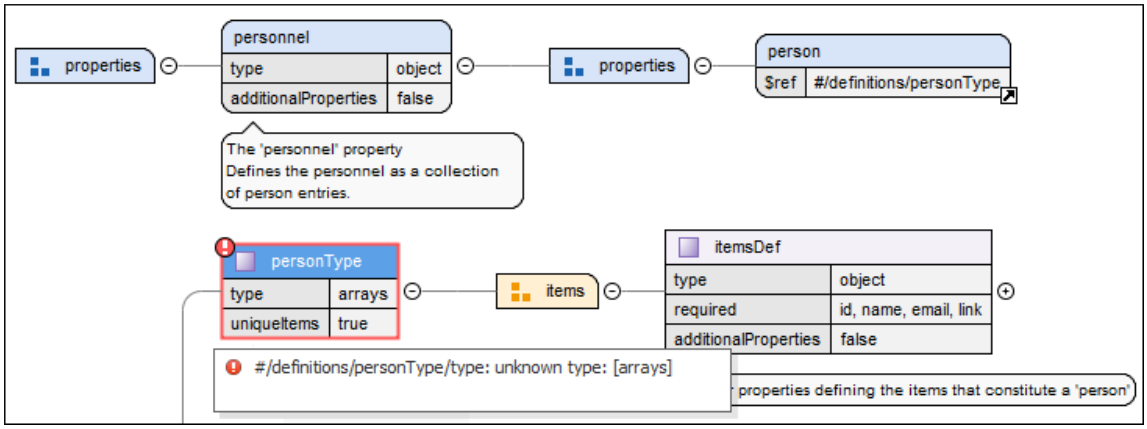
Related information

JSON Schema Components *(on page 903)*

JSON Schema Design Mode Validation

Validation for the **Design** mode is seamlessly integrated in the Oxygen XML Developer JSON Schema validation support *(on page 920)*. 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.

Figure 339. JSON Schema Design Mode Validation



Visual Error Markers

A schema validation error is presented by highlighting the invalid component in the following locations:

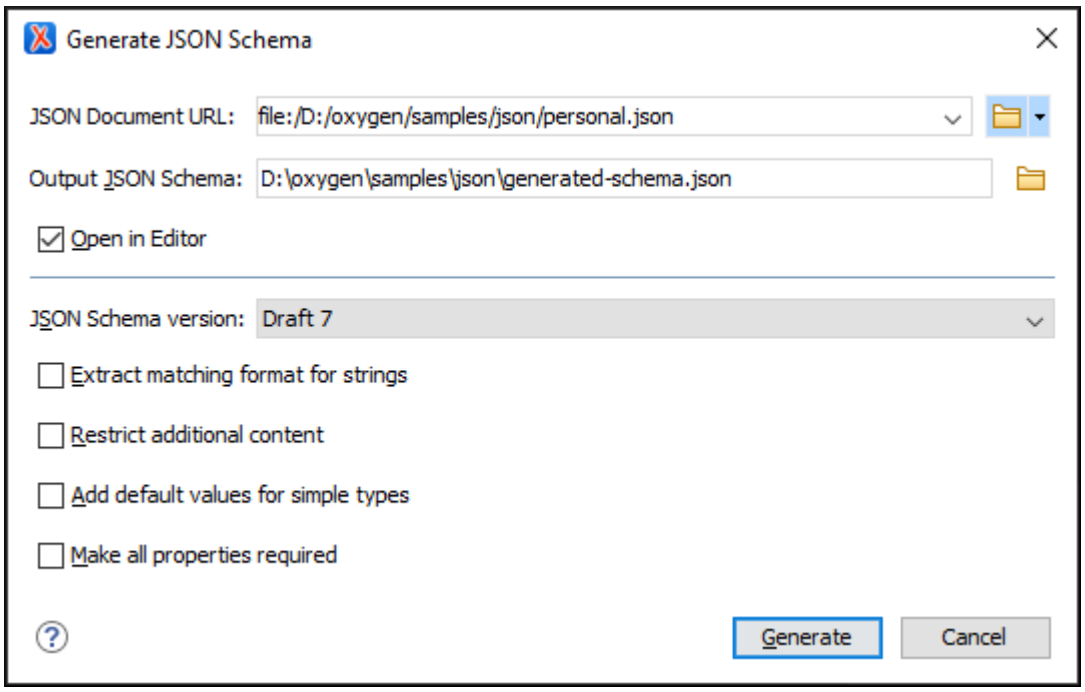
- The component is surrounded with a red or yellow border within the diagram (you can customize these colors in the Document Validation preferences page *(on page 160)*).
- A tooltip contains information about the error when hovering over the error within the diagram.
- The vertical stripe at the right 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 Developer 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 340. 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**, **2019-09**, and **2020-12**.

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 Developer 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.

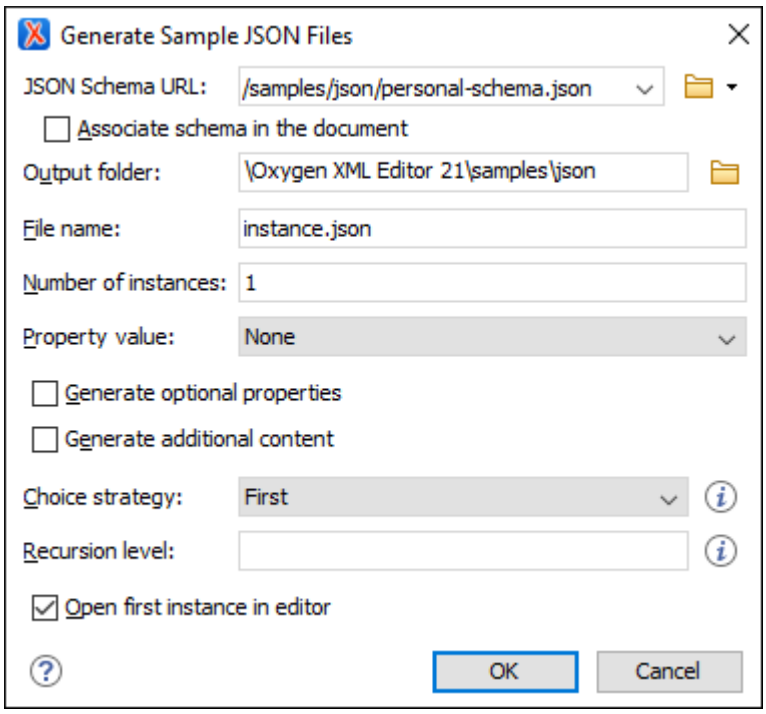
This tool requires an additional add-on to be installed, so the first time you invoke the action, Oxygen XML Developer presents a dialog box asking if you want to install it. Once installed, you need to restart Oxygen XML Developer and the **JSON Schema Documentation** action will invoke the tool.

Add-on documentation and installation details.

Generating Sample JSON Files from a JSON Schema

Oxygen XML Developer 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.

Figure 341. Generate Sample JSON Files Dialog Box



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. The tool supports schemas with versions *Draft 04*, *06*, *07*, *2019-09*, and *2020-12*.

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 Developer 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 Developer will present a dialog box asking if you want to install it. Once installed, you need to restart Oxygen XML Developer and the **XSD to JSON Schema** action will invoke the tool.

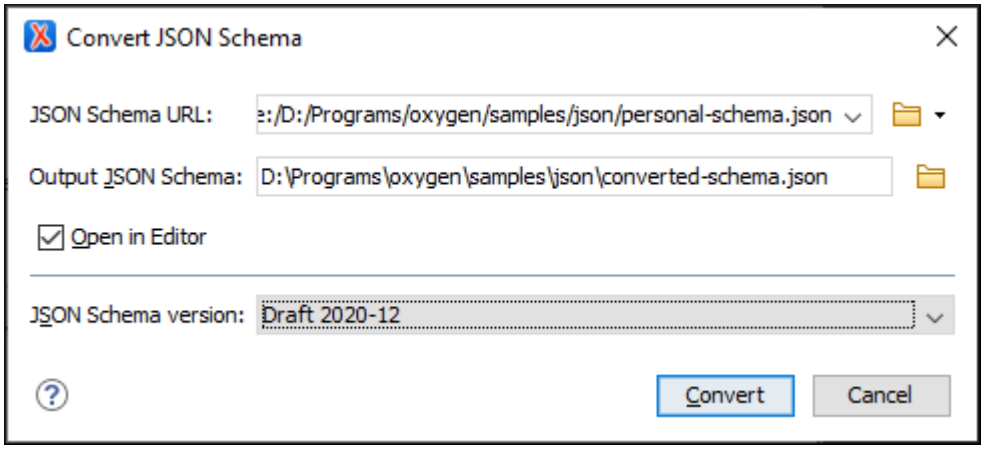
Addon documentation and installation details.

JSON Schema Converter

Oxygen XML Developer includes a tool for converting an older version of a JSON schema (Draft 4, 6, or 7) to the latest versions (2019-09 or 2020-12).

To convert a JSON schema, select **Convert JSON Schema** from the **Tools > JSON Tools** menu. The action opens a dialog box where you can configure some options for converting the JSON Schema.

Figure 342. Convert JSON Schema Dialog Box



The **Convert JSON Schema** 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 JSON Schema

The path to the folder where the converted JSON schema will be saved.

Open in Editor

If selected, the converted JSON schema is opened in the editor.

JSON Schema version

The version of the resulting JSON schema. The possible choices are: **Draft 2019-09** or **2020-12**.

You can click **Convert** at any point to generate the JSON Schema.

Conversion Notes

- The `$schema` declaration is changed according to the selected JSON schema version.
- The `definitions` keyword is converted to `$defs` and all the references are updated.
- The `dependencies` keyword is split into `dependentRequired` and `dependentSchemas`.
- The `items` keyword (tuple array) is converted to `prefixItems` (2020-12).
- The `additionalItems` keyword is converted to `items` (2020-12, only if `prefixItems` is present).
- The `exclusiveMinimum` and `exclusiveMaximum` keywords with boolean values (Draft 4) are removed.
- The `id` keyword (Draft 4) is converted to `$id`.
- The `$ref` keyword wrapped into 1-item `allOf` is unwrapped because the latest versions allow processing `$ref` along with other keywords.

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 2019-09 will be used.
- If there is a "**$schema**": "**http://json-schema.org/draft/2020-12/schema**" property in the schema root, then 2020-12 will be used.
- If there is a "**$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 859)* 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 JSON Schema Through a Validation Scenario *(on page 865)*.

For information about using a JSON Schema to validate documents, see Validating JSON Documents Against JSON Schema or Schematron *(on page 856)*.

2019-09 and 2020-12 Validator Limitations

The JSON Schema Validator handles all the newly introduced keywords in 2019-09 and 2020-12 specifications. However, there are still some limitations:

1. The keywords **"$recursiveRef"** and **"$recursiveAnchor"** (2019-09) are not supported. This is also indicated by a validation warning.
2. The keyword **"$dynamicRef"** has the same functionality as **"$ref"**, and **"$dynamicAnchor"** (2020-12) is not supported. This is also indicated by a validation warning.
3. The keywords **"unevaluatedProperties"** / **"unevaluatedItems"** can "see through" the subschemas of adjacent keywords **"if", "then", "else",** but do not know about successfully validated *properties* / *items*. This means that all the *properties* / *items* defined by those subschemas are considered evaluated.
4. The keywords **"unevaluatedProperties"** / **"unevaluatedItems"** can "see through" the nested subschemas of adjacent keywords **"oneOf", "anyOf", "allOf"**, but all the *properties* / *items* defined by those subschemas are considered evaluated, regardless of other nested restrictions.

Syntax Highlighting in JSON Schema Documents

Oxygen XML Developer 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 79)*.
2. Go to **Editor > Syntax Highlight** *(on page 158)*.
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 158)*

Flatten JSON Schema

Oxygen XML Developer includes a **Flatten Schema** action that is available in the contextual menu when editing in JSON Schema **Design** mode *(on page 894)* and in the Text mode. It allows you to flatten an entire hierarchy of JSON schemas. Starting with the main JSON schema, Oxygen XML Developer calculates its hierarchy by processing the `$ref` keys and determining all the other referenced schema files. This means that the flattened JSON schema is obtained by recursively adding the components of all referenced schemas into the main one, and by updating all the `$ref` keys to point to the components from the resulting schema.

Resolving schema references is done through XML Catalogs. That means that the sub-schemas referenced through URIs and not mapped accordingly cannot not be parsed for integration into the resulting flattened schema, affecting its logic.

The **Flatten Schema** action opens a small dialog box that allows you to configure the operation by choosing the resulting schema name and the directory to save it, and opting whether to open the resulting schema in the editing area after the operation completes.

Editing JSON Lines Documents

Oxygen XML Developer includes some basic support for working with **JSON Lines** documents. *JSON Lines* is a convenient format for storing structured data that may be processed one record at a time.

Editing JSON Lines Documents

You can edit JSON Lines documents in the **Text** mode editor in Oxygen XML Developer and you have access to its various features and actions that are common for basic text files.

Validation

Validation support is available for JSON Lines documents if you have associated a schema through a configured validation scenario.

Content Completion

Content completion support is also available for JSON Lines documents if you have associated a schema through a configured validation scenario.

Editing JSON5 Documents

Oxygen XML Developer includes support for working with **JSON5** documents (files that have the `json5` file extension). The *JSON5* format is a superset of JSON that aims to alleviate some of the limitations of JSON by expanding its syntax to include some productions from *ECMAScript 5.1*.

Editing JSON5 Documents

You can edit JSON5 documents in the **Text** mode editor in Oxygen XML Developer and you have access to its usual text editing actions, along with other JSON5 editor-specific actions, including syntax highlighting, validation, formatting and indenting.

Note:
The **Format and Indent** action works for JSON5 documents with limitations being that quotes are removed for keys and values are always double quoted (regardless of the initial quotation).

It also includes a document template to help you get started with JSON5 documents. The template is called **JSON5** and it can be found in the **New Document** folder in the **New** document wizard *(on page 299)*.

Outline View

The **Outline** view for JSON5 documents displays the list of all the components of the document you are editing in a hierarchical (tree-like) representation. It is synchronized with the main editor so you can use the view to navigate to specific parts of the document and move or delete 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.

Validation

Oxygen XML Developer includes built-in validation engine for JSON5 documents to help keep them well-formed. JSON5 documents are validated automatically as you type and the validation engine detects syntax errors, based on the JSON5 specification. One limitation is that hexadecimal values are not supported.

Editing YAML Documents

This section discusses the various features that are included in the Oxygen XML Developer YAML Editor and how to use them.

Resources

For more information about the YAML (and JSON) support in Oxygen XML Developer, see the following resources:

- **Video: YAML Support in Oxygen**
- **Webinar: Exploring Oxygen's YAML Support**
- **Webinar: JSON and JSON Schema Support in Oxygen**

YAML Editor

Oxygen XML Developer includes a specialized YAML editor with various editing features for files that have the `yaml/yml` file extension. It 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 299)*.

Tip:
You can experiment with a sample of a YAML file available at: `[OXYGEN-INSTALL-DIR]/samples/yaml/personal.yaml`.

Text Mode Editor

When editing YAML documents in the **Text** editing mode, the usual text editing actions are available, along with other YAML editor-specific actions, including:

- Syntax highlighting *(on page 930)*
- Automatic validation *(on page 924)*
- Formatting and indenting *(on page 930)*

Validating YAML Documents

Automatic Validation

Oxygen XML Developer 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

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 334)*. 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 1873)* developed independently and added to Oxygen XML Developer 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 334)*. 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 334)*. 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:

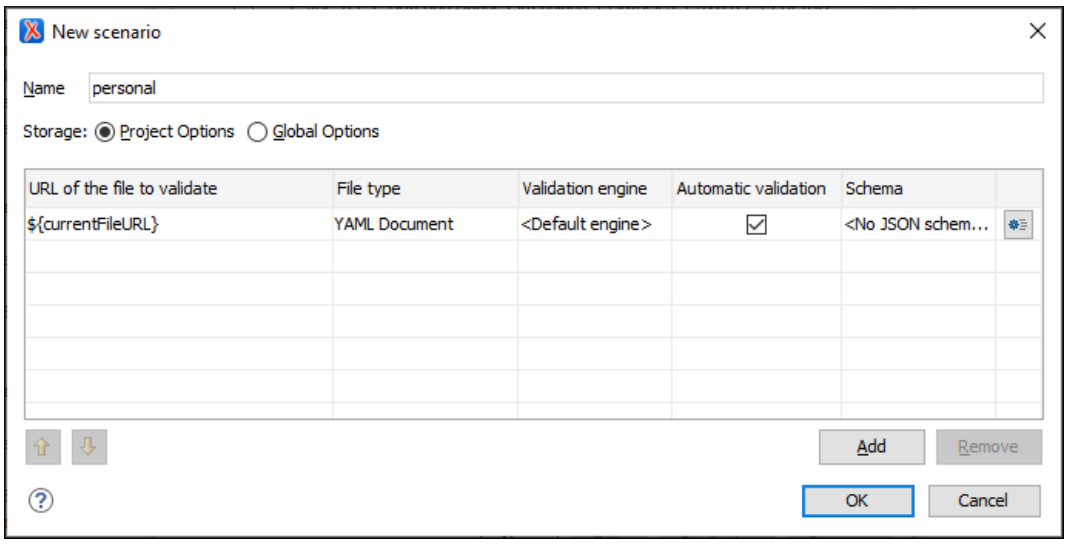
1. Select the **Configure Validation Scenario(s)** action in one of the following ways:
   - From the **Validation** toolbar drop-down menu.
   - From the **Document > Validate** menu.
   - From the **Validate** submenu, when invoking the contextual menu on a file in the **Project** view *(on page 334)*.

   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.

Figure 343. 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

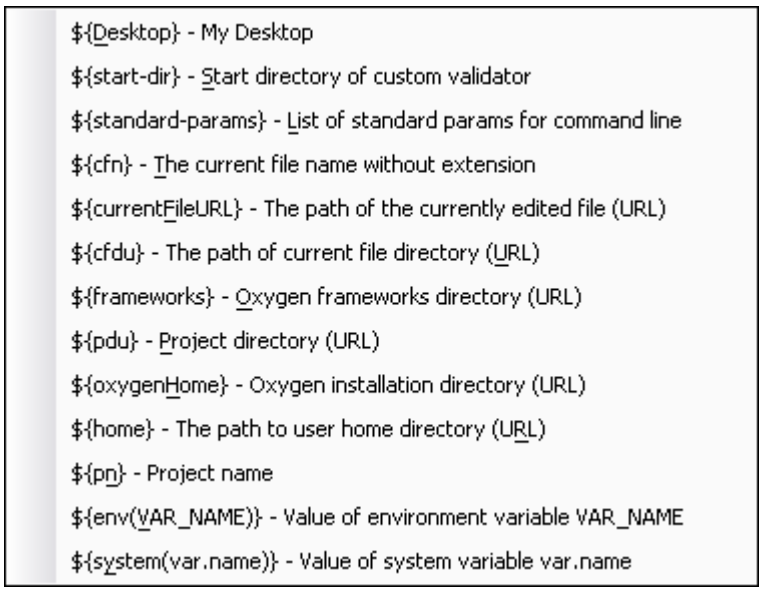
You can choose between storing the scenario in the **Project Options** *(on page 2308)* or **Global Options** *(on page 2305)*.

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 252)* or a custom editor variable *(on page 261)*.

Figure 344. Insert an Editor Variable



File type

The type of the document that is validated in the current validation unit. Oxygen XML Developer 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. If the **Automatic validation** feature is disabled in the Document Checking preferences page *(on page 160)*, 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:
Associating a JSON Schema Directly in YAML Documents *(on page 928)*

Associating a JSON Schema Directly in YAML Documents

Associate Schema Action

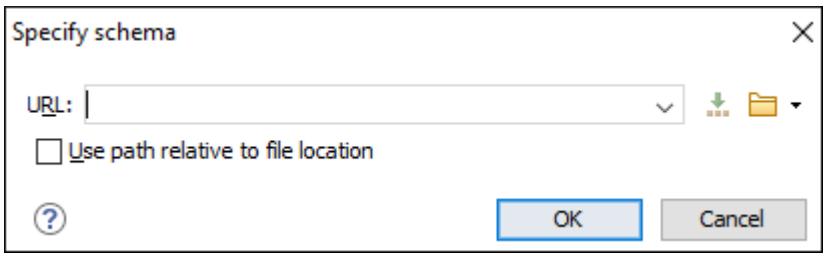
The schema used by the validation engine can be associated with the current document by using the **Associate Schema** action. The association can specify a relative file path or a URL of the schema.

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 345. Associate Schema Dialog Box

   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) or SFTP (or a custom protocol *(on page 1890)*).
   - **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 925)*

Content Completion Assistant in YAML

Oxygen XML Developer includes an intelligent *Content Completion Assistant (on page 2303)* that offers proposals for inserting YAML 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 79)*, go to **Editor > Content Completion**, and deselect the **Enable content completion** option *(on page 144)*.

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 2303)* depend on the associated JSON schema. For information about ways to associate a schema to a YAML document, see Associating a JSON Schema Directly in YAML Documents *(on page 928)*.

Using the Content Completion Assistant in YAML

The feature is activated in **Text** mode for YAML documents by pressing **Ctrl + Space** or **Alt + ForwardSlash (Command + Option + ForwardSlash on macOS)**.

You can navigate through the list of proposals by using the **Up** and **Down** keys on your keyboard. 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**.

Content Completion Options for YAML

The content that is inserted by the content completion mechanism in YAML is generated automatically from the associated schema. You can control the inserted content with options available in the **Options > Preferences > Editor > Content Completion > YAML** preferences page *(on page 149)*. You can specify whether or not required content, optional content, and additional content should be generated.

Syntax Highlighting in YAML Documents

Oxygen XML Developer 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 79)*.
2. Go to **Editor > Syntax Highlight** *(on page 158)*.
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 158)*

Folding in YAML Documents

In a large YAML document, the data corresponding to complex keys can be collapsed so that only the needed data remains in focus. The usual *folding features available for XML documents* are also available in YAML documents.

Formatting/Indenting YAML Documents

Oxygen XML Developer 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:

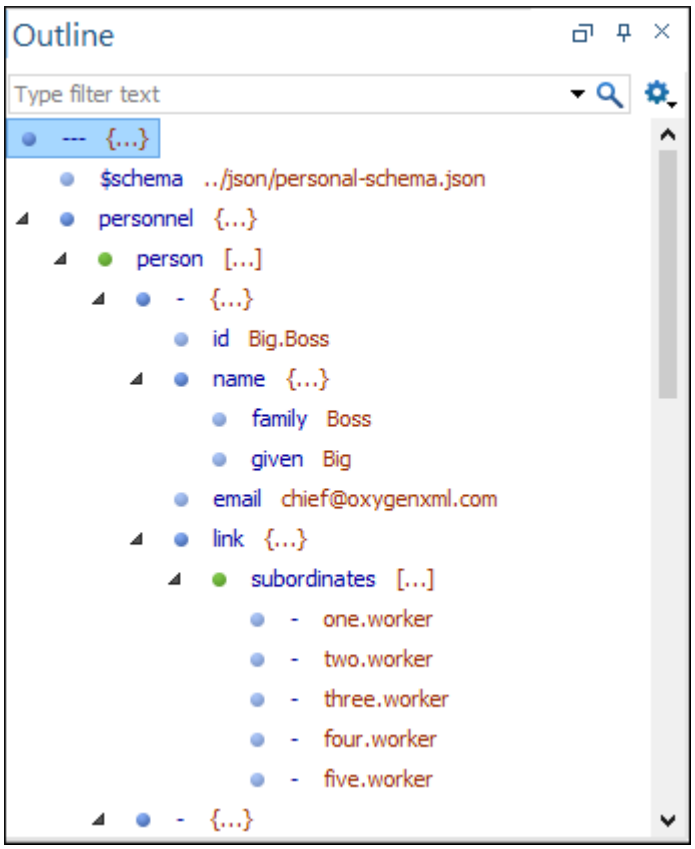
- Indents the document with the indent size specified in the **Editor > Format** preferences page *(on page 136)*.
- Removes empty lines and extra spaces between keys and values.
- Compacts the string values (for example, *description*) and limits it to 80 characters on a row.

To batch format/indent multiple YAML files, select and right-click the files in the **Project** view, then select **Format and Indent Files**.

YAML Outline View

The **Outline** view for YAML documents displays the list of all the components of the document you are editing in a hierarchical (tree-like) representation. It is synchronized with the main editor so you can use the view to navigate to specific parts of the document and move or delete 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.

Figure 346. YAML Outline View



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.

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.

Contextual Menu Actions

The following actions are available in the contextual menu of the YAML **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.

YAML to JSON Converter

Converting YAML to JSON in Oxygen

Oxygen XML Developer 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.

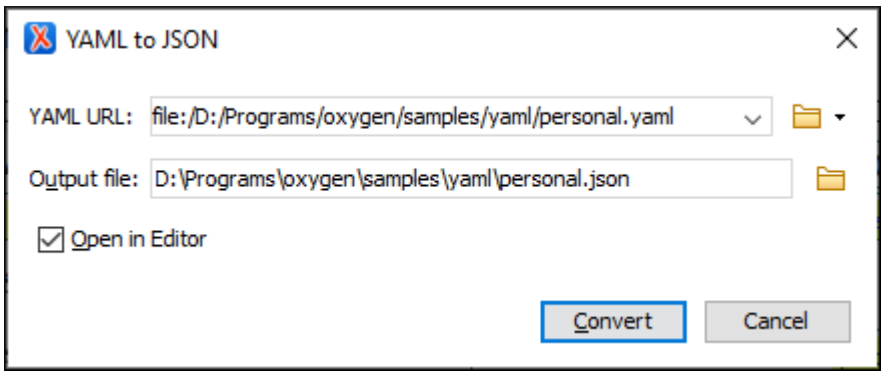
The **YAML to JSON** action for invoking the tool can be found in the **Tools > JSON Tools** menu.

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:

Figure 347. 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 879)*

JSON to YAML Converter

Converting JSON to YAML in Oxygen

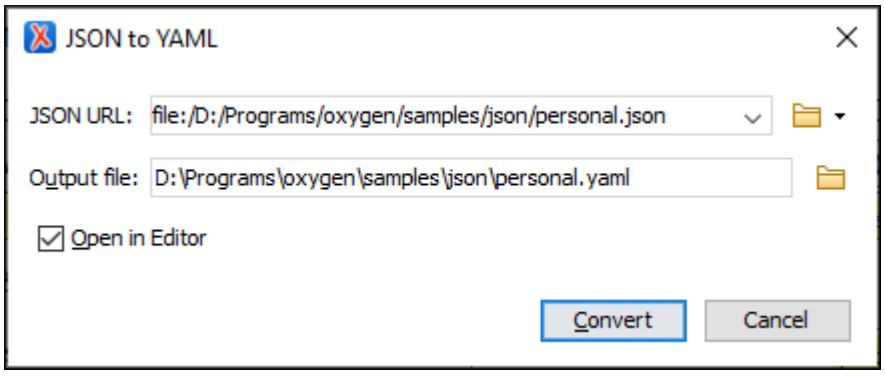
Oxygen XML Developer includes a useful and simple tool for converting JSON files to YAML. The **JSON to YAML** action for invoking the tool can be found in the **Tools > JSON Tools** menu.

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:

Figure 348. 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 879)*

Contextual Menu Actions in YAML Documents

In addition to the usual text editing actions being available in the YAML editor, it also includes some unique editor-specific actions available 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.

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 Line Comment (Ctrl + ForwardSlash (Command + ForwardSlash on macOS))

Allows you to comment (or uncomment) the current selection (or current line if no content is selected) in the YAML document you are editing.

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 391)*. 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 `ɩ`
- `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 394)*.

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 124)* 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 236)*.

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 124)* 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 236)*.

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 124)* 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 236)*.

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 124)* 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 236)*.

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 124)* 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 236)*.

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 124)* 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 236)*.

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 + Option + 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 key (if a JSON schema is associated with the YAML document and that schema contains a definition for the key).

Associating a JSON schema can be done using either of the following methods:

- By creating a YAML validation scenario *(on page 925)*.
- By directly associating a JSON schema in the YAML document *(on page 928)*.

> **Tip:**
> Holding down **CTRL (Command on macOS)** while hovering over a YAML *key-value* pair reveals a hyperlink that navigates to the key definition in the associated JSON schema (if a definition for that key is found).

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 397)*.

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 402)*.

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 Developer 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 299)*, Oxygen XML Developer 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 Developer 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 Developer JavaScript Editor and how you can use them.

JavaScript Editing Actions

Oxygen XML Developer 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 349. JavaScript Editor Text Mode

```
90    function change_sides(front) {
91      switch ($('#version-switch').text()) {
92        case 'Original':
93          $('#holder').html($('div .original[id]').html());
94          make_clickable();
95          $('#version-switch').text('Translation 1');
96          break;
97        case 'Translation 1':
98          $('#holder').html($('div .translation[id]').filter(':first').html());
99          $('#version-switch').text('Translation 2');
100         break;
101       case 'Translation 2':
102         $('#holder').html($('div .translation[id]').filter(':last').html());
103         $('#version-switch').text('Original');
104         break;
105       }
106     }
```

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 Developer 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 2305)* triangles in the stripe on the left side of the editor, the following actions are available:



Collapse Other Folds

Folds all the elements except the current element.



Collapse Child Folds

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

Unfolds all elements in the current document.

Validating JavaScript Files

You have the possibility to validate the JavaScript document you are editing. Oxygen XML Developer 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 Developer 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 2303)* presents you a list of the elements you can insert at the cursor position. It can be manually activated with the **Ctrl + Space** 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:

- - function
- - variable
- - object
- - property
- - method

Note:
These icons decorate both the elements from the content completion list of proposals and from the **Outline** view *(on page 945)*.

Figure 350. JavaScript Content Completion Assistant



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 Developer 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 79)*.
2. Go to **Editor > Syntax Highlight** *(on page 158)*.
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.

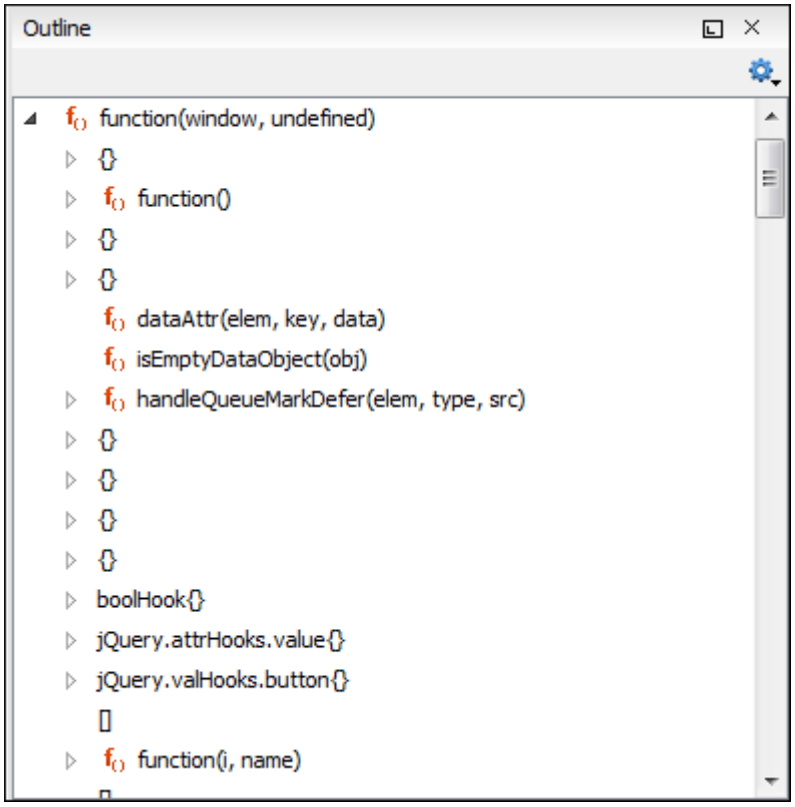
Related Information:

Syntax Highlight Preferences *(on page 158)*

JavaScript Outline View

Oxygen XML Developer 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.

Figure 351. JavaScript Outline View



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 validation 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 *XML Calabash 3.X* engine that comes bundled with Oxygen XML Developer version 28.1. The default engine offers content completion, validation, and transformation support for XProc 1.0, 3.0, and 3.1 files.

Note:
If a custom engine is used, the validation support for XProc version 3.x depends on whether it is available for the particular custom engine.

Notice:
Oxygen XML Developer supports running XSLT stylesheets that contain AI functions. For information about the supported AI functions, see Oxygen AI Positron: Tool Calls.

XProc Content Completion

Oxygen XML Developer helps you edit a XProc scripts through the *Content Completion Assistant (on page 2303)*, offering proposals that are valid at the cursor position. It can be manually activated with the **Ctrl + Space** shortcut.

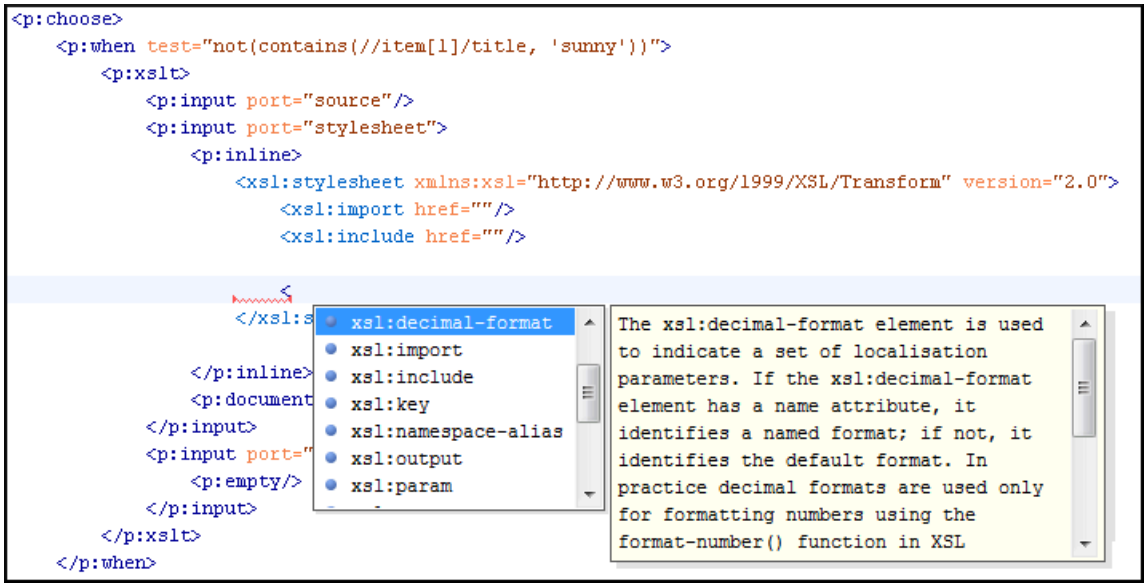
Note:
Variable names are offered after the **$** character inside XPath attributes, depending on context.

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.

Tip:
XPath 3.0 functions are also available in the *Content Completion Assistant* for XProc 3.0 documents.

Figure 352. 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 79)*.
2. Go to **Editor > Syntax Highlight** *(on page 158)*.
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

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:

```
<system-property name="com.xmlcalabash.allow-text-results" value="true"/>
```

Related Information:
Creating an XProc Transformation Scenario *(on page 1184)*
Integrating an External XProc Engine *(on page 1189)*
XProc Preferences *(on page 178)*

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 Developer 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 968)*. You can create a new Schematron schema using one of the Schematron templates available in the **New** document wizard *(on page 299)*.

For information about applying and detecting Schematron schemas, see Associating a Schema to XML Documents *(on page 554)*.

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 962)* 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 79)*, go to **XML > XML Parser > Schematron**, and select it in the **Embedded rules query language binding** option *(on page 175)*.
- 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 79)*, go to **XML > XML Parser > Schematron**, and select the version in the **XPath Version** option *(on page 175)*.

Multi-Lingual Support in Schematron Messages

You can specify the desired language for the validation messages in the **Schematron Preferences** page *(on page 174)*. 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:

```
<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 79)* and go to **Editor > Syntax Highlight**.

Schematron Transformation Scenario

When you create a Schematron document, Oxygen XML Developer 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.

Using Schematron with AI

An **Oxygen AI Positron** add-on is available that provides support for helping writers generate content by using the **Oxygen AI Positron** service. This add-on also contributes two XPath extension functions that can be used from custom Schematron schemas to rephrase content or to perform validation checks on existing content. The `ai:transform-content(instruction, content)` function can be used to automatically transform content using AI and the `ai:verify-content(instruction, content)` function can be used to automatically validate content using AI.

For more details, see Oxygen AI Positron: Custom Validation Rules.

Resources

For more information about the Schematron support in Oxygen XML Developer, watch our video demonstrations:

https://www.youtube.com/embed/HdcZA3DJi7E

https://www.youtube.com/embed/y3u3wlO92e4

https://www.youtube.com/embed/FQNSsg57S4E

Related Information:
Editing XML Documents in Text Mode *(on page 442)*
Associating a Schema to XML Documents *(on page 554)*

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 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 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:

```
<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:

```
<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:

```
<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:

```
<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:

```
<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:

```
<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:

```
<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:

```
<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:

```
<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:

```
<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:

```
<sch:schema xmlns:sch="http://purl.oclc.org/dsdl/schematron" queryBinding="xslt2">
    <sch:let name="typeValue" value="//Node1[not(@version >
```

```
                    ../Node1/@version)][1]/Type/text()"/>


    <sch:pattern>
        <sch:rule context="Type">
            <sch:assert test="text() = $typeValue">
                The Type value must be "<sch:value-of select="$typeValue"/>"
            </sch:assert>
        </sch:rule>
    </sch:pattern>
</sch:schema>
```

where the XML file would look something like this:

```
<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:

```
<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=""> </revised>
            </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=""> </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>
  </sch:pattern>
</sch:schema>
```

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:

```
<<sch:schema xmlns:sch="http://purl.oclc.org/dsdl/schematron"
  queryBinding="xslt2"
  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>
        <!-- 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>
```

```
      <!-- 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>
```

```
                <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/>
    </sqf:fix>
  </sch:rule>
 </sch:pattern>



 <!-- Template used to copy the current node -->
 <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"/>
</sch: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:

```
<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: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>
</sch:schema>
```

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 `<colspec>`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 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="colspec">
            <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>
```

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 Developer 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 cat set a main Schematron document either using the *main files* support from the **Project** view *(on page 350)*, 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 Developer 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 2303)* 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 Developer window. Each error is flagged with a severity level that can be: *warning*, *error*, *fatal* or *info*.

To set a severity level, Oxygen XML Developer 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 Developer 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 Developer 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 160)*.

Related Information:
Validating XML Documents Against a Schema *(on page 516)*
Validation Scenario *(on page 525)*
Associating a Schema to XML Documents *(on page 554)*
Presenting Validation Errors in Text Mode *(on page 519)*

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 991)* 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 2305)* (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 79)* and go to **Document Type Association > Locations** *(on page 94)*. 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 93)* 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).
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 93)*.
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.

Related Information:
Defining Schematron Quick Fixes *(on page 991)*
Associating a Schema in Validation Scenarios Defined in the Document Type *(on page 559)*

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 79)*, go to **Editor > Document Checking**, and deselect the **Enable automatic validation** option *(on page 160)*.

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 Developer 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 Developer 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 962)*

Content Completion in Schematron Documents

Oxygen XML Developer helps you edit a Schematron schema through the *Content Completion Assistant (on page 2303)*, offering proposals that are valid at the cursor position. It can be manually activated with the **Ctrl + Space** shortcut.

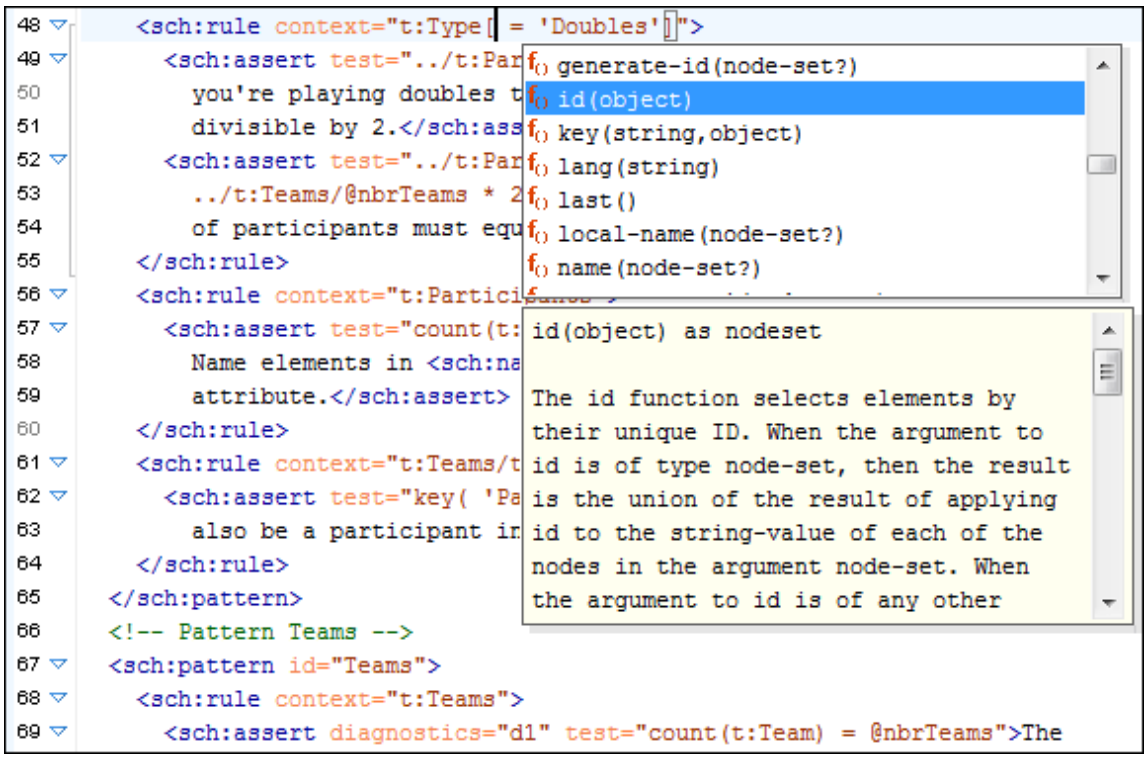
When you edit the value of an attribute that refers a component, the proposed components are collected from the entire schema hierarchy. For example, if the editing context is `phase/active/@pattern`, the *Content Completion Assistant* proposes all the defined patterns.

Note:
For Schematron resources, the *Content Completion Assistant* collects its components starting from the *main files (on page 2306)*. 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 350)*.

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 174)* that are set in Preferences pages. If the Saxon 6.5.5 namespace (`xmlns:saxon="http://icl.com/saxon"`) or the Saxon 12.9 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 353. XSLT Extension Functions in Schematron Schema Content Completion



The *Content Completion Assistant* also includes code templates that can be used to quickly insert code fragments *(on page 462)* into Schematron documents.

Syntax Highlighting in Schematron

Oxygen XML Developer 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 79)*.
2. Go to **Editor > Syntax Highlight** *(on page 158)*.
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 Developer 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 159)*.

Related Information:
Syntax Highlight Preferences *(on page 158)*

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 Developer 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-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 Developer 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

```
<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

```
<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>
    <start>
     ..............
    </start>
</grammar>
```

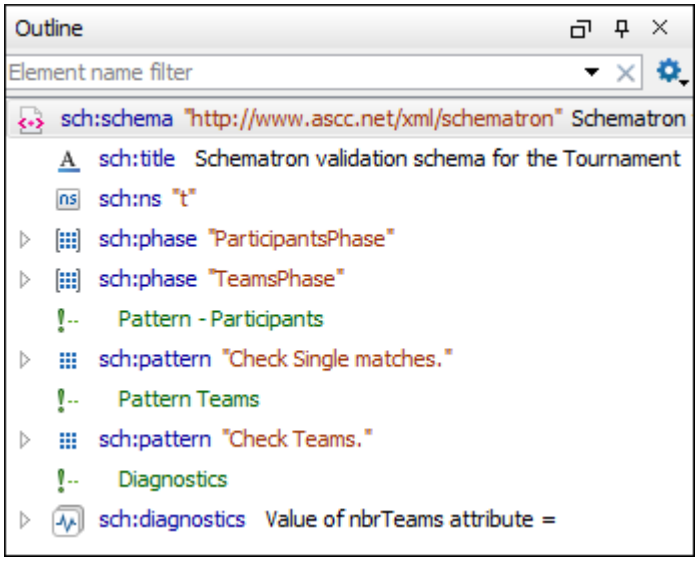
Related Information:

Embedding Schematron Quick Fixes in Relax NG or XML Schema *(on page 1005)*

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.

Figure 354. Schematron 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 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 235)*.

Configure displayed attributes

Displays the XML Structured Outline preferences page *(on page 235)*.

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.

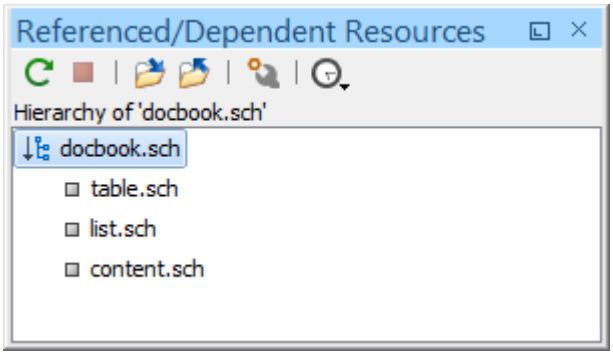
Schematron Referenced/Dependent Resources View

The **Referenced/Dependent Resources** 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 334)* and choose **Show referenced resources** from the contextual menu.

If you want to see the dependencies of a schema, select the desired schema in the **Project** view *(on page 334)* and choose **Show dependent resources** from the contextual menu.

Figure 355. Referenced/Dependent Resources View



The following actions are available on the toolbar of the **Referenced/Dependent Resources** view:

Refresh

Refreshes the resource structure.

Stop

Stops the computing.

Show hierarchy for

Computes the hierarchical structure of the references for a resource.

Show dependencies for

Computes the structure of the dependencies for a resource.

Configure dependencies search scope

Allows you to configure a scope to compute the dependencies. 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 **Referenced/Dependent Resources** view contains the following actions:

Open

Opens the resource. You can also double-click a resource within the hierarchical 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.

Show references resources

Shows the references for the selected resource.

Show dependent resources

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 hierarchical structure.

Collapse All

Collapses all children of the selected resource from the hierarchical structure.

Tip:
When a recursive reference is encountered in the 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 972)*.

Moving/Renaming Schematron Resources

You can move and rename a resource presented in the **Referenced/Dependent Resources** 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 **Referenced/Dependent Resources** 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 **Referenced/Dependent Resources** 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 Developer 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 79)* 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 474)*.

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:

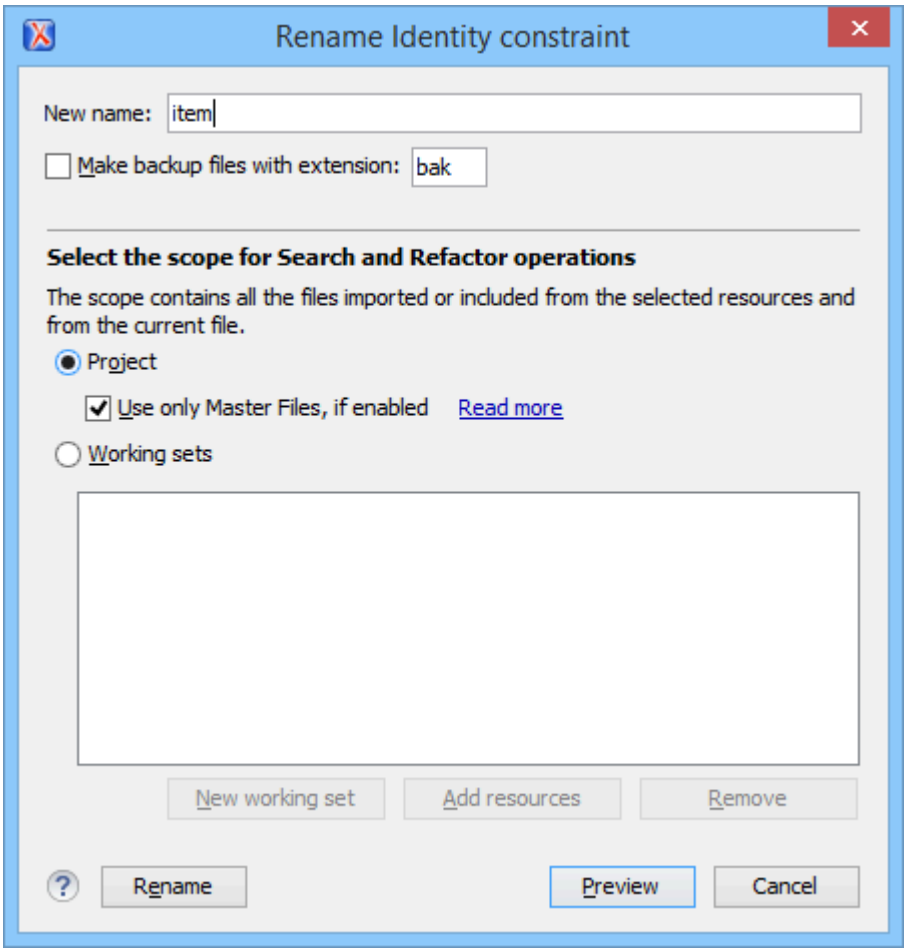
Rename Component

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.

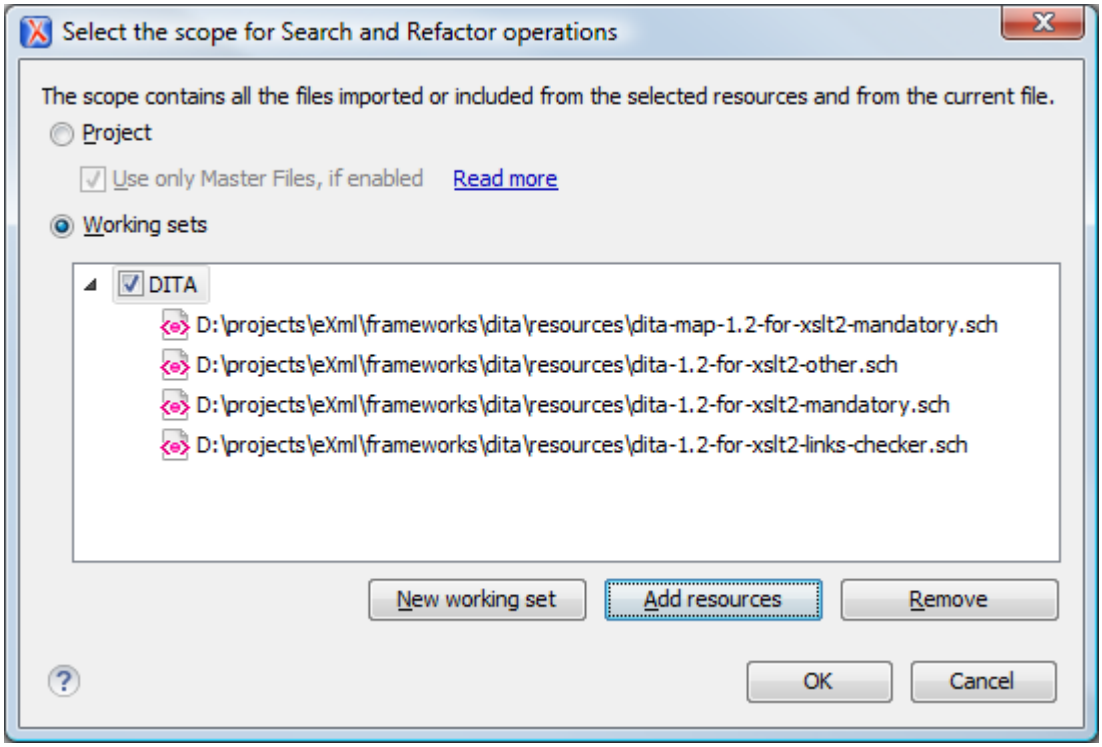
Figure 356. Rename Identity Constraint Dialog Box



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 **Referenced/ Dependent Resources** view's toolbar. You can restrict the scope to the current project or to one or multiple *working sets (on page 2309)*. 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 350)*.

Figure 357. Change Scope Dialog Box



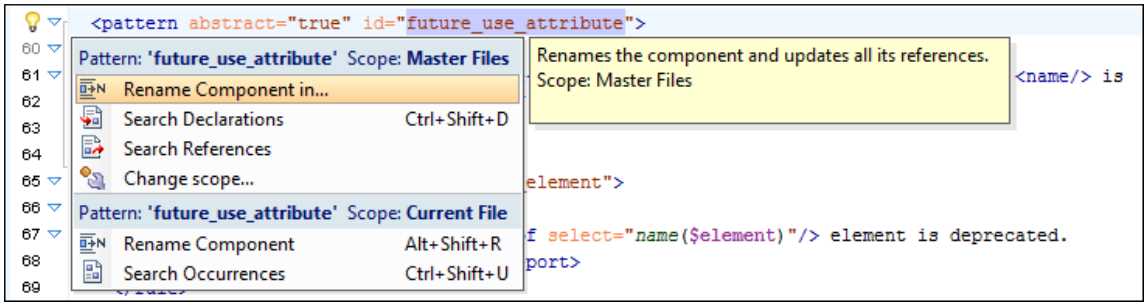
The scope you define is applied to all future search and refactor operations until you modify it. Contextual menu actions allow you to add or delete files, folders, and other resources to the *working set (on page 2309)* structure.

Quick Assist Support in Schematron Documents

The *Quick Assist* support *(on page 2308)* 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 2308)* 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.

Figure 358. Schematron Quick Assist Support



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.

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 2305)*.

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:

```
<sch:pattern>
  <sch:rule context="section">
    <sch:assert test="title" id="a002">
      section should have a title
    </sch:assert>
  </sch:rule>
</sch:pattern>
```

The XSpec test could look like this:

```
<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 330)* in Oxygen XML Developer, create a `catalog.xml` file in the project directory. This catalog will then be loaded automatically.
- Edit *(on page 1207)* the **Run XSpec Test** transformation scenario, go to the **Parameters** tab *(on page 1143)*, and set the value of the `catalog` parameter to the location of your catalog file.

Editing Schematron Quick Fixes

Oxygen XML Developer provides support for editing the Schematron *Quick Fixes (on page 553)* (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 Developer assists you in editing Schematron *Quick Fixes* with schema-based content completion, syntax highlighting, and validation as you type.

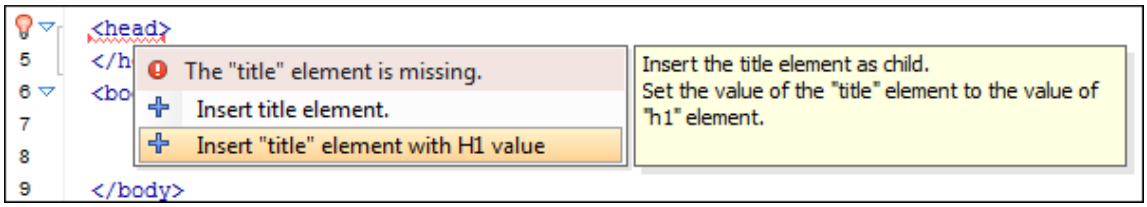
A typical use case is using Schematron *Quick Fixes* to assist content authors with common editing tasks. For example, you can use Schematron rules to automatically report certain validation warnings (or errors) when performing regular editing tasks, such as inserting specific elements or changing IDs 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 554)*.

Displaying the Schematron Quick Fix Proposals

The defined Schematron *Quick Fixes* are displayed on validation errors in **Text** mode.

Figure 359. Example of a Schematron Quick Fix



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:

- 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 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 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:

```
<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:

```
<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:

```
<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:

```
<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:

```
<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:

```
<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:

```
<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:

```
<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:

```
<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:

```
<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:

```
<sch:schema xmlns:sch="http://purl.oclc.org/dsdl/schematron" queryBinding="xslt2">
    <sch:let name="typeValue" value="//Node1[not(@version >
                    ../Node1/@version)][1]/Type/text()"/>

    <sch:pattern>
        <sch:rule context="Type">
            <sch:assert test="text() = $typeValue">
                The Type value must be "<sch:value-of select="$typeValue"/>"
            </sch:assert>
        </sch:rule>
    </sch:pattern>
</sch:schema>
```

where the XML file would look something like this:

```
<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:

```
<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=""> </revised>
            </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=""> </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>
  </sch:pattern>
</sch:schema>
```

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:

```
<<sch:schema xmlns:sch="http://purl.oclc.org/dsdl/schematron"
  queryBinding="xslt2"
  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>
      <!-- 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>
      <!-- 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>
                  <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/>
      </sqf:fix>
    </sch:rule>
```

```
  </sch:pattern>



  <!-- Template used to copy the current node -->
  <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"/>
</sch: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:

```
<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: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>
</sch:schema>
```

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 `<colspec>`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 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="colspec">
            <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

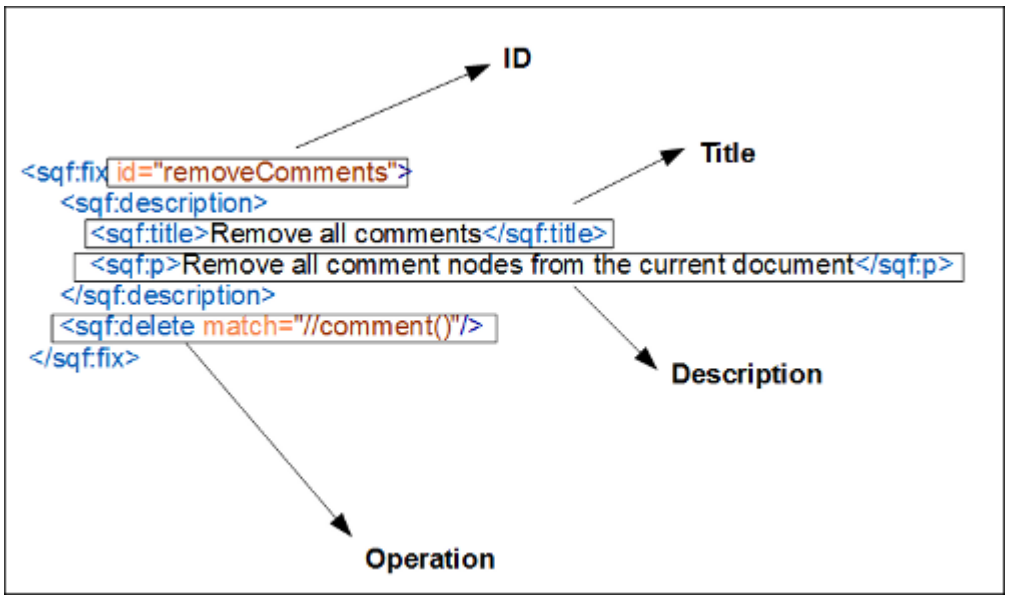
You can define and customize Schematron *Quick Fixes (on page 2308)* directly in the current Schematron file or in a separate Schematron file. The Schematron *Quick Fixes* are an extension of the Schematron language and they allow you to define fixes for Schematron error messages. You can reference the *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 992)* 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 360. 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 997)*.

<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

The `sqf:default-fix` attribute is also supported and taken into account when invoking the **Apply all default quick fix proposals** action from:

- The contextual menu in **Text** mode *(on page 492)*.
- The **Tools** menu *(on page 616)*.
- The **Refactoring** sub-menu in the contextual menu of the **Project** view *(on page 616)*.
- The contextual menu in the **Results** view *(on page 476)*.
- The Quick Fix menu *(on page 2308)*.

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 2308)*: **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 Developer will insert the required element content.

An Example of the `<sqf:add>` Element:

```
<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 2308)* 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 2308)* 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:

```
<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:delete match="@xml:lang"/>
            </sqf:fix>
        </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:

```
<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">
                ph element is not allowed in title.</report>
            <sqf:fix id="resolvePh">
                <sqf:description>
                    <sqf:title>Change the ph element into text</sqf:title>
                </sqf:description>
                <sqf:replace match="ph">
                    <value-of select="."/>
                </sqf:replace>
            </sqf:fix>
        </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 2308)* 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 ***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 `\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 `<sqf:stringReplace>` element always have the *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 `@regex` attribute).

Attention:
The context of the content within the `<sqf:stringReplace>` element is set to the whole text node, rather than the current sub-string.

An Example of the `<sqf:stringReplace>` Element:

```
<?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:rule>
```

```
    </sch:pattern>
</sch:schema>
```

Related Information:

User Entry SQF Operation *(on page 997)*
Restricting Quick Fix Operations *(on page 997)*
Examples of Schematron Rules and Quick Fixes *(on page 950)*

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 Developer 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:

```
<sqf:fix id="editTitle">
    <sqf:description>
        <sqf:title>Edit the journal title</sqf:title>
    </sqf:description>
    <sqf:user-entry name="newTitle" default="@title">
        <sqf:description>
            <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>
```

Related Information:

Basic Schematron Quick Fix Operations *(on page 992)*
Examples of Schematron Rules and Quick Fixes *(on page 950)*

Restricting Quick Fix Operations

To restrict a *Quick Fix (on page 2308)* 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:

```
<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>
   <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 992)*

Formatting/Indenting Content Inserted by SQF Operations

Content that is inserted by the **Add**, **Replace**, or **String Replace** Schematron *Quick Fix (on page 2308)* 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:

```
<sqf:add position="last-child">
    <row><xsl:text>
    </xsl:text>
        <entry>First column</entry><xsl:text>
        </xsl:text>
        <entry>Second column</entry><xsl:text>
        </xsl:text>
    </row><xsl:text>
    </xsl:text>
</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:

```
<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 992)*

Executing Schematron Quick Fixes in Other Documents

You can apply Schematron *Quick Fixes (on page 2308)* 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 2308)* 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:fix>
</sch:rule>
```

Localizing SQF Messages

Oxygen XML Developer 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 175)*. 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 Developer, 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.

```
<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"> Fugen Sie einen Knochen hinzu </sch:diagnostic>
  <sch:diagnostic id="fix_desc_de"> Fugen Sie ein Knochenelement als
    untergeordnetes Element hinzu </sch:diagnostic>
</sch:diagnostics>
```

Integrating SQF in a Framework and Sharing Them

You can use Schematron *Quick Fixes (on page 2308)* to assist your content authors by imposing rules for an entire *framework (on page 2305)* (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 (`<fig>` element) inside a paragraph (`<p>` element) that contains other content since it may result in undesirable placement or spacing in the output. The Schematron rule and its *Quick Fix* for this particular use-case could look like this:

```
<schema xmlns="http://purl.oclc.org/dsdl/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>
                 <sqf:title>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 `<fig>` element inside a `<p>` element and they are presented with the option of selecting the *Quick Fix* that would move the figure outside the paragraph.

How to Integrate SQF in a Framework

To integrate a Schematron *Quick Fix* in a *framework (on page 2305)*, 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 *Quick Fix* for a rule *(on page 991)* 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 79)* and go to **Document Type Association > Locations** *(on page 94)*. 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 93)* 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).
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 93)*.
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 559)*.
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.

Related Information:
Defining Schematron Quick Fixes *(on page 991)*
Basic Schematron Quick Fix Operations *(on page 992)*
Associating a Schema in Validation Scenarios Defined in the Document Type *(on page 559)*

Validating Schematron Quick Fixes

By default, Schematron *Quick Fixes (on page 2308)* 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 79)*, go to **Editor > Document Checking**, and deselect the **Enable automatic validation** option *(on page 160)*.

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 516)*
Validation Scenario *(on page 525)*
Presenting Validation Errors in Text Mode *(on page 519)*

Content Completion in SQF

Oxygen XML Developer helps you edit Schematron *Quick Fixes (on page 2308)* embedded in a Schematron document by offering proposals that are valid at the cursor position in a *Content Completion Assistant (on page 2303)*. It can be manually activated with the **Ctrl + Space** 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 174)* 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 2308)* ID in a Schematron document, Oxygen XML Developer 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 79)* 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 474)*.

Searching and Refactoring Operations in SQF

Search Actions

The following search actions can be applied on *Quick Fix (on page 2308)* 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:

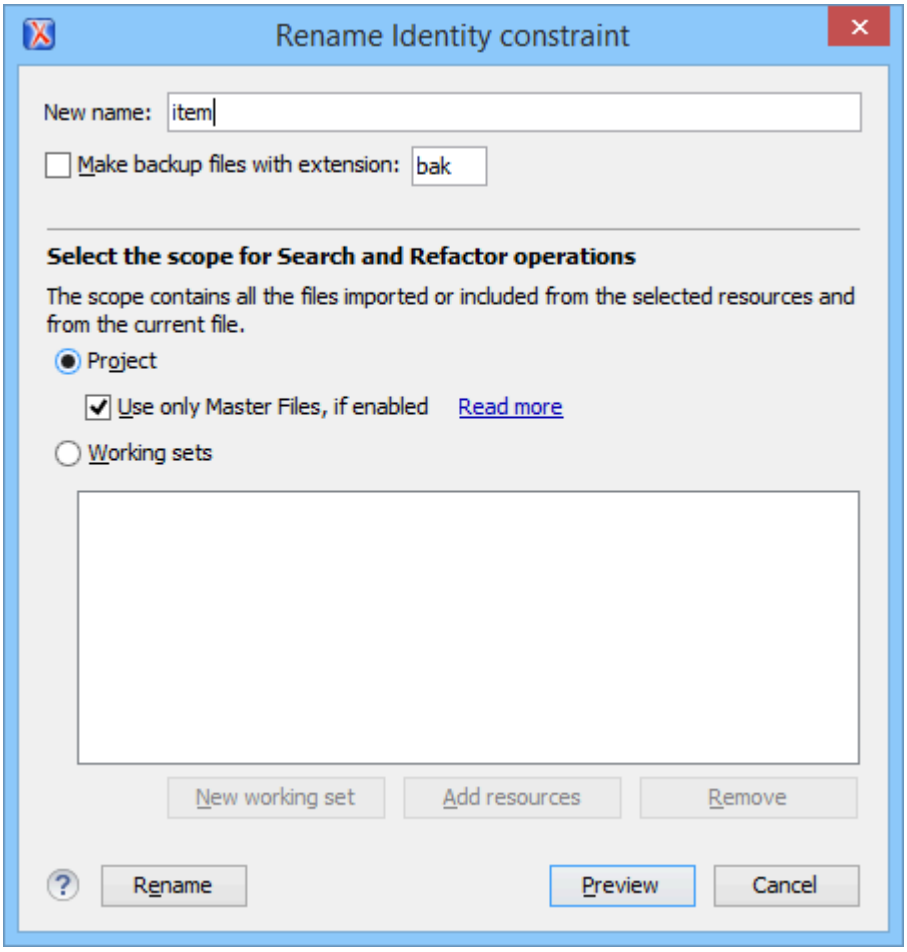
Rename Component

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.

Figure 361. Rename Identity Constraint Dialog Box



Embedding Schematron Quick Fixes in Relax NG or XML Schema

Schematron *Quick Fixes (on page 2308)* 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 967)*.

Oxygen XML Developer 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 516)*.
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

```
<xsd:appinfo>
  <sch:pattern>
    <sch:rule context="...">
      <sch:assert test="..." sqf:fix="fixId">Message.</sch:assert>
        <sqf:fix id="fixId">
```

```
          ......
        </sqf:fix>
    </sch:rule>
  </sch:pattern>
</xsd:appinfo>
```

Example: Embedded Schematron Quick Fix in Relax NG

```
<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 967)*
Defining Schematron Quick Fixes *(on page 991)*

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 2305)* (including DocBook) have support for SVG by defining the graphics directly in the document.

Oxygen XML Developer 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 564)* 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 Developer `lib` folder.
2. Restart the application.

SVG 1.2 Rendering Issues

Oxygen XML Developer 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.

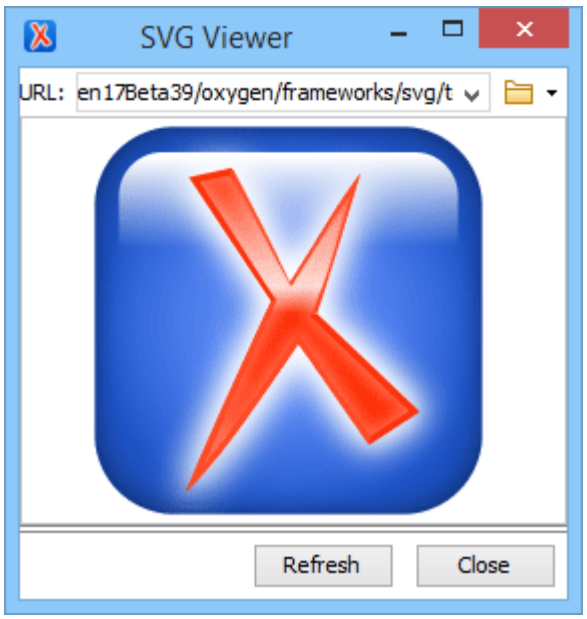
This partial support could lead to some rendering issues in Oxygen XML Developer. For example, if you are using the *Inkscape* SVG editor, it is possible for it to save the SVG as 1.1, while using SVG 1.2 elements (such as `<flowRoot>`) inside it. This means that the image will not be properly rendered inside the application.

Standalone SVG Viewer

Oxygen XML Developer includes a simple **SVG Viewer** that allows you to work with SVG images.

To open the viewer, select **SVG Viewer** from the **Tools** menu.

Figure 362. SVG Viewer



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 334)* 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 in 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 1006)*

Integrated SVG Viewer in the Results Panel

Oxygen XML Developer 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 474)* 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 1112)*. When you run the transformation, the SVG result is then rendered in an integrated SVG viewer in the **Results** panel *(on page 474)*.

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 Developer 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 1102)*.
4. While configuring the transformation scenario, select **Show in results view as > SVG** in the **Output** tab *(on page 1112)* 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 474)* at the bottom of the editor.

Figure 363. Integrated SVG Viewer



Editing HTML Documents

Oxygen XML Developer 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.

Oxygen XML Developer also includes a built-in XHTML framework *(on page 1058)* (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 Developer also includes support for importing HTML files as an XML document *(on page 1822)*.

Resources

For more information about the HTML support in Oxygen XML Developer, see the following resources:

- Webinar: **HTML5 Support in Oxygen**.
- Video: **HTML Support in Oxygen**.

Related information
XHTML Document Type (Framework) *(on page 1058)*

HTML Editor

Oxygen XML Developer 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 Developer includes a new document template to help you get started creating HTML content. It is available when creating new documents from templates *(on page 299)* 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 using all of its useful features. It includes content completion based on a special HTML schema, syntax highlighting *(on page 1013)*, a specialized Outline view *(on page 1014)* that presents the structure, folding support *(on page 1014)*, 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 481)*.

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

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).

HTML Validation

Oxygen XML Developer 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. The validator in Oxygen XML Developer only supports HTML5 structure. It presents the errors in the editor similar to XML documents. It also checks the embedded CSS content and the warnings and errors are presented similar to the CSS editor *(on page 817)*.

Validating HTML Against a Schematron

It is also possible to validate HTML documents against a Schematron schema. Besides the default HTML validator, Oxygen XML Developer 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 Developer 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

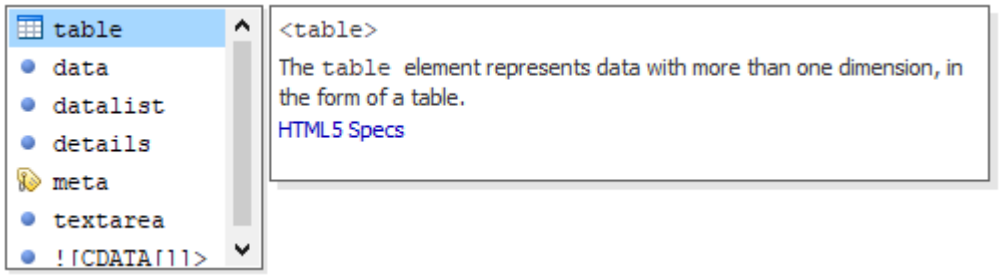
If your HTML document is XML well-formed, you could also configure a validation scenario to validate it as an XML document against other types of schemas. You would create or edit a validation scenario, make sure the **File type** column is set to *XML Document*, select the appropriate **Validation engine**, and specify the schema document in the **Schema** column.

HTML Content Completion Assistant

Oxygen XML Developer includes an intelligent *Content Completion Assistant (on page 2303)* 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 79)*, go to **Editor > Content Completion**, and deselect the **Enable content completion** option *(on page 144)*.

Figure 364. Content Completion Assistant in HTML



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 458)*

Code Templates in the Content Completion

Oxygen XML Developer 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 462)*.

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 Developer 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 79)*.
2. Go to **Editor > Syntax Highlight** *(on page 158)*.
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 158)*

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 454)* are also available in HTML documents, but it also provides folding for nested elements that are not closed.

Minifying 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

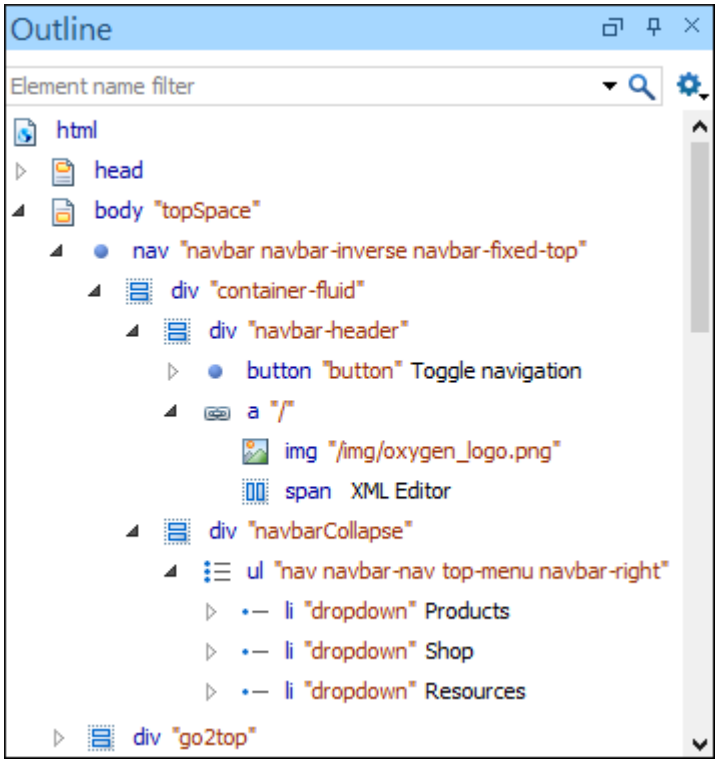
If selected (default), the resulting output file will be opened in Oxygen XML Developer.

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, 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.

Figure 365. HTML Outline View



Querying HTML Documents with XPath

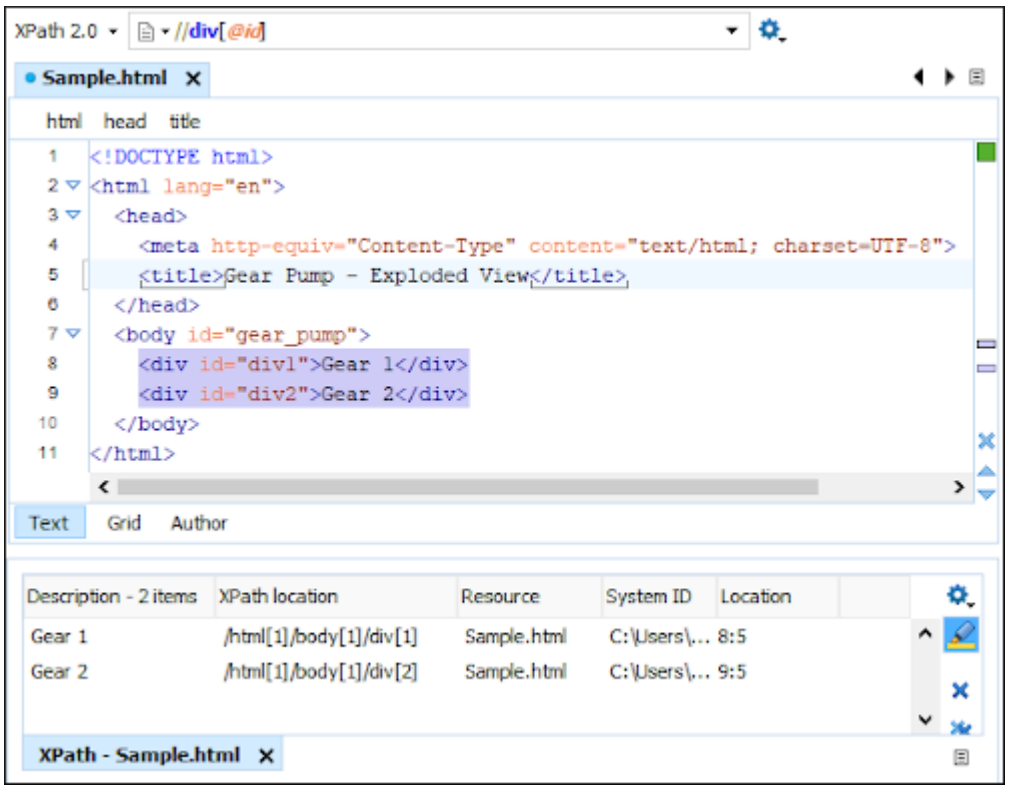
Oxygen XML Developer provides an **XPath** toolbar that makes it easy to quickly query HTML documents using XPath expressions. You can also use the dedicated **XPath Builder** view *(on page 1729)* that allows you to compose more complex XPath expressions and execute them over HTML documents (even if 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 1727)*.

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 366. XPath Toolbar for HTML



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 Developer 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 Developer 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 2304)* 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 Developer, see our webinar: **Oxygen Markdown Support**.

Markdown Editor

Oxygen XML Developer 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.

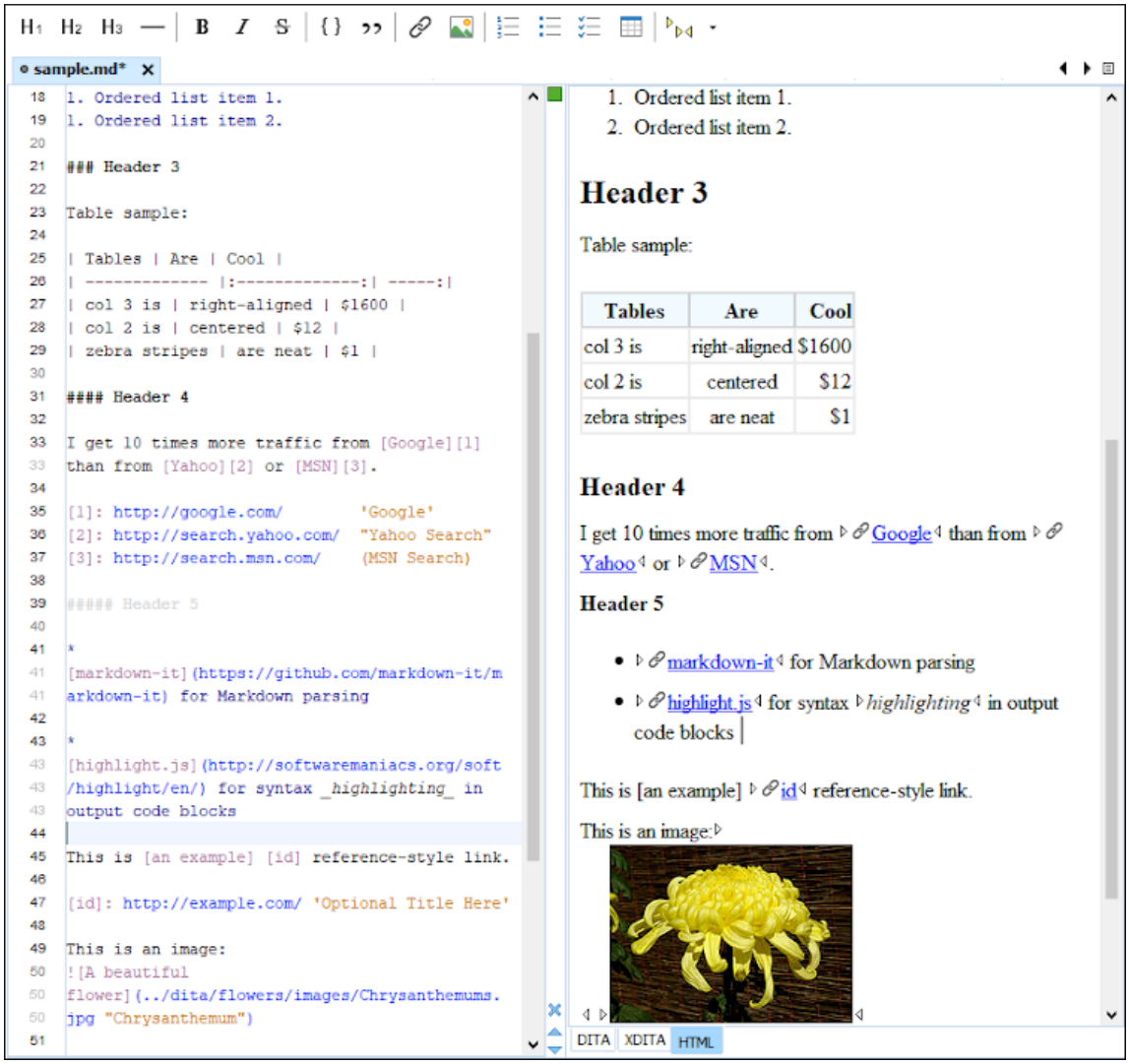
Markdown Text Editor Pane (Left Side)

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 Developer.

The features of this special editor that are unique for Markdown documents include:

- **Unique Markdown Syntax Rules** - The Markdown editor in Oxygen XML Developer uses an integration of rules *(on page 1027)* 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 Developer 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 1026)*.
- **Automatic Spell Checking** - The Markdown editor supports the Oxygen XML Developer automatic spell checking feature *(on page 388)* 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 163)*, then click the **Select editors** button and select **Markdown Editor**.
- **Helpful Toolbar and Contextual Menu Actions** - A variety of unique actions *(on page 1019)* 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 293)* in Oxygen XML Developer also work in the Markdown editor.

Figure 367. Markdown Editor



Related information

Markdown Editor Syntax Rules and Specifications *(on page 1027)*

Actions Available in the Markdown Editor *(on page 1019)*

Creating New Markdown Documents *(on page 1018)*

Creating New Markdown Documents

To create a new Markdown document in Oxygen XML Developer, 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 *(on page 1017)*.

Related Information:
Markdown Editor *(on page 1017)*

Actions Available in the Markdown Editor

Aside from the actions that are available in Oxygen XML Developer 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:

Header (1st Level)

Inserts an *atx-style* first-level header *(on page 1028)* at the cursor position.

Header (2st Level)

Inserts an *atx-style* second-level header *(on page 1028)* at the cursor position.

Header (3rd Level)

Inserts an *atx-style* third-level header *(on page 1028)* at the cursor position.

Horizontal Rule

Inserts a horizontal rule *(on page 1028)* at the cursor position.

Bold (Strong)

Marks the selected text with bold *(on page 1029)*.

Italic (Emphasis)

Marks the selected text with italics *(on page 1029)*.

Strikethrough

Marks the selected text with a strikethrough *(on page 1029)*.

Code Block

Inserts (or surrounds selected text in) a codeblock *(on page 1033)*.

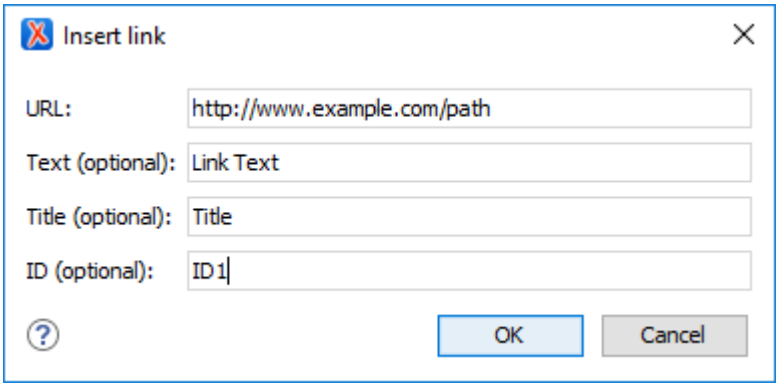
Blockquote

Inserts a blockquote *(on page 1032)* at the cursor position.

Insert Link

Opens the **Insert Link** dialog box that allows you to define a link *(on page 1030)* to insert at the cursor position.

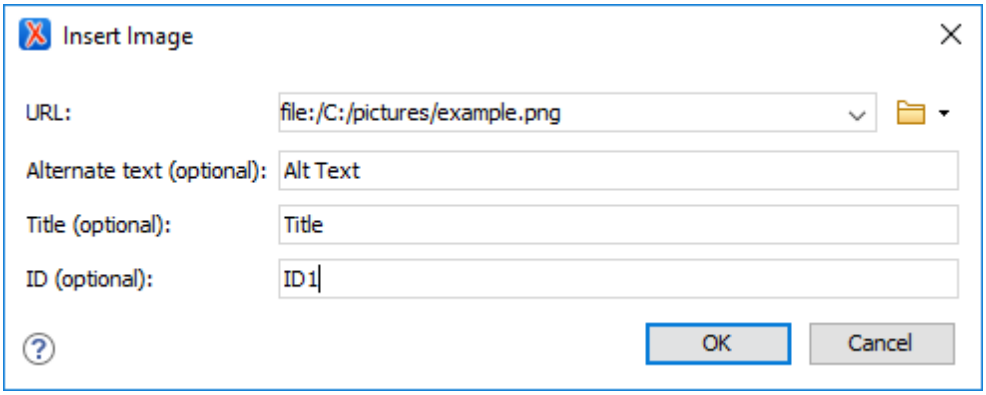
Figure 368. Insert Link Dialog Box



Insert Image

Opens the **Insert Image** dialog box that allows you to define an image *(on page 1032)* 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.

Figure 369. Insert Link Dialog Box



Insert Ordered List

Inserts an ordered list *(on page 1035)* 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 1035)* 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 1037)* 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 1037)* at the cursor position.

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 391)*. 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 `ŉ`
- `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 394)*.

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 124)* 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 236)*.

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 124)* 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 236)*.

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 124)* 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 236)*.

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 124)* 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 236)*.

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 124)* 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 236)*.

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 124)* 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 236)*.

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 + Option + 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 397)*.

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 402)*.

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 1017)*
Markdown Editor Syntax Rules and Specifications *(on page 1027)*

Syntax Highlighting in the Markdown Editor

Oxygen XML Developer 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.

Note:
For Markdown documents that contain code from other languages, those blocks are rendered with proper syntax highlights to make them more distinguishable. The languages that are rendered with syntax highlights within Markdown documents include JavaScript, JSON, YAML, XML, XSLT, XSD, HTML, RNG, RNC, Java, Python, and CSS.

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 79)*.
2. Go to **Editor > Syntax Highlight** *(on page 158)*.
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 158)*
Markdown Editor *(on page 1017)*

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 474)* 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 204)*. 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 564)* for one of the following URIs:
    - **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** - 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 204)* or the path to the Schematron file is empty.

  Warning:
  If you are using a custom version of DITA-OT *(on page 200)*, 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 1017)*

Markdown Editor Syntax Rules and Specifications

The Markdown editor in Oxygen XML Developer 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 Developer 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

```
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

```
# 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

```


* * *


*****


---------------------------------------


_ _ _ _
```

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.

  ```
  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;&#x61;i&#x6C;&#x74;&#x6F;:&#x61;&#x64;&#x64;&#x72;&#x65;

&#115;&#115;&#64;&#101;&#120;&#x61;&#109;&#x70;&#x6C;e&#x2E;&#99;&#111;

&#109;">&#x61;&#x64;&#x64;&#x72;&#x65;&#115;&#115;&#64;&#101;&#120;&#x61;

&#109;&#x70;&#x6C;e&#x2E;&#99;&#111;&#109;</a>
```

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:

  ```
  Text with ![Alt text](/path/to/img.jpg "Optional title") inline image and a title.
  ```

  Without a title:

  ```
  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.

  ```
  Text with ![Alt text1][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.

  ```
  [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,
  > 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. Suspendisse
> id sem consectetuer 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

```
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:

```
<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 Developer 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: How to Add Syntax Highlights for Codeblocks in the Output *(on page 1341)*.

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

```
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 1032)* and *code blocks (on page 1033)* 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.

  ```
  * 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:

```
<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.

  ```
  | First Header  | Second Header |
  | ------------- | ------------- |
  | Column 1 Row 1 Cell  | Column 2 Row 1 Cell  |
  | Column 1 Row 2 Cell  | Column 2 Row 2 Cell  |
  ```

- **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).

  ```
  | First Header  | Second Header |
  | --- | --- |
  | `inline code`  | Content with **bold text** inside cell  |
  ```

- **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.

  ```
  | Left-aligned | Center-aligned | Right-aligned |
  | :---         |     :---:      |          ---: |
  | Content Cell | Content Cell   | Content Cell  |
  ```

- **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).

  ```
  | First Header | Second Header | Third Header | Fourth Header |
  | ------------ | ------------- | ------------ | ------------- |
  | Content Cell | *Cell Span Over 3 Columns* |||
  ```

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:

```
\   backslash
`   backtick
*   asterisk
_   underscore
{}  curly braces
[]  square brackets
()  parentheses
#   hash mark
```

```
+   plus sign

-   minus sign (hyphen)

.   dot

!   exclamation mark
```

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.

Literal Character	Escaping Code
<	<
>	>
&	&

Footnotes

The Markdown editor in Oxygen XML Developer 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.]
```

Latex Mathematical Equations

Mathematical equations in Latex format can be specified either inline or as blocks.

Example of an inline equation:

```
$f(x) = x^2$
```

Such equations are automatically preserved as `<foreign>` elements when converting the Markdown content to DITA XML and with the addition of editing and publishing plugins, they can be rendered in the Markdown editor's **DITA** preview tab and when the Markdown content is published as part of a DITA map.

Related information

Default Markdown Syntax

GitHub Flavored Markdown Rules

Markdown Editor *(on page 1017)*

Actions Available in the Markdown Editor *(on page 1019)*

Other Supported Document Types

Along with the fully supported built-in *frameworks* (document types) *(on page 1042)*, Oxygen XML Developer 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 Developer include:

- EPUB (NCX, OCF, OPF 2.0, 3.0, & 3.1) *(on page 1064)* - A standard for e-book files.
- OOXML *(on page 1738)* - An XML-based file format for representing spreadsheets, charts, presentations, and word processing documents.
- ODF *(on page 1738)* - 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 1053)* - For resolving cross-references when using *olinks*.
- 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.
- 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 2309)* document that describes a mapping between external entity references and locally-cached equivalents.
- Other Non-XML Files *(on page 329)*- Oxygen XML Developer also includes a simple text editor and a variety of helpful features for creating and editing non-XML files.

External Document Types

Some document types are available to be manually installed from GitHub:

- StratML (Part 1 & 2) - Part 1 and 2 of the Strategy Markup Language specification. Available to be installed from GitHub.
- EAD - Encoded Archival Description is an XML standard for encoding archival finding aids. Available to be installed from GitHub.
- KML - Keyhole Markup Language is an XML notation for expressing geographic visualization in maps and browsers. Available to be installed from GitHub.

9.

Built-in Frameworks (Document Types)

Oxygen XML Developer 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 Developer includes fully supported built-in *frameworks (on page 2305)* for popular XML document types (e.g. DITA, DocBook, TEI, XHTML, JATS), as well as other document types (e.g. JSON, YAML, OpenAPI, Markdown, HTML, CSS), each with a specialized set of editing features for the particular document type *(on page 441)*.

The built-in *frameworks (on page 2305)* 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 **Text** mode.
- Built-in transformation scenarios used for publishing XML documents.
- *XML Catalogs (on page 2309)* used for mapping resources.
- New document templates to make it easy to create XML documents.

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, 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 299)* 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 XML Catalog

The default *XML Catalog (on page 2309)*, `catalog.xml`, is stored in *`[OXYGEN_INSTALL_DIR]`*`/frameworks/docbook/`.

Transformation Scenarios

Oxygen XML Developer includes numerous built-in DocBook transformation scenarios that allow you to transform DocBook 4 documents to a variety of outputs, such as 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 1210)*.

For more information, see the DocBook Transformation Scenarios *(on page 1099)* section.

Resources

-
- DocBook Specifications

Related Information:
Editing XML Documents in Text Mode *(on page 442)*

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:

```
<chapter id="user_accounts">
<title>Administering User Accounts</title>
<para>blah blah</para>
```

You can form a cross reference to that chapter by adding an `<olink>`, as in the following example:

```
You may need to update your
<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 Developer includes a built-in new document template called **DocBook Targetset Map** available in the **New** document wizard *(on page 299)* 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 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 name="mailadmin">
        <document targetdoc="MailAdminGuide">
          &agtargets;
        </document>
      </dir>
    </dir>
    <dir name="reference">
      <dir name="mailref">
        <document targetdoc="MailReference">
          &reftargets;
        </document>
      </dir>
    </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:

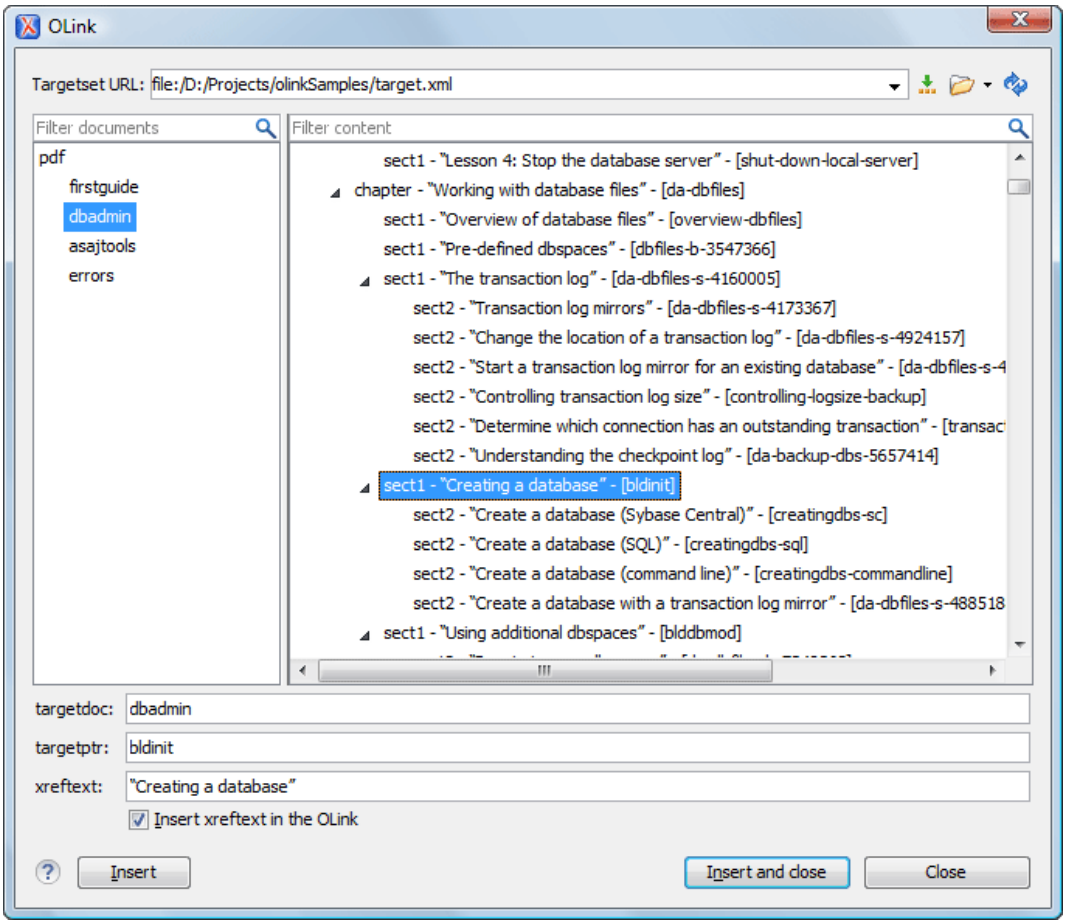
```
<div element="book" href="#MailAdminGuide" number="1" targetptr="user_accounts">
 <ttl>Administering User Accounts</ttl>
 <xreftext>How to administer user accounts</xreftext>
 <div element="part" href="#d5e4" number="I">
  <ttl>First Part</ttl>
  <xreftext>Part I, “First Part”</xreftext>
  <div element="chapter" href="#d5e6" number="1">
    <ttl>Chapter Title</ttl>
    <xreftext>Chapter 1, Chapter Title</xreftext>
    <div element="sect1" href="#src_chapter" number="1" targetptr="src_chapter">
      <ttl>Section1 Title</ttl>
      <xreftext>xreflabel_here</xreftext>
    </div>
  </div>
 </div>
</div>
```

5. Insert `<olink>` elements in the DocBook documents.

   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.

   Figure 370. Insert OLink Dialog Box



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, 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 299)* 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.

Transformation Scenarios

Oxygen XML Developer includes numerous built-in DocBook transformation scenarios that allow you to transform DocBook 5 documents to a variety of outputs, such as PDF, HTML, HTML Chunk, XHTML, XHTML Chunk, and EPUB. Oxygen XML Developer also includes a DocBook 5.1 transformation scenario for *Assembly* documents *(on page 1051)*. All of them are listed in the **DocBook 5** section in the **Configure Transformation Scenario(s)** dialog box *(on page 1210)*.

For more information, see the DocBook Transformation Scenarios *(on page 1099)* section.

Resources

- DocBook 5.0 (and older) Specifications
- DocBook 5.1 Specifications
- DocBook 5.1: The Definitive Guide

Related Information:

Editing XML Documents in Text Mode *(on page 442)*
DocBook Assembly (5.1 and Later) *(on page 1051)*
DocBook Topic (5.1 and Later) *(on page 1052)*

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:

```
<chapter id="user_accounts">
<title>Administering User Accounts</title>
<para>blah blah</para>
```

You can form a cross reference to that chapter by adding an `<olink>`, as in the following example:

```
You may need to update your
<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 Developer includes a built-in new document template called **DocBook Targetset Map** available in the **New** document wizard *(on page 299)* 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 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 name="mailadmin">
          <document targetdoc="MailAdminGuide">
            &agtargets;
          </document>
        </dir>
      </dir>
      <dir name="reference">
        <dir name="mailref">
          <document targetdoc="MailReference">
            &reftargets;
          </document>
```

```
        </dir>
      </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.

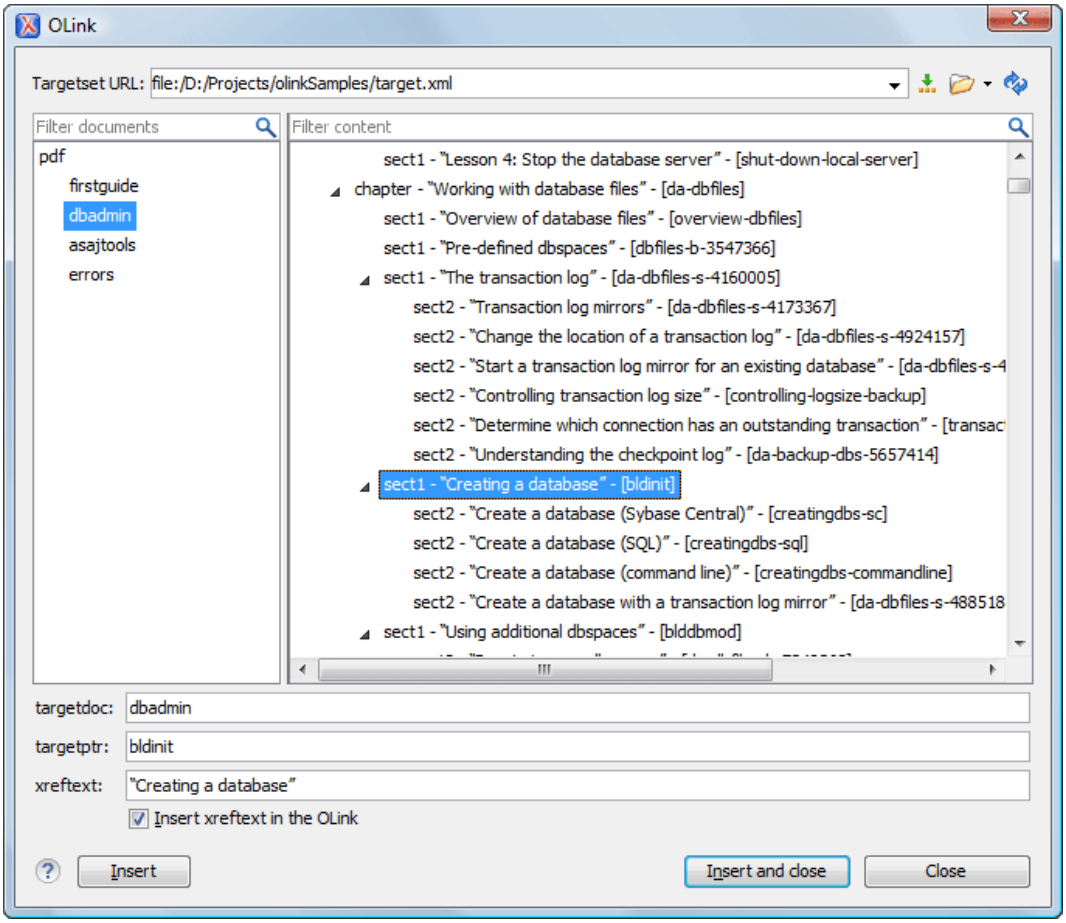
   Example: An example of a `target.db` file:

```
<div element="book" href="#MailAdminGuide" number="1" targetptr="user_accounts">
 <ttl>Administering User Accounts</ttl>
 <xreftext>How to administer user accounts</xreftext>
 <div element="part" href="#d5e4" number="I">
  <ttl>First Part</ttl>
  <xreftext>Part I, “First Part”</xreftext>
  <div element="chapter" href="#d5e6" number="1">
    <ttl>Chapter Title</ttl>
    <xreftext>Chapter 1, Chapter Title</xreftext>
    <div element="sect1" href="#src_chapter" number="1" targetptr="src_chapter">
      <ttl>Section1 Title</ttl>
      <xreftext>xreflabel_here</xreftext>
    </div>
  </div>
 </div>
</div>
```

5. Insert `<olink>` elements in the DocBook documents.
   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.

Figure 371. Insert OLink Dialog Box



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 Assembly (5.1 and Later)

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 1052)*) 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 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 299)* 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 Developer 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 1210)*.

Resources

- The Definitive Guide - DocBook Assemblies
- DocBook Specifications
- Sample files: *[OXYGEN_INSTALL_DIR]*`/samples/docbook/v5/assembly/`

DocBook Topic (5.1 and Later)

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 *Assembly (on page 1051)* documents.

For detailed information about the DocBook *Topic* document type, see The Definitive Guide - DocBook 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 299)* 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 1052)*. 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 1210)*.

Resources

- The Definitive Guide - DocBook Topic
- DocBook Specifications
- Sample files: `[OXYGEN_INSTALL_DIR]/samples/docbook/v5/assembly/`

Related Information:
DocBook Assembly (5.1 and Later) *(on page 1051)*

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 299)* 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*).

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 94)* is selected from the **Document Type Association** preferences page *(on page 93)*.

Default Document Templates

There are a variety of default *DITA topic* templates available when creating new documents from templates *(on page 299)* 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 XML Catalogs

The default *XML Catalogs (on page 2309)* 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 Developer 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 1210)*.

Resources

- DITA Specifications
- DITA Style Guide Best Practices for Authors
- Oxygen Video Tutorial: DITA Editing

Related Information:
Editing XML Documents in Text Mode *(on page 442)*

DITA Map Document Type (Framework)

DITA maps (on page 2304) 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.

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 94)* from the **Document Type Association** preferences page *(on page 93)* is selected.

Default Document Templates

There are a variety of default *DITA map* templates available when creating new documents from templates *(on page 299)* 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/`.

Default XML Catalogs

The default *XML Catalogs (on page 2309)* 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 Developer 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)** dialog box *(on page 1210)*.

For more information, see the DITA Map Transformation Scenarios *(on page)* section.

Resources

- DITA Specifications
- DITA Style Guide Best Practices for Authors
- Oxygen Video Tutorial: DITA Maps Manager

Related Information:
Editing XML Documents in Text Mode *(on page 442)*

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® has a modular architecture based on the following components:

- **Data Module** - An XML-based, uniquely identified unit of information that can be managed, reused, and exchanged independently within the S1000D environment.
- **Data Module Structure** - Describes the internal organization of a *Data Module*, typically divided into:
  - 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)** - The managed repository where S1000D objects (such as *Data Modules*, *Publication Modules*, and illustrations) are stored, controlled, and related to each other, and from which publications are assembled.

- **Business Rules Exchange (BREX)** - A specialized *Data Module* that defines project- or organization-specific business rules governing how S1000D is applied, and against which other *Data Modules* can be checked for conformance.
- **Publication Module (PM)** - An XML-based control file that defines the structure, sequence, and inclusion of *Data Modules* in a deliverable publication, effectively acting as the publication's table of contents.
- **Information Control Number (ICN)** - A unique identifier assigned to illustrations and other multimedia objects, so they can be managed and referenced as standalone entities within the *CSDB*.
- **Data Dispatch Note (DDN)** - A control document used to describe, package, and manage the exchange of *Data Modules* and related S1000D objects between organizations or systems.
- **Data Module Requirement List (DMRL)** - A structured list of required *Data Modules* for a project, used as a planning and tracking tool before and during authoring.
- **Comment** - A Data Module type used to record review comments, feedback, or annotations that reference one or more existing *Data Modules*.

Oxygen XML Developer includes built-in support for **S1000D**. You can work with S1000D issues 4.0, 4.1, and 4.2 using built-in support for editing, visual authoring, templates, BREX handling, and smart paste.

Default Document Templates

You can get assistance with creating new S1000D® Data Modules or Publication Modules for issues 4.0, 4.1, and 4.2 by using the provided file templates available when creating new documents from templates *(on page 299)*.

Default Schema for Validation and Content Completion

The default schemas used for validation and content completion are stored in `[OXYGEN_INSTALL_DIR]/frameworks/s1000d-base/` (with separate folders for each supported issue).

S1000D document editing features include schema-guiding, validation, and content completion support. Also, documentation for S1000D elements and attributes can be viewed directly in the content completion window and in the **Attributes** view.

BREX Editing and Transformation

BREX modules can be edited in the **Text** or **Author** mode with content completion and validation support. You can also convert BREX modules to Schematron using the provided transformation scenario.

If you want more advanced **S1000D** editing features, you can ask some of our partners.

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 299)* 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 XML Catalogs

The default *XML Catalogs (on page 2309)* 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 Developer 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 1210)*.

Related information

Editing HTML Documents *(on page 1010)*

Editing XML Documents in Text Mode *(on page 442)*

XHTML Specifications

XHTML Validation

XHTML documents can be validated in Oxygen XML Developer using the same validation features as with any other XML document. In addition, Oxygen XML Developer 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 (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 514)*

Batch Validation and Transformation *(on page 346)*

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 299)* 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 XML Catalogs

The default *XML Catalogs (on page 2309)* 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 Developer 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 1210)*.

Resources

-
- TEI: P5 Guidelines

Related Information:
Editing XML Documents in Text Mode *(on page 442)*

How to Install a TEI Framework with the Latest Schema and Stylesheets

The TEI framework that is bundled in Oxygen XML Developer has the TEI schema and stylesheets that were available at the time of its release. When the *TEI Consortium* releases a new TEI version (Schema and Stylesheets), they also release a new version of the Oxygen XML Developer TEI framework with them.

Warning:
TEI Consortium, who maintains these releases, chose to keep them compatible with Oxygen XML Developer version 18.1. This means that some features or improvements that were developed in later versions of Oxygen XML Developer might be missing from these frameworks.

The following procedure describes how to install a release of the TEI framework done by *TEI Consortium* that has a newer version of the TEI Schema and TEI XSL https://github.com/TEIC/oxygen-tei/blob/master/oxygen-tei-plugin.md

1. Go to **Help > Install new add-ons**.
2. Enter or paste **https://www.tei-c.org/release/oxygen/updateSite.oxygen** in the **Show add-ons from** field.

3. Choose the TEI framework that 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 **Install**.
5. Restart the application.

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 299)* 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 XML Catalogs

The default *XML Catalogs (on page 2309)* 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 Developer 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 1210)*.

Resources

-
- TEI: Getting Started with ODD

Related Information:
Editing XML Documents in Text Mode *(on page 442)*

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 299)* 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 XML Catalogs

The default *XML Catalogs (on page 2309)* 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 Developer 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 1210)*.

Resources

-
- *jTEI* Article Guidelines

Related Information:
Editing XML Documents in Text Mode *(on page 442)*

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//DTD`.

Default Document Templates

There are some default *JATS* templates available when creating new documents from templates *(on page 299)* and they can be found in: **Framework Templates > JATSKit - 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 XML Catalog

The default *XML Catalog (on page 2309)*, `jatskit-catalog.xml`, is stored in *[OXYGEN_INSTALL_DIR]*`/frameworks/jats/lib/schemas/`.

Transformation Scenarios

Oxygen XML Developer 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 1210)*.

Resources

-
- NLM Journal Archiving and Interchange Tag Suite

Related Information:
Editing XML Documents in Text Mode *(on page 442)*

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 Developer includes an **Archive Browser** view *(on page 1735)* that allows you to view the contents and structure of this type of file.

Three distinct *frameworks (on page 2305)* 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 Developer 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 299)* 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 1738)*

OpenAPI (Swagger) Document Type (Framework)

OpenAPI specification, previously known as Swagger specification, is a 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 API's services and are represented in either YAML or JSON format.

Oxygen XML Developer includes three OpenAPI frameworks:

- OpenAPI 2.0
- OpenAPI 3.0
- OpenAPI 3.1

Editing OpenAPI Documents

You can edit OpenAPI files in **Text** mode and you have access to all the usual text editing actions.

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 Developer.

Validation and Content Completion

Validation and content completion is supported in Oxygen XML Developer for OpenAPI documents (version 2.0, 3.0, 3.1). The validation and content completion in OpenAPI documents are driven by schemas according to the OpenAPI version in the document. Each of the three frameworks (OpenAPI 2.0, OpenAPI 3.0, and

OpenAPI 3.1) have a unique schema specified for content completion and validation. When opening an OpenAPI document (in JSON or YAML format), Oxygen XML Developer automatically associates the corresponding schema based on the OpenAPI version of the document.

Resources

- Oxygen XML Developer includes a tool for generating documentation for OpenAPI components in HTML format: Generating OpenAPI Documentation *(on page 1998)*.
- Oxygen XML Developer includes a testing tool for *OpenAPI* files: OpenAPI Tester *(on page 1983)*.
- For more details about the *OpenAPI Specification*, along with example documents, go to https://spec.openapis.org/oas/latest.html.
- Video: **OpenAPI Document Editing in Oxygen XML Editor**
- Webinar: **OpenAPI Editing, Testing, and Documenting**
- Webinar: **OpenAPI/AsyncAPI Support in Oxygen**

OpenAPI Test Scenario Document Type (Framework)

Oxygen XML Developer includes a specialized framework for working with OpenAPI test scenario files. It includes the usual text editing actions, a visual editing interface, content completion, and automatic validation.

Default Document Template

There is a default scenario file template available when creating new documents from templates *(on page 299)* and they can be found in the **Framework Templates > OpenAPI Test Scenario**.

Content Completion

Oxygen XML Developer helps you edit OpenAPI test scenario files through the *Content Completion Assistant (on page 2303)*, offering proposals for properties and values that can be inserted at the cursor position. It can be manually activated with the **Ctrl + Space** shortcut.

Validation

Oxygen XML Developer includes built-in validation for OpenAPI test scenario documents to help you keep them well-formed. The documents are validated automatically as you type against the schema specified in the framework and problems are highlighted within the document.

OpenAPI Tools

The following additional OpenAPI tools are available as free add-ons:

- **OpenAPI Tester** - 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. For details, see OpenAPI Tester Add-on.
- **Run OpenAPI Test Scenario** - Provides the ability to run a test suite for an OpenAPI document in JSON format. It performs the requests based on the specified OpenAPI document and the data entered in the

test file, and then checks if the server responses are as expected. For details, see Run OpenAPI Test Scenario *(on page 1983)*.

- **OpenAPI Documentation Generator** - Provides the ability to generate 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. For details, see OpenAPI Documentation Generator Add-on *(on page 1998)*.

Resources

- For more details about the *OpenAPI Specification*, along with example documents, go to https://spec.openapis.org/oas/latest.html.
- Video: **OpenAPI Document Editing in Oxygen XML Editor**
- Webinar: **OpenAPI Editing, Testing, and Documenting**
- Webinar: **OpenAPI/AsyncAPI Support in Oxygen**

AsyncAPI Document Type (Framework)

Oxygen XML Developer includes a specialized framework for working with AsyncAPI files. The application supports the following AsyncAPI versions: `1.0.0`, `1.1.0`, `1.2.0`, `2.0.0`, `2.1.0`, `2.2.0`, `2.3.0`, `2.4.0`, `3.0.0`.

Editing AsyncAPI Documents

You can edit AsyncAPI files in **Text** mode and you have access to all the usual text editing actions.

Default Document Templates

There are some default AsyncAPI templates available when creating new documents from templates *(on page 299)* and they can be found in the **Framework Templates > AsyncAPI 1.x** and **Framework Templates > AsyncAPI 2.x** folders. Each of those folders contain a default new document template for a JSON version and a YAML version. Some other useful examples can be found at public AsyncAPI GitHub project.

Content Completion

Oxygen XML Developer helps you edit AsyncAPI files through the *Content Completion Assistant (on page 2303)*, offering proposals for properties and values that can be inserted at the cursor position. It can be manually activated with the **Ctrl + Space** shortcut.

Validation

Oxygen XML Developer includes built-in validation for AsyncAPI documents to help you keep them well-formed. The documents are validated automatically as you type against the schema specified in the framework and problems are highlighted within the document.

Resources

- For details about the *AsyncAPI Specification*, go to https://www.asyncapi.com/docs.
- Webinar: **OpenAPI/AsyncAPI Support in Oxygen**

JSON-LD Document Type (Framework)

Oxygen XML Developer includes a specialized framework for working with JSON-LD files. *JSON-LD* (JavaScript Object Notation for Linked Data) consists of multi-dimensional arrays and is considered an easy-to-use lightweight *Linked Data* format.

Editing JSON-LD Documents

You can edit JSON-LD files in the specialized JSON text mode editor *(on page 851)* and you have access to its various features and actions.

Default Document Template

There are some default *JSON-LD* templates available when creating new documents from templates *(on page 299)* and they can be found in: **Framework Templates > JSON-LD**. Some other useful examples can be found at public JSON-LD GitHub project.

Content Completion

Oxygen XML Developer includes an intelligent *Content Completion Assistant (on page 2303)* that offers proposals for inserting JSON structures that are valid at the current editing location. For more details, see Content Completion Assistant in JSON *(on page 863)*.

Validation

Oxygen XML Developer includes built-in validation for JSON-LD documents to help you keep them well-formed. The documents are validated automatically as you type against the schema specified in the framework and problems are highlighted within the document. For more details, see Validating JSON Documents *(on page 855)*.

10.

Additional XML Editing Frameworks (Document Types)

Oxygen XML Developer supports custom frameworks (document types) contributed by the XML community. They provide support for additional functionality and XML vocabularies.

Similar to the built-in *frameworks (on page 2305)*, the additional frameworks may define:

- A default grammar used for validation and content completion in **Text** mode.
- Built-in transformation scenarios used for publishing XML documents.
- *XML Catalogs (on page 2309)* used for mapping resources.
- New document templates to make it easy to create XML documents.

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 Developer 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 Developer, 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.

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 Developer, 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 Developer includes preconfigured built-in transformation scenarios *(on page 1071)*, but you can also create new transformation scenarios *(on page 1101)*.

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 1102)* and XML Transformation with XQuery *(on page 1117)*.
- **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 XML *(on page 1159)*.
- **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 XML *(on page 1190)*.
- **Scenarios that Apply to JSON Files** - This type of scenario contains the location of an XSLT stylesheet that is applied on the edited JSON document, as well as other transformation parameters. For more information, see JSON Transformation with XSLT *(on page 1144)* and XSLT Transformation on JSON *(on page 1178)*.
- **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 1206)*.
- **DITA-OT Scenarios** - This type of scenario provides the parameters for an Ant transformation that executes a DITA-OT build script. Oxygen XML Developer 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 1127)*.
- **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 1140)*.



Note:

Status messages generated during the transformation process are displayed in the **Information** view *(on page 437)*.

Built-in Transformation Scenarios

Oxygen XML Developer 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 1210)* or **Transformation Scenarios** view *(on page 1217)*. All the built-in *document types (frameworks) (on page 2305)* that are included in Oxygen XML Developer 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 334)*:

- **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 1210)* and begin the transformation process. If an association is not detected, this action will open the **Configure Transformation Scenario(s)** dialog box *(on page 1210)* 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 1210)* 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 Developer 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 1101)*
Editing a Transformation Scenario *(on page 1207)*
Configure Transformation Scenario(s) Dialog Box *(on page 1210)*
Applying Associated Transformation Scenarios *(on page 1210)*
Transformation Scenarios View *(on page 1217)*

DITA Map Transformation Scenarios

Built-in transformation scenarios allow you to transform *DITA maps (on page 2304)* to a variety of outputs, such as WebHelp, PDF, ODF, XHTML, EPUB, CHM, Kindle, and MS Word. Oxygen XML Developer also includes a special **Integrate/Install DITA-OT Plugins** *(on page 1096)* 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 1210)*.

A variety of transformations scenarios are available for *DITA maps (on page 2304)*:

- Built-in transformation scenarios allow you to transform a *DITA map* to a variety of outputs, such as WebHelp, PDF, ODF, XHTML, EPUB, CHM, Metrics Report, and MS Word.
- **Integrate/Install DITA-OT Plugins** *(on page 1096)* - 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.

Related Information:

Editing a Transformation Scenario *(on page 1207)*

Configure Transformation Scenario(s) Dialog Box *(on page 1210)*

Applying Associated Transformation Scenarios *(on page 1210)*

DITA Topic Transformation Scenarios *(on page)*

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 1222)* are designed to adapt to any device and screen size to provide an optimal viewing and interaction experience. Oxygen XML Developer also provides numerous possibilities for customizing the WebHelp Responsive output *(on page 1308)*.

WebHelp Responsive Transformation Scenario

To publish a *DITA map (on page 2304)* as **WebHelp Responsive** output, follow these steps:

1. Select the **Configure Transformation Scenario(s)** action from the 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)* - This tab contains a set of built-in publishing templates *(on page 1267)* 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 1308)*.
   - Parameters Tab *(on page)* - This tab includes numerous parameters that can be set to customize your WebHelp system output. See the Parameters section *(on page 1074)* below for details about the most commonly used parameters for WebHelp Responsive transformations.
   - Feedback Tab *(on page)* - 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)* - This tab allows you to filter certain content elements from the generated output.

- Advanced Tab *(on page)* - This tab allows you to specify some advanced options for the transformation scenario.
- Output Tab *(on page)* - 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 Developer)

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.csh.disable.topicID.fallback

Specifies whether or not topic ID *fallbacks* are enabled when computing the mapping of context sensitive help and `resourceid` information is not available. Possible values are **false** (default) and **true**.

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.enable.scroll.to.search.term

Specifies whether or not the page should scroll to the first search term when opening the search results page. Possible values are **no** (default) and **true**.

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 Developer (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 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.labels.generation.mode

Controls whether or not labels are generated in the output. These labels are useful because users can easily search for topics with the same label by simply clicking on the label presented in the output. Possible values are:

- **keywords-label** - Generates labels for each defined `<keyword>` element that has the `@outputclass` attribute value set to **label**.
- **keywords** - Generates labels for each defined `<keyword>` element. If the topic contains `<keyword>` elements with the `@outputclass` attribute value set to **label**, then only these elements will have labels generated for them in the output.
- **disable** - Disables the generation of labels in the Webhelp Responsive output.

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.search.default.operator

Makes it possible to change the default operator for the search engine. Possible values are `and`, `or` (default). If set to `and` while the search query is WORD1 WORD2, the search engine only returns results for topics that contain both WORD1 and WORD2. If set to `or` and the search query is WORD1 WORD2, the search engine returns results for topics that contain either WORD1 or WORD2.

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 terms page will be displayed in the output. The default value is `yes` (meaning the index terms icon is displayed). If set to `false`, the index terms icon is not displayed in the output and the index terms page is not generated.

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.expand.collapse.sections

Specifies if links to collapse sections will be presented within each topic in the output. Collapsing sections will collapse all collapsible elements (nested topics with titles, sections with titles, or tables with titles). 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.skip.main.page.generation

If set to `true`, the default main page is not generated in the output. The default value is `false`.

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 (nested topics with titles, sections with titles, tables with titles or 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 1308)*
Layout and Features *(on page 1222)*

DITA Map PDF - based on HTML5 & CSS Transformation

Oxygen XML Developer 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 Developer comes bundled with a built-in CSS-based PDF processing engine called **Oxygen PDF Chemistry**. Oxygen XML Developer 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 Developer. 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 Developer 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 Developer installation kit) - A third-party component that needs to be purchased from http://www.antennahouse.com/antenna1/formatter/.

How to Create the Transformation Scenario

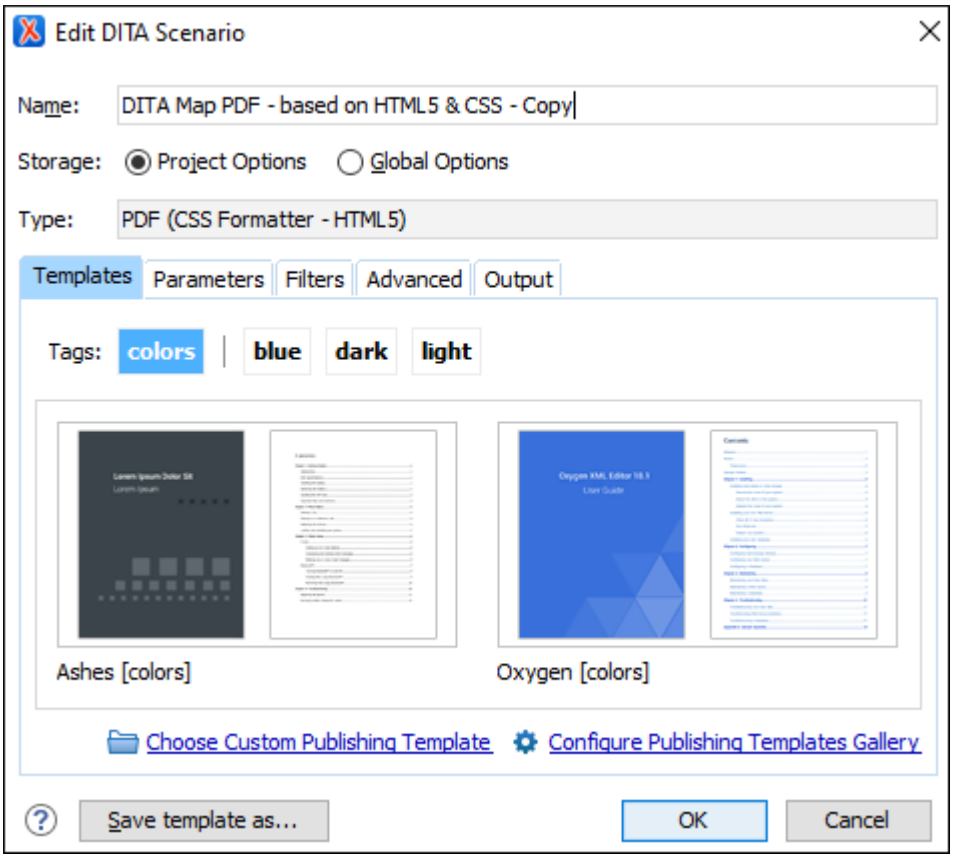
To create a **DITA Map PDF - based on HTML5 & CSS** transformation scenario, follow these steps:

1. Click the  **Configure Transformation Scenario(s)** button.
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)* - 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 372. DITA Map to PDF Templates



- **Parameters** Tab *(on page)* - This tab includes numerous parameters that can be set to customize the transformation.
- **Filters** Tab *(on page)* - This tab allows you to filter certain content elements from the generated output.
- **Advanced** Tab *(on page)* - This tab allows you to specify some advanced options for the transformation scenario.
- **Output** Tab *(on page)* - 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
   - **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 196)*.

- **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 1438)*.

Related Information:

Editing a Transformation Scenario *(on page 1207)*
Configure Transformation Scenario(s) Dialog Box *(on page 1210)*
Oxygen PDF Chemistry User Guide
CSS-based DITA to PDF Customization *(on page 1438)*

DITA Map PDF - based on XSL-FO Transformation

Oxygen XML Developer comes bundled with the DITA Open Toolkit that provides a mechanism for converting *DITA maps (on page 2304)* 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.
2. Select **DITA Map PDF - based on XSL-FO** and click the **Edit** button (or use the **Duplicate** button if your *framework (on page 2305)* 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 1713)*

DITA Map MS Office Word Transformation

Oxygen XML Developer comes bundled with a transformation scenario that allows you to convert *DITA maps (on page 2304)* 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-OT/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. The Word application used to edit the **Normal.docx** should be configured with English locale as the style names for each Word element must be in English.

Configuring the Transformation Scenario

To configure a **DITA Map to MS Office Word** transformation scenario, follow these steps:

1. Open the DITA map.
2. Click the **Configure Transformation Scenario(s)** button.
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 1207)*
Configure Transformation Scenario(s) Dialog Box *(on page 1210)*

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 Developer needs `Microsoft HTML Help Workshop` to be installed on your computer. Oxygen XML Developer 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 Developer 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 Developer 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.

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-OT/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 Metrics Report Transformation

A **DITA Map Metrics Report** action is available on the **DITA Maps Manager** toolbar and in the **DITA Maps** main menu. It generates an overview report that contains useful statistics for a DITA map.

As an alternate approach, to create a metrics report from a *DITA map (on page 2304)* using a transformation scenario, follow these steps:

1. Select the **Configure Transformation Scenario(s)** action from the toolbar.
2. Select the **DITA Map Metrics Report** scenario from the **DITA Map** section.
3. Run the transformation.

The generated HTML report contains information such as:

- The number of processed maps and topics.
- The number of `map/bookmap/topic/task/concept/reference` types in the DITA map.
- Content reuse percentage.
- Number of elements and attributes of different types used in the entire *DITA map* structure.
- Number of 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 is not 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.

The metrics report can also be obtained in XML format, making it possible to construct a metrics report evolution between multiple versions of the same DITA project.

DITA Map Zendesk Publishing

Oxygen XML Developer 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.

Attention:
This feature is only available in the **Enterprise** edition of Oxygen XML Developer.

To run the transformation, follow these steps:

1. Start Oxygen XML Developer and open a *DITA map*.
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

An API token, generated in the *Zendesk* admin pages, necessary for authentication to the server: https://support.zendesk.com/hc/en-us/articles/226022787-Generating-a-new-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-us/articles/218222877-Organizing-knowledge-base-content-in-categories-and-sections#topic_hjs_tl4_kk.

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_ysj_wtt_zz.



Note:
When publishing to a subsection, a path of section names separated by '///' can be passed (for example: `Main section///Subsection 1///Subsection 1.1`).

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`.

Visible to

The name of the user segment that can view the articles.

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.

Create table of contents

When the `zendesk.create.toc` parameter is set to `true` (default value), a table of contents is published as the first article in the Zendesk section and this initial article references all other articles. The title of the TOC article is the title of the DITA map.

Specify a prefix title for the table of contents

Set the `zendesk.toc.title.prefix` parameter to specify a prefix to use for the title of the table of contents article.

Specify a suffix title for the table of contents

Set the `zendesk.toc.title.suffix` parameter to specify a suffix to use for the title of the table of contents article.

5. Save the changes and run the transformation.

Important:

There may be cases when the publishing breaks, presenting an error related to **HTTPS** certificates, similar to this one:

```
Error: org.zendesk.client.v2.ZendeskException: java.net.ConnectException: PKIX path building failed:
sun.security.provider.certpath.SunCertPathBuilderException: unable to find valid certification path
to requested target
```

This usually occurs if an HTTPS proxy server is installed in your company's network. In this case, if running on Windows, you can edit the transformation scenario you are using to publish **DITA** to **Zendesk** and in the **Advanced** tab, go to the **JVM Arguments** field and set this value:

```
-Djavax.net.ssl.trustStoreType=Windows-ROOT -Djavax.net.ssl.trustStore=C:\\Windows\\win.ini
```

Resources

For more information about publishing content to the Zendesk Help Center, watch the following video demonstration:

https://www.youtube.com/embed/QZ_9Fk_LOk8

Integrate/Install DITA-OT Plugins Transformation

Oxygen XML Developer comes bundled with a transformation scenario designed to integrate DITA-OT plugins *(on page 2307)*. These DITA-OT *plugins* are used for various customizations. It is called **Integrate/Install DITA-OT Plugins** and is found in the **DITA Map** section of the **Configure Transformation Scenario(s)** dialog box *(on page 1210)*.

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 200)*.

CAUTION:
Oxygen XML Developer support engineers do not officially offer support and troubleshooting assistance for custom DITA-OT distributions or custom installed DITA-OT plugins. If you discover any issues or inconsistent behavior while using a custom DITA-OT or a DITA-OT that contains custom DITA-OT plugins, you should revert to the default built-in DITA-OT.

Running the Transformation Scenario

To integrate a DITA-OT *plugin*, follow these steps:

1. If Oxygen XML Developer 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 1210)* 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 1217)*).
3. Select the **Integrate/Install DITA-OT Plugins** 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 1210)*.
5. Check the **Results** panel at the bottom of the application to make sure the build was successful.
6. Restart Oxygen XML Developer with your normal permissions.

Related Information:
Configure Transformation Scenario(s) Dialog Box *(on page 1210)*

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 your DITA documents by using the **Validate** action from the **Validation** toolbar drop-down menu, the **Document > Validate** menu, or from the **Validate** menu when invoking the contextual menu in the **Project** view *(on page 334)*.
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 1127)*.
4. If the transformation results in errors or warnings, they are displayed in the **Results** panel *(on page 474)* 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**.

DITA Topic Transformation Scenarios

Oxygen XML Developer 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 1210)*.

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 Developer comes bundled with a built-in CSS-based PDF processing engine called **Oxygen PDF Chemistry**. Oxygen XML Developer 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 1469)* or publishing template *(on page 1457)* that you use for converting entire DITA maps.
- **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 1207)*
Configure Transformation Scenario(s) Dialog Box *(on page 1210)*
Applying Associated Transformation Scenarios *(on page 1210)*
DITA Map Transformation Scenarios *(on page)*

DocBook Transformation Scenarios

Built-in transformation scenarios allow you to transform DocBook documents to a variety of outputs, such as PDF, HTML, HTML Chunk, XHTML, XHTML Chunk, and EPUB. Oxygen XML Developer 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 1210)*.

Related Information:
Editing a Transformation Scenario *(on page 1207)*
Configure Transformation Scenario(s) Dialog Box *(on page 1210)*
Applying Associated Transformation Scenarios *(on page 1210)*

DocBook to DITA Transformation

Oxygen XML Developer 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 1207)*
Configure Transformation Scenario(s) Dialog Box *(on page 1210)*

DocBook to PDF Transformation

Oxygen XML Developer 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 1724)*.

Related Information:
Editing a Transformation Scenario *(on page 1207)*
Configure Transformation Scenario(s) Dialog Box *(on page 1210)*
DocBook to PDF Output Customization *(on page 1724)*

DocBook to EPUB Transformation

Oxygen XML Developer 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:

```
@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:

```
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 1210)*, 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 Developer 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 1207)* or **Duplicate** *(on page 1209)* 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 Developer 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 1217)*. 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 2308)* - 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 2305)* - 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 1101)* or edit an existing one *(on page 1207)*, 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 252)* 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 183)*, 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 1103)* 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 1116)*. 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 252)* 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 Developer for performing a transformation. These include the built-in engines and the external engines defined in the Custom Engines preferences page *(on page 191)*. 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 1106)* for the current transformation scenario. To configure the same options globally, go to the Saxon-HE/PE/EE preferences page *(on page 181)*. 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 1104)* 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 1104)*. 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 1105)* 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 1105)* 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 1105)*, 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 252)* (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 252)* 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 252)* 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 252)*

XSLT Extensions

The **Extensions** button opens a dialog box that allows you to specify the *JARS (on page 2306)* 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 252)* 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 181)* 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 2305)* defined in the preferences page.

Saxon-HE/PE/EE Options

The advanced options for Saxon 12.9 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 1106)*, 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 1106)*, the value in this option takes precedence.



Tip:
If your stylesheet includes `<xsl:template name="xsl:initial-template">`, Oxygen XML Developer 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 12.9 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.

> **Attention:**
> Oxygen XML Developer does not support the `ALLOWED_PROTOCOLS` Saxon property (*http://saxon.sf.net/feature/allowedProtocols*). This feature is specific to Saxon and controls access by Saxon (but not by underlying software, such as the XML parser). For more information, see https://www.saxonica.com/documentation12/index.html#!configuration/config-features.

Debugger trace into XPath expressions (applies to debugging sessions)

Instructs the XSLT Debugger *(on page 1845)* 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 12.9 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 12.9 processors.



Note:
Saxon-CE, being JavaScript-based, was designed to run inside a web browser. This means that you will use Oxygen XML Developer 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 12.9 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 79)* and go to **XML > XML Parser > XML Schema** and use the **Default XML Schema version** option *(on page 172)*.

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. However, in certain instances, the Saxon XSLT processor may auto detect and use streaming even if this option is not 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.

Using Saxon Integrated Extension Functions

Saxon, the transformation and validation engine used by Oxygen XML Developer, 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:

```
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 Developer 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 99)*.
- In a validation scenario *(on page 526)*, 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 1104)* 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 1857)*.

FO Processor Tab (XSLT Transformations)

When you create a new transformation scenario *(on page 1101)* or edit an existing one *(on page 1207)*, 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 192)*.

Output Tab (XSLT Transformations)

When you create a new transformation scenario *(on page 1101)* or edit an existing one *(on page 1207)*, 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 252)* button, or the **Browse** button.

Open in Browser/System Application

If selected, Oxygen XML Developer 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 252)* 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 Developer displays the transformation result in an XML viewer panel at the bottom of the application window with syntax highlighting *(on page 158)*.
- **SVG** - If this is selected, Oxygen XML Developer displays the transformation result in an integrated SVG viewer in the **Results** panel *(on page 1009)* 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 Developer 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 252)* 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 Developer.

Supported XSLT Processors

Oxygen XML Developer includes the following XSLT processors:

- **Xalan 2.7.2** (Deprecated) - 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 12.9 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 12.9 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 Developer. 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 172)*.

  > **Note:**
  > Oxygen XML Developer 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 Developer 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:
  >
  > 1. Download Saxon 12.9 Professional Edition (PE) or Enterprise Edition (EE) from http://www.saxonica.com.
  > 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 Developer.

- **Saxon-CE (Client Edition)** is Saxonica's implementation of XSLT 2.0 for use on web browsers. Oxygen XML Developer provides support for editing stylesheets that contain Saxon-CE extension functions and instructions. This support improves the validation, content completion, and syntax highlighting.

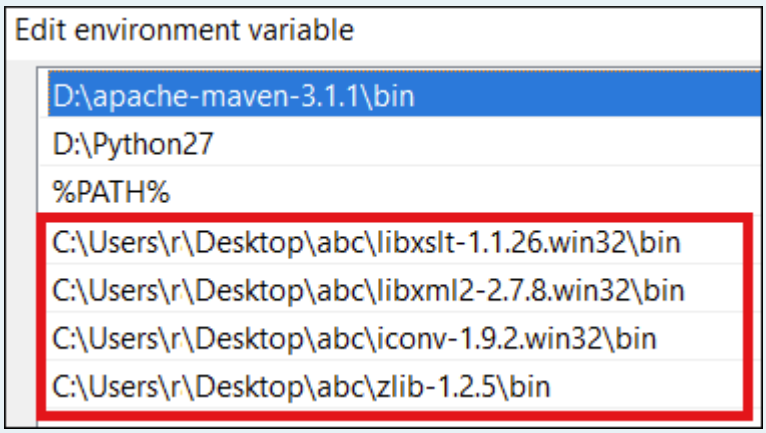
Note:
Saxon-CE, being JavaScript-based, was designed to run inside a web browser. This means that you will use Oxygen XML Developer only for developing the Saxon-CE stylesheet, leaving the execution part to a web browser. See more details about executing such a stylesheet on Saxonica's website.

Note:
A specific template, named **Saxon-CE stylesheet**, is available in the **New** document wizard *(on page 299)*.

- **Xsltproc (libxslt)** (Deprecated) - 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 Developer 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:
  1. Go to http://ftp.zlatkovic.com/libxml.en.html and download the following ZIP files: **iconv-1.9.2.win32.zip**, **libxml2-2.7.8.win32.zip**, **libxslt-1.1.26.win32.zip**, **zlib-1.2.5.win32.zip**.
  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:

  4. Restart Oxygen XML Developer.

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 184)*.

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 2309)* files of the built-in document types (DocBook, TEI, DITA, etc.) are not handled properly by LIBXML if Oxygen XML Developer 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.

For information about configuring the XSLT preferences, see the XSLT options *(on page 180)* section.

Configuring Custom XSLT Processors

Oxygen XML Developer 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 79)* 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 191)*.
4. Click **OK**.

Note:

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 Developer linked message *(on page 523)*, 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 191)*

Configuring the XSLT Processor Extensions Paths

The Saxon and Xalan 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 (Deprecated)** - http://xml.apache.org/xalan-j/extensions.html
- **Saxon 6.5.5** - http://saxon.sourceforge.net/saxon6.5.5/extensions.html
- **Saxon 12.9** - 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 1104)* 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 1217)*. 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 2308)* - 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 2305)* - 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 1101)* or edit an existing one *(on page 1207)*, 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 252)* 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 183)*, 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 252)* 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 Developer for performing a transformation. These include the built-in engines and the external engines defined in the Custom Engines preferences page *(on page 191)*. 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 1120)* for the current transformation scenario. To configure the same options globally, go to the Saxon-HE/PE/EE preferences page *(on page 181)*. 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 1119)* 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 can not 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 1120)* 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 252)* (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 252)* 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 252)* 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 252)*

XQuery Extensions

The **Extensions** button is used to specify the *JAR (on page 2306)* 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 186)* 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 2305)* defined in the preferences page.

Saxon-HE/PE/EE Options

The advanced options for Saxon 12.9 Home Edition (HE), Professional Edition (PE), and Enterprise Edition (EE) are as follows:

Use a configuration file ("-config")

Sets a Saxon 12.9 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 1123)* is not set.

Saxon-PE/EE Options

The following advanced options are specific for Saxon 12.9 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 12.9 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 1101)* or edit an existing one *(on page 1207)*, 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 192)*.

Output Tab (XQuery Transformations)

When you create a new transformation scenario *(on page 1101)* or edit an existing one *(on page 1207)*, 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 252)* button, or the **Browse** button.

Open in Browser/System Application

If selected, Oxygen XML Developer 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 252)* 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 Developer displays the transformation result in an XML viewer panel at the bottom of the application window with syntax highlighting *(on page 158)*.
- **SVG** - If this is selected, Oxygen XML Developer displays the transformation result in an integrated SVG viewer in the **Results** panel *(on page 1009)* 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 Developer 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 252)* 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 1217)*. 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 2308)* - 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 2305)* - 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 1101)* or edit an existing one *(on page 1207)*, 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** *(on page 252)* 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** *(on page 252)* 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 *(on page 101)*) are applied to the transformation in addition to any CSS referenced directly in the document or specified in the CSS URL field *(on page 1126)*.

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 *(on page 101)*).
- CSS files referenced directly in the document.
- CSS files specified in the CSS URL field *(on page 1126)*.

Processor options link

Opens the **CSS-based Processors** preferences page *(on page 196)* 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 *(on page 1101)* or edit an existing one *(on page 1207)*, 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 252)* button, or the **Browse** button.

Open in Browser/System Application

If selected, Oxygen XML Developer 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 196)* where you can configure some options for generating PDF output.

DITA-OT Transformation

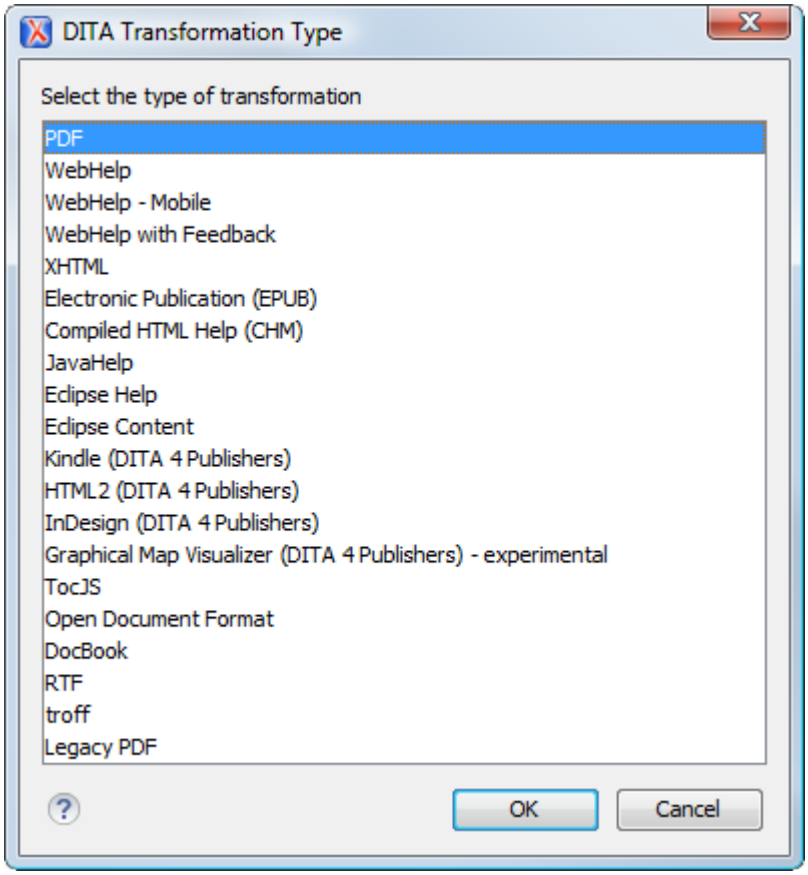
This type of transformation specifies the parameters for an Ant transformation that executes a DITA-OT build script. Oxygen XML Developer includes a built-in version of Ant and a built-in version of DITA-OT, but other versions can be set in the scenario.

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 1217)*. 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 373. 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 2308)* - 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 2305)* - The scenario is saved in the global options that are stored in the user home directory and is not accessible to other users.

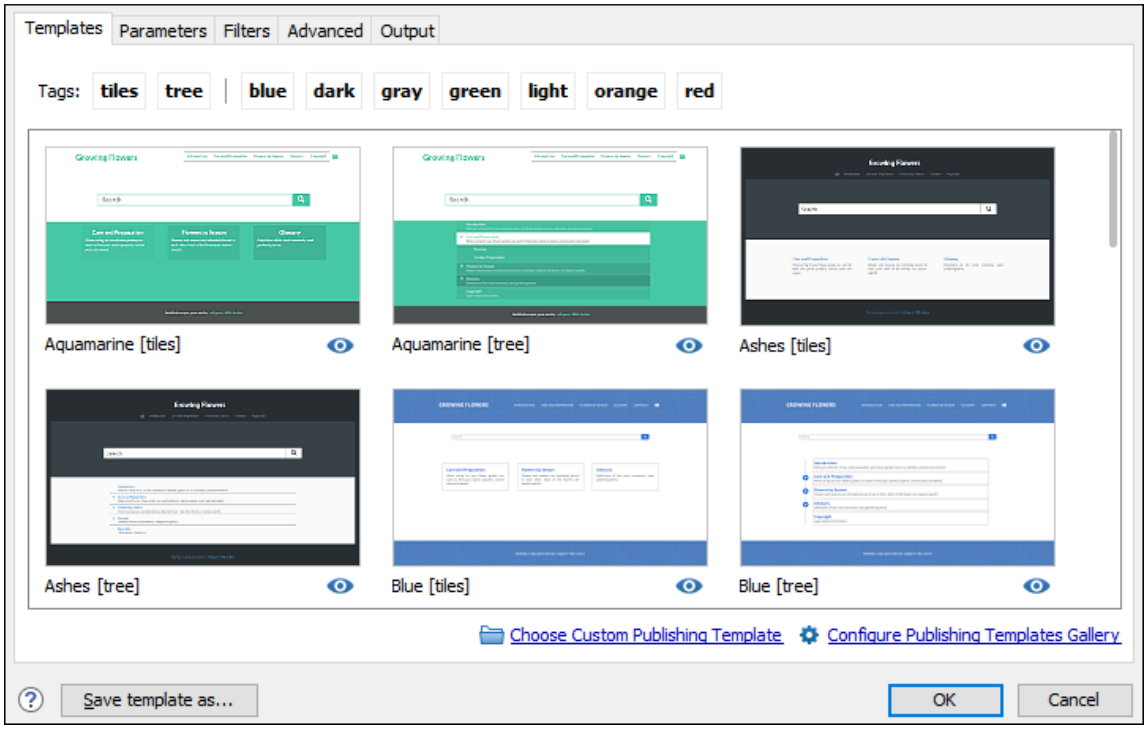
The lower part of the dialog box contains several tabs that allow you to configure the options that control the transformation.

Templates Tab (DITA-OT Transformations)

When you create a new transformation scenario *(on page 1101)* or edit an existing one *(on page 1207)*, 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 1267)*. You can use one of them to publish your documentation or as a starting point for a new publishing template.

Figure 374. Templates Tab



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 Developer 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 Developer scans the locations specified in the **DITA > Publishing** preferences page *(on page 203)* 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.

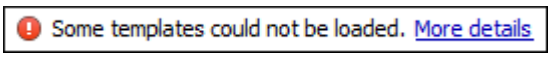
You can also add your custom templates *(on page 1324)* to the list of templates displayed in the **Templates** tab. To do this, store them in a directory, then click the **Configure Publishing Templates Gallery** link to open the **DITA > Publishing** preferences page *(on page 203)* where you can add 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 1464)*. Clicking this button will open a template package configuration dialog box *(on page)* 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:



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 203)* 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`).

Resources

For more information about customizing publishing templates, watch our video demonstration:

https://www.youtube.com/embed/zNmXfKWXwO8

Related Information:

Publishing Templates *(on page 1267)*
Publishing Template Package Contents for PDF Customizations *(on page 1459)*
Publishing Template Package Contents for WebHelp Responsive Customizations *(on page 1271)*

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 1464)*. 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 1073)* or **DITA Map to PDF - based on HTML5 & CSS** *(on page 1087)*). 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 1287)* into your template package. They are helpful if you want to change the structure of the generated HTML pages.

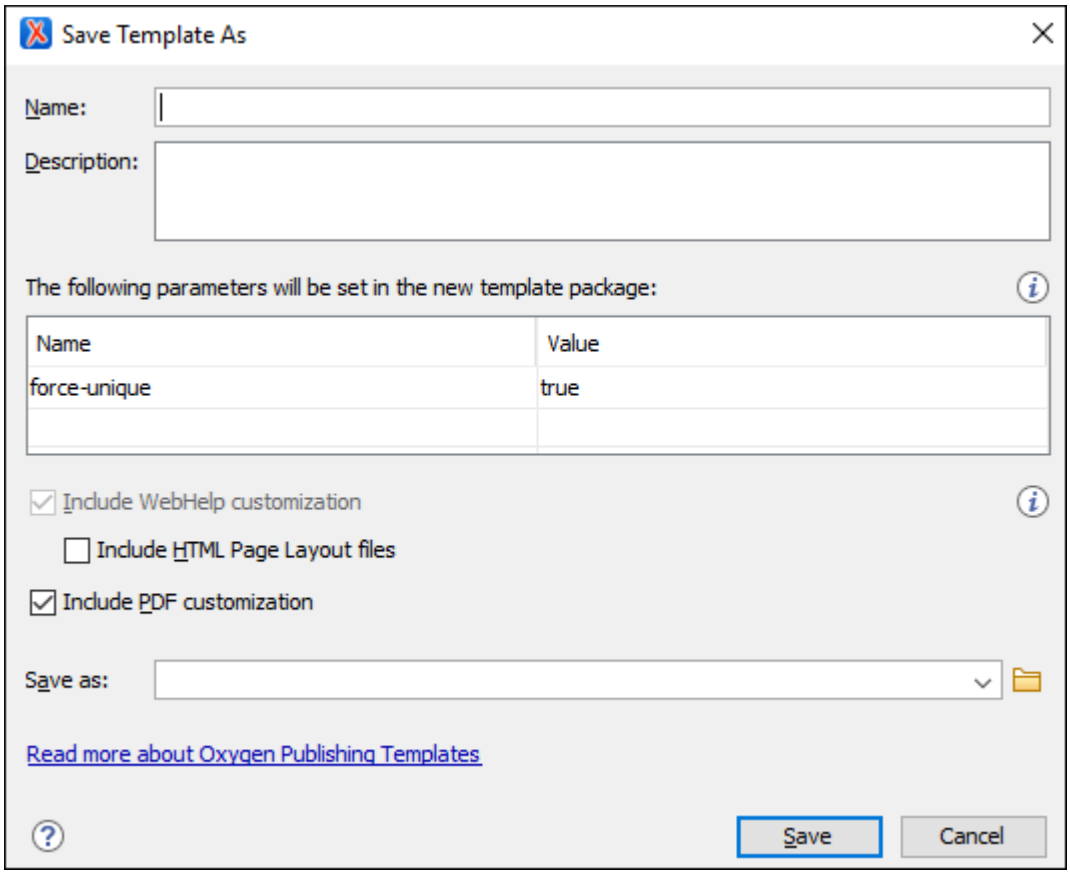
Include PDF Customization

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 1073)* or **DITA Map to PDF - based on HTML5 & CSS** *(on page 1087)*). 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 375. Template Package Configuration Dialog Box



Related Information:

Publishing Templates *(on page 1267)*

Publishing Template Package Contents for PDF Customizations *(on page 1459)*

Publishing Template Package Contents for WebHelp Responsive Customizations *(on page 1271)*

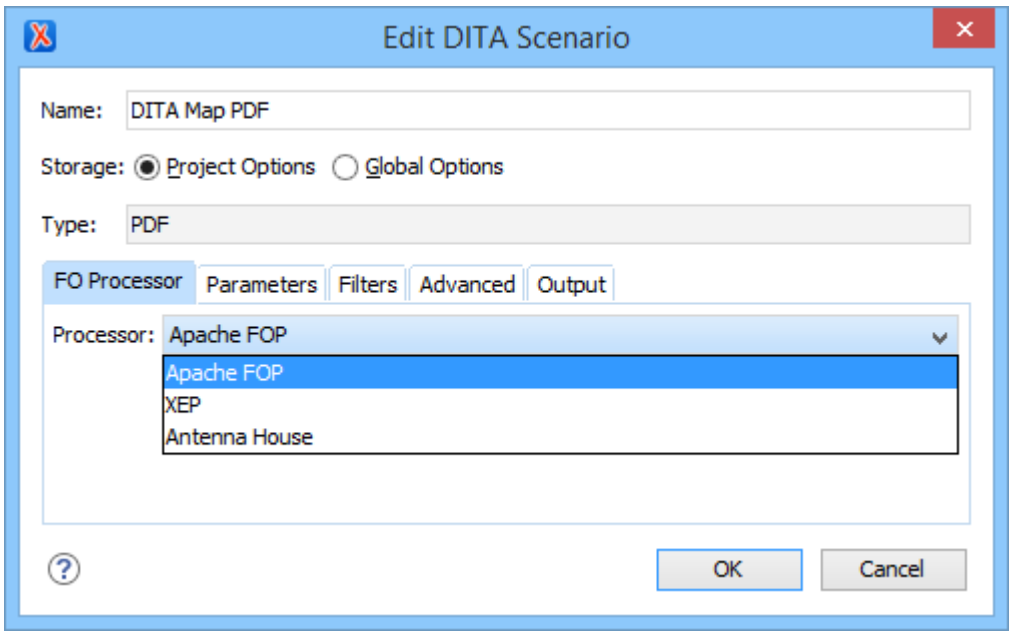
FO Processor Tab (DITA-OT Transformations)

When you create a new transformation scenario *(on page 1101)* or edit an existing one *(on page 1207)*, 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 376. FO Processor Configuration Tab



You can choose one of the following processors:

Apache FOP

The default processor that comes bundled with Oxygen XML Developer.

XEP

The RenderX XEP processor. If XEP is already installed, Oxygen XML Developer 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 192)*.
- 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 Developer installation directory.

Antenna House

The Antenna House (AH Formatter) processor. If Antenna House is already installed, Oxygen XML Developer 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 Developer 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)*.
3. Add the `env.AXF_OPT` parameter and point to the Antenna House configuration file.

Related information

FO Processors Preferences *(on page 192)*

XSL-FO (Apache FOP) Processor for Generating PDF Output *(on page 1174)*

Parameters Tab (DITA-OT Transformations)

When you create a new transformation scenario *(on page 1101)* or edit an existing one *(on page 1207)*, 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 252)* 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 253)* 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 252)* 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 1457)* descriptor file are displayed in italics. After creating a publishing template *(on page 1464)* and adding it to the templates gallery *(on page 1324)*, when you select the template in the **Templates** tab *(on page)*, 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 1101)* or edit an existing one *(on page 1207)*, 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 1457)*, either as an HTML fragment extension point *(on page 1278)* or as a transformation parameter *(on page 1276)* (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 1101)* or edit an existing one *(on page 1207)*, 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 2304)*, 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 252)* 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)*), the filters are combined (neither file takes precedence over the other).

Exclude from output all elements with any of the following attributes

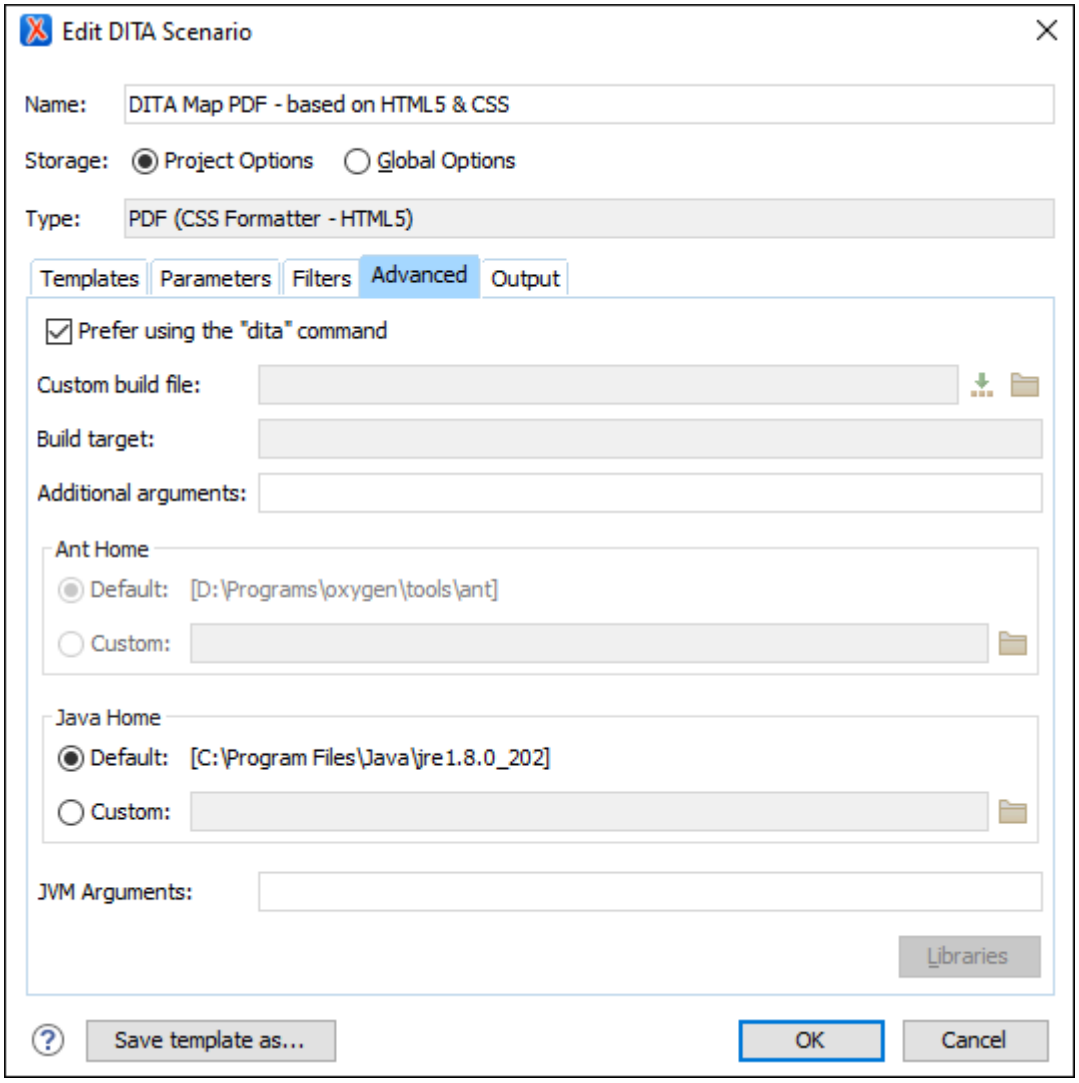
By using the **New**, **Edit**, or **Delete** buttons at the bottom of the pane, you can configure a list of attributes (name and value) to exclude all elements that contain any of these attributes from the output.

Advanced Tab (DITA-OT Transformations)

When you create a new transformation scenario *(on page 1101)* or edit an existing one *(on page 1207)*, 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.

Figure 377. Advanced Settings Tab



You can specify the following options:

Prefer using the "dita" command

When selected, Oxygen XML Developer 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)* is used. You can specify the path by using the text field, the **Insert Editor Variables** *(on page 252)* 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 197)*.

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 Developer.

Note:
It may be possible that the used Java version is incompatible with the DITA Open Toolkit engine. 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 about a quarter of the total memory available on the machine.

Libraries

By default, Oxygen XML Developer 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 2306)* files or additional class paths) to be used by the transformer.

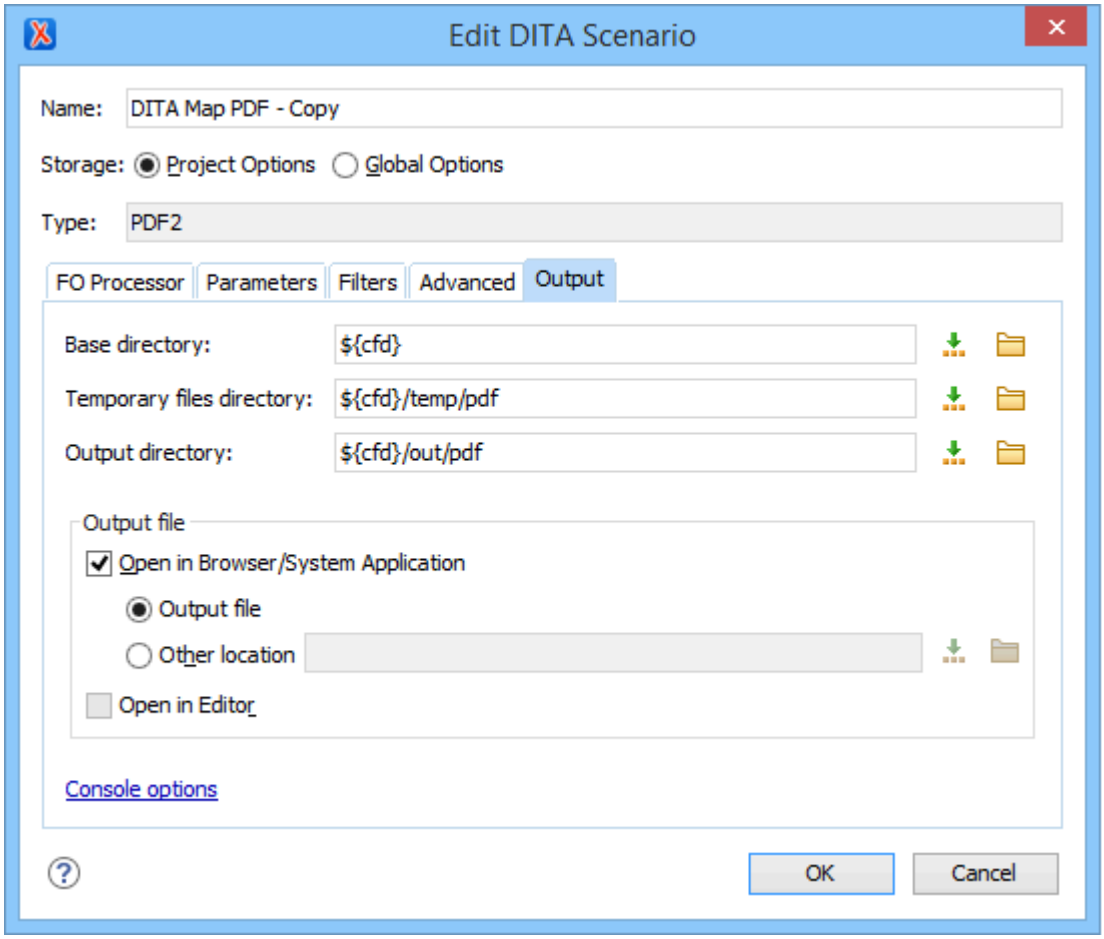
Tip:
You can specify the path to the additional libraries using wildcards (for example, `${oxygenHome}/lib/*.jar`).

Output Tab (DITA-OT Transformations)

When you create a new transformation scenario *(on page 1101)* or edit an existing one *(on page 1207)*, 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 378. 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 252)* 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 252)* 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 252)* button, or the **Browse** button.

Note:
If the *DITA map (on page 2304)* 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 Developer 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 252)* 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).

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 Developer 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 2303)* version 1.10.9) 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 1141)* 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 1217)*. 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 2308)* - 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 2305)* - 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.

Since **Ant** transformation scenarios are run as separate processes, to instruct Oxygen XML Developer to open certain resources created by the separate process, you can use `<echo>` elements in your build files with the following syntax:

```
<echo>Finished publishing: "absolute/path/to/file.ext"</echo>
```

Related Information:
Transforming Ant Build Files *(on page 677)*

Options Tab (Ant Transformations)

When you create a new transformation scenario *(on page 1101)* or edit an existing one *(on page 1207)*, 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 252)* 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 252)* 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 197)*.

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 Developer.

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 about a quarter of the total memory available on the machine.

Libraries

By default, Oxygen XML Developer 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 2306)* 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 1101)* or edit an existing one *(on page 1207)*, 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 252)* 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 253)* 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 252)* 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 2303)*.

Related Information:
Content Completion in Ant Build Files *(on page 678)*

Output Tab (Ant Transformations)

When you create a new transformation scenario *(on page 1101)* or edit an existing one *(on page 1207)*, 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 252)* button, or the **Browse** 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.

JSON Transformation with XSLT

This type of transformation specifies the transformation parameters and location of an XSLT stylesheet that is applied to the edited JSON document. This scenario is useful when you develop a JSON document and the XSLT document is in its final form.

To create a **JSON 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 **JSON transformation with XSLT**.
- Go to **Window > Show View** and select **Transformation Scenarios** to display this view *(on page 1217)*. Click the **New Scenario** drop-down menu button and select **JSON 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 2308)* - 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 2305)* - 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 (JSON Transformations)

When you create a new transformation scenario *(on page 1101)* or edit an existing one *(on page 1207)*, 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:

JSON URL

Specifies the source JSON file. You can specify the path by using the text field, its history drop-down, the **Insert Editor Variables** *(on page 252)* 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.

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 252)* 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 Developer for performing a transformation. 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 1106)* for the current transformation scenario. To configure the same options globally, go to the Saxon-

HE/PE/EE preferences page *(on page 181)*. 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 1104)* 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 1146)*. 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 1105)* 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 1105)* 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 1105)*, 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 252)* (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 252)* 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 252)* 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 252)*

XSLT Extensions

The **Extensions** button opens a dialog box that allows you to specify the *JARS (on page 2306)* 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 252)* 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.

FO Processor Tab (JSON Transformations)

When you create a new transformation scenario *(on page 1101)* or edit an existing one *(on page 1207)*, 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 192)*.

Output Tab (JSON Transformations)

When you create a new transformation scenario *(on page 1101)* or edit an existing one *(on page 1207)*, 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 252)* button, or the **Browse** button.

Open in Browser/System Application

If selected, Oxygen XML Developer 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 252)* 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 Developer displays the transformation result in an XML viewer panel at the bottom of the application window with syntax highlighting *(on page 158)*.
- **SVG** - If this is selected, Oxygen XML Developer displays the transformation result in an integrated SVG viewer in the **Results** panel *(on page 1009)* 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 Developer 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 252)* 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.

JSON Transformation with XQuery

This type of transformation specifies the transformation parameters and location of an XQuery file that is applied to the edited JSON document. This scenario is useful when you develop a JSON document and the XQuery file is in its final form.

To create a **JSON 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 **JSON transformation with XQuery**.
- Go to **Window > Show View** and select **Transformation Scenarios** to display this view *(on page 1217)*. Click the **New Scenario** drop-down menu button and select **JSON 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 2308)* - 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 2305)* - 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 (JSON Transformations)

When you create a new transformation scenario *(on page 1101)* or edit an existing one *(on page 1207)*, 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:

JSON URL

Specifies the source JSON file. You can specify the path by using the text field, its history drop-down, the **Insert Editor Variables** *(on page 252)* 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.

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 252)* 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 Developer for performing a transformation. These include the built-in engines and the external engines defined in the Custom Engines preferences page *(on page 191)*. 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 1120)* for the current transformation scenario. To configure the same options globally, go to the Saxon-HE/PE/EE preferences page *(on page 181)*. 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 1119)* 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 can not 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 1120)* 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 252)* (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 252)* 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 252)* 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 252)*

XQuery Extensions

The **Extensions** button is used to specify the *JAR (on page 2306)* 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 186)* 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 2305)* defined in the preferences page.

Saxon-HE/PE/EE Options

The advanced options for Saxon 12.9 Home Edition (HE), Professional Edition (PE), and Enterprise Edition (EE) are as follows:

Use a configuration file ("-config")

Sets a Saxon 12.9 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 1123)* is not set.

Saxon-PE/EE Options

The following advanced options are specific for Saxon 12.9 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 12.9 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 1101)* or edit an existing one *(on page 1207)*, 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 192)*.

Output Tab (XQuery Transformations)

When you create a new transformation scenario *(on page 1101)* or edit an existing one *(on page 1207)*, 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 252)* button, or the **Browse** button.

Open in Browser/System Application

If selected, Oxygen XML Developer 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 252)* 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 Developer displays the transformation result in an XML viewer panel at the bottom of the application window with syntax highlighting *(on page 158)*.
- **SVG** - If this is selected, Oxygen XML Developer displays the transformation result in an integrated SVG viewer in the **Results** panel *(on page 1009)* 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 Developer 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 252)* button, or the **Browse** button.

XSLT Transformation on XML

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 on XML** 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 **XSLT transformation on XML**.
- Go to **Window > Show View** and select **Transformation Scenarios** to display this view *(on page 1217)*. Click the **New Scenario** drop-down menu button and select **XSLT transformation on XML**.

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 2308)* - 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 2305)* - 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 1101)* or edit an existing one *(on page 1207)*, 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 252)* 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 183)*, 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 1160)* 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 1116)*. 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 252)* 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 Developer for performing a transformation. These include the built-in engines and the external engines defined in the Custom Engines preferences page *(on page 191)*. 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 1106)* for the current transformation scenario. To configure the same options globally, go to the Saxon-HE/PE/EE preferences page *(on page 181)*. 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 1104)* 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 1161)*. 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 1105)* 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 1105)* 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 1105)*, 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 252)* (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 252)* 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 252)* 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 252)*

XSLT Extensions

The **Extensions** button opens a dialog box that allows you to specify the *JARS (on page 2306)* 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 252)* 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 181)* 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 2305)* defined in the preferences page.

Saxon-HE/PE/EE Options

The advanced options for Saxon 12.9 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 1164)*, 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 1164)*, the value in this option takes precedence.

> **Tip:**
> If your stylesheet includes `<xsl:template name="xsl:initial-template">`, Oxygen XML Developer 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 12.9 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.

> **Attention:**
> Oxygen XML Developer does not support the `ALLOWED_PROTOCOLS` Saxon property (*http://saxon.sf.net/feature/allowedProtocols*). This feature is specific to Saxon and controls access by Saxon (but not by underlying software, such as the XML parser). For more information, see https://www.saxonica.com/documentation12/index.html#!configuration/config-features.

Debugger trace into XPath expressions (applies to debugging sessions)

Instructs the XSLT Debugger *(on page 1845)* 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 12.9 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 12.9 processors.

> **Note:**
> Saxon-CE, being JavaScript-based, was designed to run inside a web browser. This means that you will use Oxygen XML Developer 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 12.9 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 79)* and go to **XML > XML Parser > XML Schema** and use the **Default XML Schema version** option *(on page 172)*.

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. However, in certain instances, the Saxon XSLT processor may auto detect and use streaming even if this option is not 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.

Using Saxon Integrated Extension Functions

Saxon, the transformation and validation engine used by Oxygen XML Developer, 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:

```
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 Developer 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 99)*.
- In a validation scenario *(on page 526)*, 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 1104)* 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 1857)*.

FO Processor Tab (XSLT Transformations)

When you create a new transformation scenario *(on page 1101)* or edit an existing one *(on page 1207)*, 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 192)*.

Output Tab (XSLT Transformations)

When you create a new transformation scenario *(on page 1101)* or edit an existing one *(on page 1207)*, 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 252)* button, or the **Browse** button.

Open in Browser/System Application

If selected, Oxygen XML Developer 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 252)* 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 Developer displays the transformation result in an XML viewer panel at the bottom of the application window with syntax highlighting *(on page 158)*.
- **SVG** - If this is selected, Oxygen XML Developer displays the transformation result in an integrated SVG viewer in the **Results** panel *(on page 1009)* 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 Developer 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 252)* 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 Developer.

Supported XSLT Processors

Oxygen XML Developer includes the following XSLT processors:

- **Xalan 2.7.2** (Deprecated) - 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 12.9 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 12.9 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 Developer. 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 172)*.

  Note:
  Oxygen XML Developer 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 Developer 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:
  1. Download Saxon 12.9 Professional Edition (PE) or Enterprise Edition (EE) from http://www.saxonica.com.
  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 Developer.

- **Saxon-CE (Client Edition)** is Saxonica's implementation of XSLT 2.0 for use on web browsers. Oxygen XML Developer 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 Developer 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.

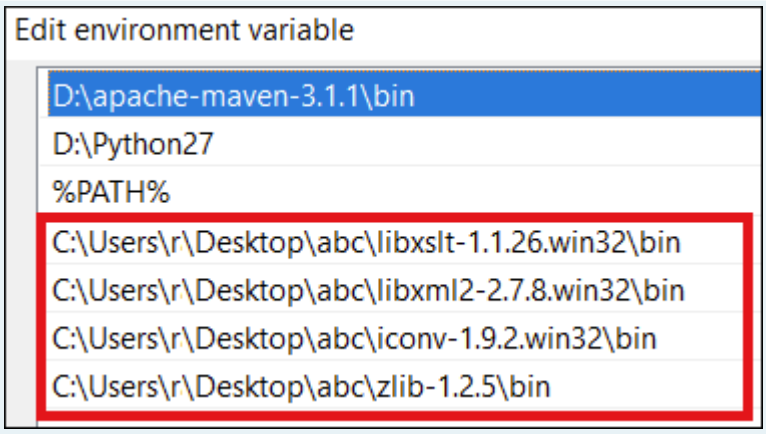
Note:
A specific template, named **Saxon-CE stylesheet**, is available in the **New** document wizard *(on page 299)*.

- **Xsltproc (libxslt)** (Deprecated) - 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 Developer 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:

1. Go to http://ftp.zlatkovic.com/libxml.en.html and download the following ZIP files: **iconv-1.9.2.win32.zip**, **libxml2-2.7.8.win32.zip**, **libxslt-1.1.26.win32.zip**, **zlib-1.2.5.win32.zip**.
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:



4. Restart Oxygen XML Developer.

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 184)*.

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 2309)* files of the built-in document types (DocBook, TEI, DITA, etc.) are not handled properly by LIBXML if Oxygen XML Developer 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.

For information about configuring the XSLT preferences, see the XSLT options *(on page 180)* section.

Configuring Custom XSLT Processors

Oxygen XML Developer 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 79)* 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 191)*.
4. Click **OK**.

Note:

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 Developer linked message *(on page 523)*, 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 191)*

Configuring the XSLT Processor Extensions Paths

The Saxon and Xalan 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 (Deprecated)** - http://xml.apache.org/xalan-j/extensions.html
- **Saxon 6.5.5** - http://saxon.sourceforge.net/saxon6.5.5/extensions.html
- **Saxon 12.9** - 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 1104)* in the **Edit scenario** dialog box.

XSL-FO (Apache FOP) Processor for Generating PDF Output

The Oxygen XML Developer 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 192)*.

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 1176)*.

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 79)*, 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 1102)* 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 1102)* 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 `char-set="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 1174)*.

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.

   Note:
   DITA PDF transformations have their own `fop.xconf` (*`DITA-OT-DIR`*`/plugins/org.dita.pdf2.fop/fop/conf/fop.xconf`). If the font is not installed in the system, it needs to be referenced in the `fop.xconf`.

   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 79)*, 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 1102)*, 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 1102)*, 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:

- *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 `char-set="default"` attribute must contain the name of the font.
- *DITA-OT-DIR*`/plugins/org.dita.pdf2/fop/conf/fop.xconf` - 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 Developer 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 Developer, 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 Developer, follow this procedure:

1. Create a folder called `fop` in the *`[OXYGEN_INSTALL_DIR]`*`/lib` folder.
2. Download the compiled *JAR (on page 2306)* from OFFO.
3. Copy the `fop-hyph.jar` file into the *`[OXYGEN_INSTALL_DIR]`*`/lib/fop` folder.
4. Restart Oxygen XML Developer.

Adding Support for PDF Images

To add support for PDF images in an older version of Oxygen XML Developer, 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 Developer.

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 Developer 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.

XSLT Transformation on JSON

This type of transformation specifies the parameters and location of a JSON document that the edited XSLT stylesheet is applied on. This scenario is useful when you develop an XSLT document and the JSON document is in its final form.

To create an **XSLT transformation on JSON** 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 **XSLT transformation on JSON**.
- Go to **Window > Show View** and select **Transformation Scenarios** to display this view *(on page 1217)*. Click the **New Scenario** drop-down menu button and select **XSLT transformation on JSON**.

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 2308)* - 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 2305)* - 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 (JSON Transformations)

When you create a new transformation scenario *(on page 1101)* or edit an existing one *(on page 1207)*, 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:

JSON URL

Specifies the source JSON file. You can specify the path by using the text field, its history drop-down, the **Insert Editor Variables** *(on page 252)* 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.

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 252)* 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 Developer for performing a transformation. 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 1106)* for the current transformation scenario. To configure the same options globally, go to the Saxon-HE/PE/EE preferences page *(on page 181)*. 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 1104)* 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 1180)*. 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 1105)* 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 1105)* 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 1105)*, 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 252)* (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 252)* 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 252)* 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 252)*

XSLT Extensions

The **Extensions** button opens a dialog box that allows you to specify the *JARS (on page 2306)* 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 252)* 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.

FO Processor Tab (JSON Transformations)

When you create a new transformation scenario *(on page 1101)* or edit an existing one *(on page 1207)*, 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 192)*.

Output Tab (JSON Transformations)

When you create a new transformation scenario *(on page 1101)* or edit an existing one *(on page 1207)*, 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 252)* button, or the **Browse** button.

Open in Browser/System Application

If selected, Oxygen XML Developer 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 252)* 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 Developer displays the transformation result in an XML viewer panel at the bottom of the application window with syntax highlighting *(on page 158)*.
- **SVG** - If this is selected, Oxygen XML Developer displays the transformation result in an integrated SVG viewer in the **Results** panel *(on page 1009)* 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 Developer 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 252)* 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.

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 1217)*. 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 2308)* - 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 2305)* - 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:
Integrating an External XProc Engine *(on page 1189)*
Editing XProc Scripts *(on page 946)*

XProc Tab

When you create a new transformation scenario *(on page 1101)* or edit an existing one *(on page 1207)*, 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 252)* 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 choose from **XML Calabash 1.X**, **XML Calabash 3.X**, or a custom engine that is configured in the **XProc Preferences** page *(on page 178)*.

Configuration File

You can specify a configuration file to be sent to the XProc Transformer. You can specify the path by using the text field, history drop-down, **Insert Editor Variables** *(on page 252)* button, or the **Browse** button.

Extensions

Opens a dialog box where you can specify additional extension libraries to be added to the class path used in the XProc transformation.

Inputs Tab (XProc Transformations)

When you create a new transformation scenario *(on page 1101)* or edit an existing one *(on page 1207)*, 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 252)* and custom editor variables *(on page 261)* 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 252)* and custom editor variables *(on page 261)* 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 1101)* or edit an existing one *(on page 1207)*, a configuration dialog box is displayed that allows you to customize the transformation with various options in several tabs.



Notice:
This tab is not available for XProc 3.0, since that version does not use parameters.

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 252)* and custom editor variables *(on page 261)* 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 1101)* or edit an existing one *(on page 1207)*, 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 252)* 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 474)*.

Edit

Opens an **Edit** dialog box that allows you to edit an existing output port and its URL. An editor variable *(on page 252)* 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 474)*.

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

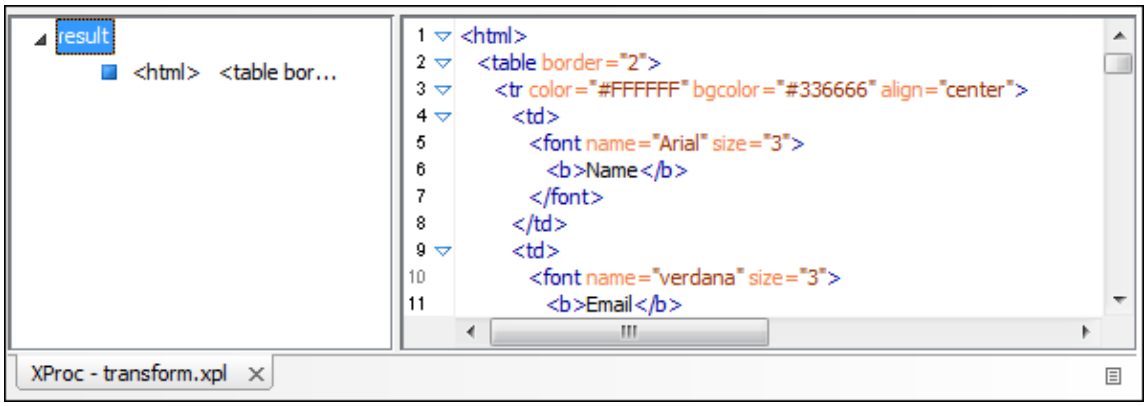
specify the path by using the text field, its history drop-down, the **Insert Editor Variables** *(on page 252)* button, or the browsing actions in the **Browse** drop-down list.

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 379. XProc Transformation Results View



Options Tab (XProc Transformations)

When you create a new transformation scenario *(on page 1101)* or edit an existing one *(on page 1207)*, 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 252)* and custom editor variables *(on page 261)* 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. Uncommment the `<system-property name="com.xmlcalabash.fo-processor" value="com.xmlcalabash.util.FoXEP"/>` system property.
3. Uncommment 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. Uncommment the references to the XEP libraries. Edit them to point to the matching library names from the XEP installation directory.
5. Restart Oxygen XML Developer.

Integrating an External XProc Engine

Oxygen XML Developer 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 178)*. 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 Developer: 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*.

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 2306)* 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 one or more `<library>` elements with the `@name` attribute that contains either of the following:
- The relative or absolute location of the libraries necessary to run this integration.
- A folder name (for example, `lib/`) that contains the libraries necessary to run this integration. Note that the folder name must end in a slash character (**/**).

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 **AditionalXProcEngine** extension *(on page 1869)*):

```
<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>
```

Related Information:
Editing XProc Scripts *(on page 946)*
Creating an XProc Transformation Scenario *(on page 1184)*
Additional XProc Engine Plugin Extension *(on page 1869)*

XQuery Transformation on XML

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 on XML** 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 on XML**.
- Go to **Window > Show View** and select **Transformation Scenarios** to display this view *(on page 1217)*. Click the **New Scenario** drop-down menu button and select **XQuery transformation on XML**.

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 2308)* - 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 2305)* - 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 1101)* or edit an existing one *(on page 1207)*, 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 252)* 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 183)*, 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 252)* 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 Developer for performing a transformation. These include the built-in engines and the external engines defined in the Custom Engines preferences page *(on page 191)*. 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 1120)* for the current transformation scenario. To configure the same options globally, go to the Saxon-HE/PE/EE preferences page *(on page 181)*. 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 1119)* 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 can not 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 1120)* 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 252)* (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 252)* 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 252)* 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 252)*

XQuery Extensions

The **Extensions** button is used to specify the *JAR (on page 2306)* 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 186)* 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 2305)* defined in the preferences page.

Saxon-HE/PE/EE Options

The advanced options for Saxon 12.9 Home Edition (HE), Professional Edition (PE), and Enterprise Edition (EE) are as follows:

Use a configuration file ("-config")

Sets a Saxon 12.9 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 1123)* is not set.

Saxon-PE/EE Options

The following advanced options are specific for Saxon 12.9 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 12.9 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 1101)* or edit an existing one *(on page 1207)*, 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 192)*.

Output Tab (XQuery Transformations)

When you create a new transformation scenario *(on page 1101)* or edit an existing one *(on page 1207)*, 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 252)* button, or the **Browse** button.

Open in Browser/System Application

If selected, Oxygen XML Developer 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 252)* 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 Developer displays the transformation result in an XML viewer panel at the bottom of the application window with syntax highlighting *(on page 158)*.
- **SVG** - If this is selected, Oxygen XML Developer displays the transformation result in an integrated SVG viewer in the **Results** panel *(on page 1009)* 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 Developer 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 252)* button, or the **Browse** button.

XQuery Transformation on JSON

This type of transformation specifies the parameters and location of a JSON document that the edited XQuery file is applied on. This scenario is useful when you develop an XQuery file and the JSON document is in its final form.

To create an **XQuery transformation on JSON** 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 on JSON**.
- Go to **Window > Show View** and select **Transformation Scenarios** to display this view *(on page 1217)*. Click the **New Scenario** drop-down menu button and select **XQuery transformation on JSON**.

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 2308)* - 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 2305)* - 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 (JSON Transformations)

When you create a new transformation scenario *(on page 1101)* or edit an existing one *(on page 1207)*, 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:

JSON URL

Specifies the source JSON file. You can specify the path by using the text field, its history drop-down, the **Insert Editor Variables** *(on page 252)* 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.

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 252)* 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 Developer for performing a transformation. These include the built-in engines and the external engines defined in the Custom Engines preferences page *(on page 191)*. 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 1120)* for the current transformation scenario. To configure the same options globally, go to the Saxon-HE/PE/EE preferences page *(on page 181)*. 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 1119)* 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 can not 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 1120)* 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 252)* (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 252)* 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 252)* 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 252)*

XQuery Extensions

The **Extensions** button is used to specify the *JAR (on page 2306)* 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 186)* 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 2305)* defined in the preferences page.

Saxon-HE/PE/EE Options

The advanced options for Saxon 12.9 Home Edition (HE), Professional Edition (PE), and Enterprise Edition (EE) are as follows:

Use a configuration file ("-config")

Sets a Saxon 12.9 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 1123)* is not set.

Saxon-PE/EE Options

The following advanced options are specific for Saxon 12.9 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 12.9 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 1101)* or edit an existing one *(on page 1207)*, 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 192)*.

Output Tab (XQuery Transformations)

When you create a new transformation scenario *(on page 1101)* or edit an existing one *(on page 1207)*, 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 252)* button, or the **Browse** button.

Open in Browser/System Application

If selected, Oxygen XML Developer 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 252)* 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 Developer displays the transformation result in an XML viewer panel at the bottom of the application window with syntax highlighting *(on page 158)*.
- **SVG** - If this is selected, Oxygen XML Developer displays the transformation result in an integrated SVG viewer in the **Results** panel *(on page 1009)* 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 Developer 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 252)* 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 1217)*. 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 2308)* - 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 2305)* - 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** *(on page 252)* 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.

Connection

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 205)*.

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 2305)* are read-only, to edit one of these scenarios you will need to duplicate it and edit the duplicated scenario *(on page 1209)*.

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.

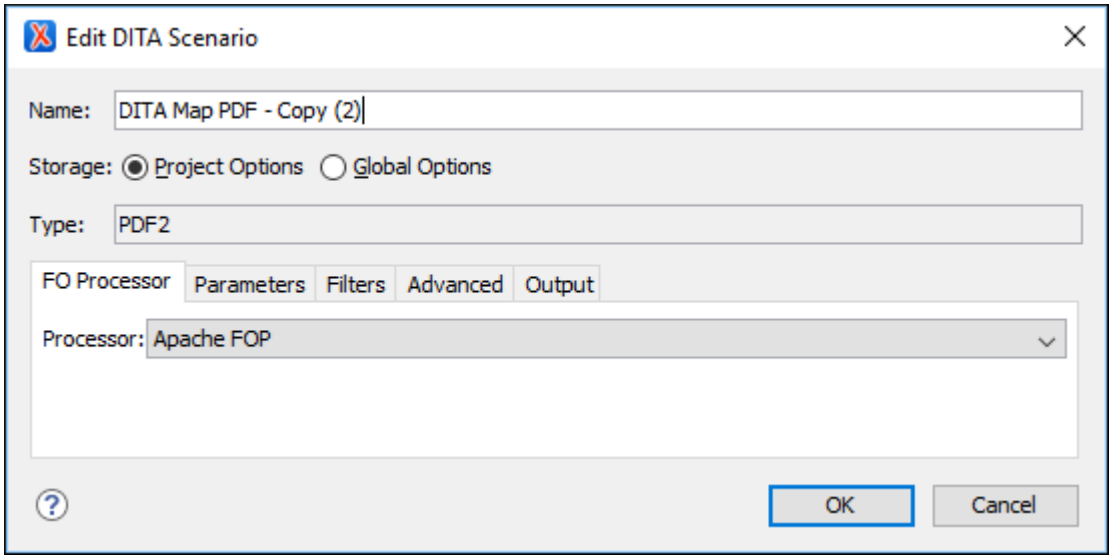
   Step Result: The **Configure Transformation Scenario(s)** dialog box *(on page 1210)* is opened.

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 1217)* 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.

Figure 380. Edit Scenarios Configuration Dialog Box



Transformation Types

The **Configure Transformation Scenario(s)** dialog box *(on page 1210)* 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)*
  - FO Processor Tab *(on page)* (Available for PDF output)
  - Parameters Tab *(on page)*
  - Filters Tab *(on page)*
  - Advanced Tab *(on page)*
  - Output Tab *(on page)*
- **ANT** - This type of transformation includes configurable options in the following tabs:

    - Options Tab *(on page 1141)*
    - Parameters Tab *(on page 1143)*
    - Output Tab *(on page 1144)*
- **XSLT** - This type of transformation includes configurable options in the following tabs:
    - XSLT Tab *(on page 1102)*
    - FO Processor Tab *(on page 1111)*
    - Output Tab *(on page 1112)*
- **XProc** - This type of transformation includes configurable options in the following tabs:
    - XProc Tab *(on page 1185)*
    - Inputs Tab *(on page 1186)*
    - Parameters Tab *(on page 1186)*
    - Outputs Tab *(on page 1187)*
    - Options Tab *(on page 1188)*
- **XQuery** - This type of transformation includes configurable options in the following tabs:
    - XQuery Tab *(on page 1118)*
    - FO Processor Tab *(on page 1123)*
    - Output Tab *(on page 1123)*

Related Information:

Creating New Transformation Scenarios *(on page 1101)*

Duplicating a Transformation Scenario *(on page 1209)*

Configure Transformation Scenario(s) Dialog Box *(on page 1210)*

Applying Associated Transformation Scenarios *(on page 1210)*

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 1210)* 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 1217)* 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 1208)*.

Related Information:
Creating New Transformation Scenarios *(on page 1101)*
Editing a Transformation Scenario *(on page 1207)*

Applying Associated Transformation Scenarios

If you have associated transformation scenarios for the current document (in the **Configure Transformation Scenario(s)** dialog box *(on page 1210)* or **Transformation Scenarios** view *(on page 1217)*), Oxygen XML Developer 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 1210)* 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 1217)*.
- Right-click a file in the **Project** view *(on page 334)* and select **Transform >** **Apply Transformation Scenario(s)**.
- Use the **Apply Associated** button in the **Configure Transformation Scenario(s)** dialog box *(on page 1210)*.

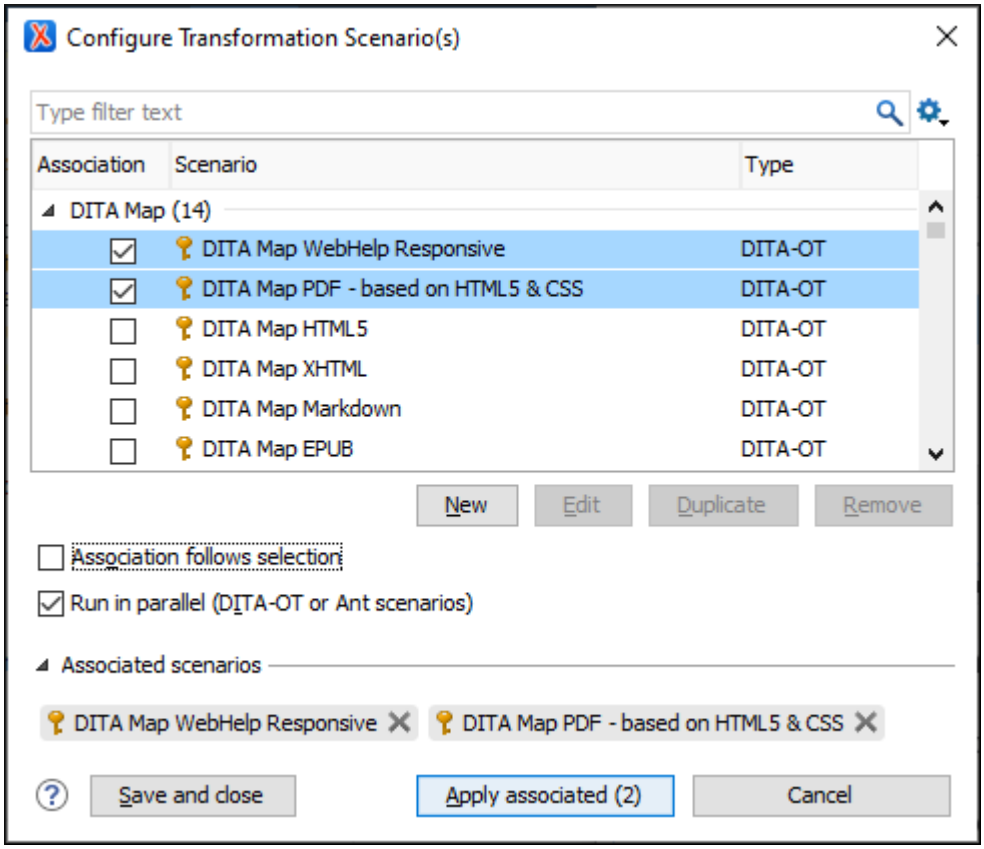
Related Information:
Creating New Transformation Scenarios *(on page 1101)*
Editing a Transformation Scenario *(on page 1207)*
Configure Transformation Scenario(s) Dialog Box *(on page 1210)*
Transformation Scenarios View *(on page 1217)*

Configure Transformation Scenario(s) Dialog Box

You can use the **Configure Transformation Scenarios(s)** dialog box for editing existing transformation scenarios *(on page 1207)* or creating new ones *(on page 1101)*.

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.

Figure 381. Configure Transformation Scenario(s) Dialog Box



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 Developer 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 Developer 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 Developer adds `imported` to the name of the imported scenario.

Export selected scenarios

Use this option to export selected scenarios individually. Oxygen XML Developer 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 2308)*. 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 Developer, see the Transformation Types section *(on page 1208)*.

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 1207)* that allows you to configure the options of the transformations scenario.

Duplicate

Use this button to create a duplicate transformation scenario *(on page 1209)*.

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 2308)* or **Global Options** *(on page 2305)*. 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 Developer 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 Developer adds `imported` to the name of the imported scenario.

Export selected scenarios

Use this option to export selected scenarios individually. Oxygen XML Developer 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 1101)*.

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 1208)*.

> **Note:**
> If you try to edit a transformation scenario associated with a defined document type, Oxygen XML Developer 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 1209)* to edit instead.

Duplicate

Use this button to create a duplicate transformation scenario *(on page 1209)*.

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 1207)*
Duplicating a Transformation Scenario *(on page 1209)*
Applying Associated Transformation Scenarios *(on page 1210)*
Creating New Transformation Scenarios *(on page 1101)*
Sharing Transformation Scenarios *(on page 1216)*

Batch Transformations

A transformation action can be applied on a batch of selected files from the contextual menu of the **Project** view *(on page 340)* 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 334)* 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:**
    > 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 334)* each time you want to transform them, but this can be tedious.

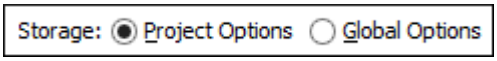
2. Select the files you want to transform (or the newly created logical folder) and from the contextual menu, select **Transform > Configure Transformation Scenario(s)** to choose one or more transformation scenarios to be applied on all the files in the logical folder.
3. Use Oxygen XML Developer editor variables *(on page 252)* 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 252)* is assigned for the input file. For example, for a *DocBook PDF transformation*, make sure the **XML URL** input box is set to the *${currentFileURL}* editor variable *(on page 258)*. For a DITA PDF transformation, make sure the `args.input` parameter is set to the *${cf}* editor variable *(on page 257)*.
    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 252)*, 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 252)*

Sharing Transformation Scenarios

The transformation scenarios and their settings can be shared with other users by saving them at project level *(on page 2308)* or by exporting them to a specialized scenarios file *(on page 251)* 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 2308)* or **Global Options** *(on page 2305)*.



Selecting **Project Options** *(on page 2308)* 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 2305)* stores the scenario in the global options that are stored in the user home directory.

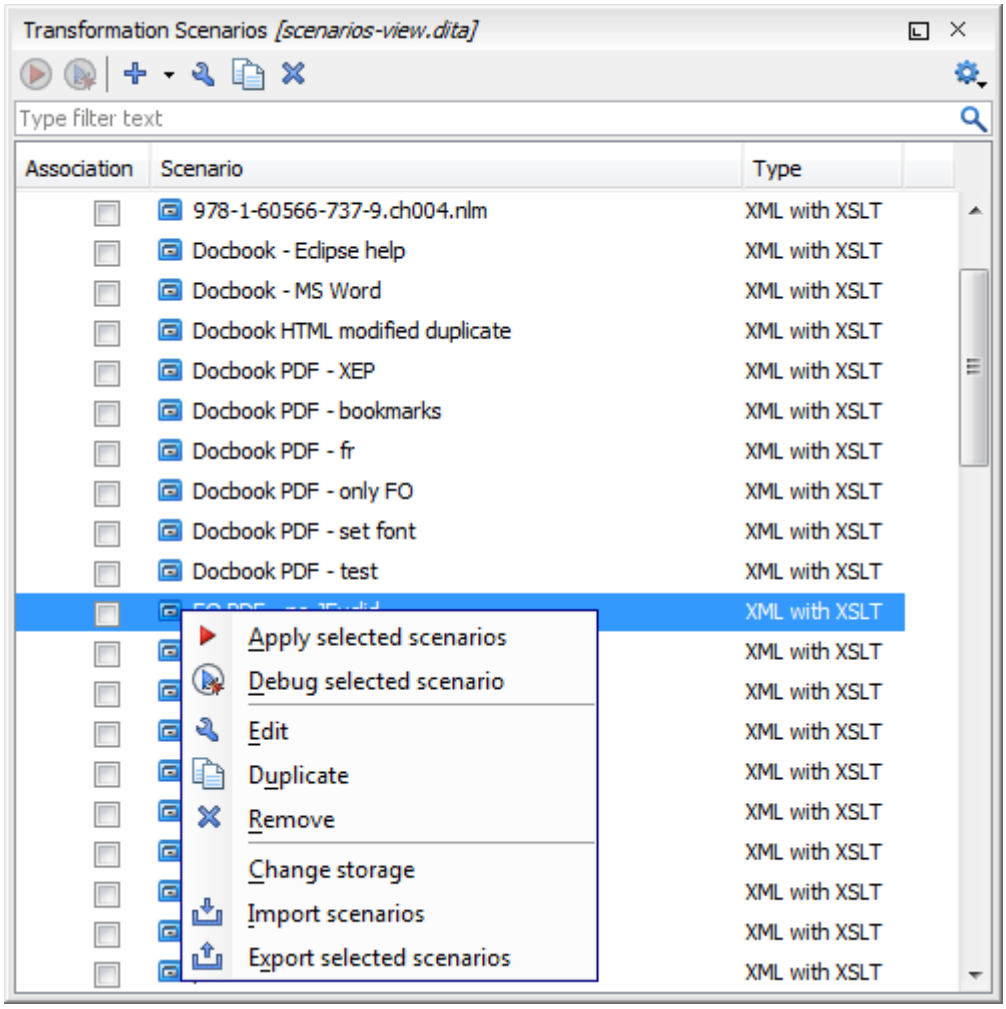
You can also change the storage options on existing transformation scenarios by using the **Change storage** action *(on page 1210)* from the contextual menu of the list of scenarios.

Related Information:
Sharing Application Settings *(on page 241)*

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**.

Figure 382. Transformation Scenarios view



Oxygen XML Developer 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 1210)*.

The **Transformation Scenarios** view presents both global and project-level *(on page 2308)* scenarios. By default, Oxygen XML Developer presents the items in the following order:

1. Scenarios that match the current *framework (on page 2305)*.
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 2307)* 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 1101)*. Oxygen XML Developer 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 1208)*.

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 Developer 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 Developer adds `imported` to the name of the imported scenario.

Export selected scenarios

Use this option to export transformation and validation scenarios individually. Oxygen XML Developer 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 Developer 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 2308)* or **Global Options** *(on page 2305)*. 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 Developer 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 Developer adds `imported` to the name of the imported scenario.

Export selected scenarios

Use this option to export selected scenarios individually. Oxygen XML Developer 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 1207)*
Creating New Transformation Scenarios *(on page 1101)*

WebHelp Output Customization

Oxygen XML WebHelp provides the ability to generate **WebHelp Responsive** from DITA XML content.

Table 35. WebHelp System Feature Matrix

Features	Responsive
Desktop Systems	✔
Mobile Devices	✔
Built-in Skins	✔
Built-in Templates	✔
Search Capabilities	✔
Modern Layout	✔
Adaptable to Any Screen Size	✔
Oxygen Feedback Commenting Platform	✔
DITA Documents	✔

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 1073)* 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 383. WebHelp Responsive Output on a Normal Screen

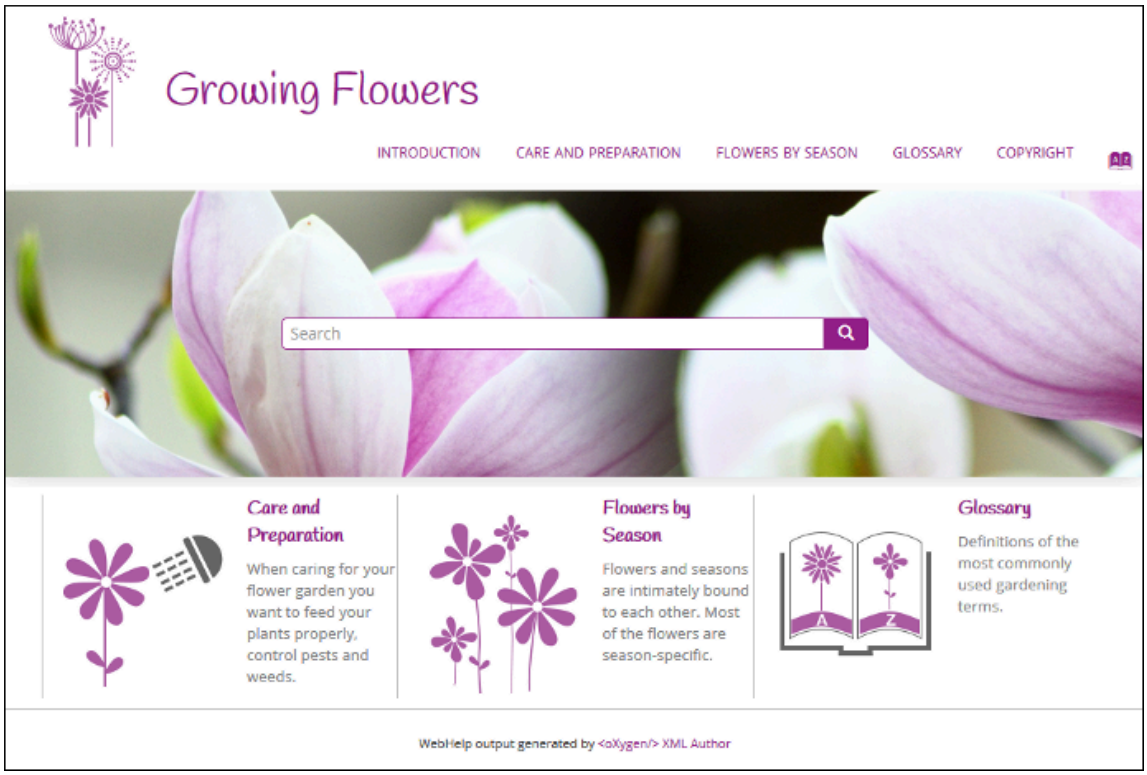


Figure 384. WebHelp Responsive Output on a Narrow Screen



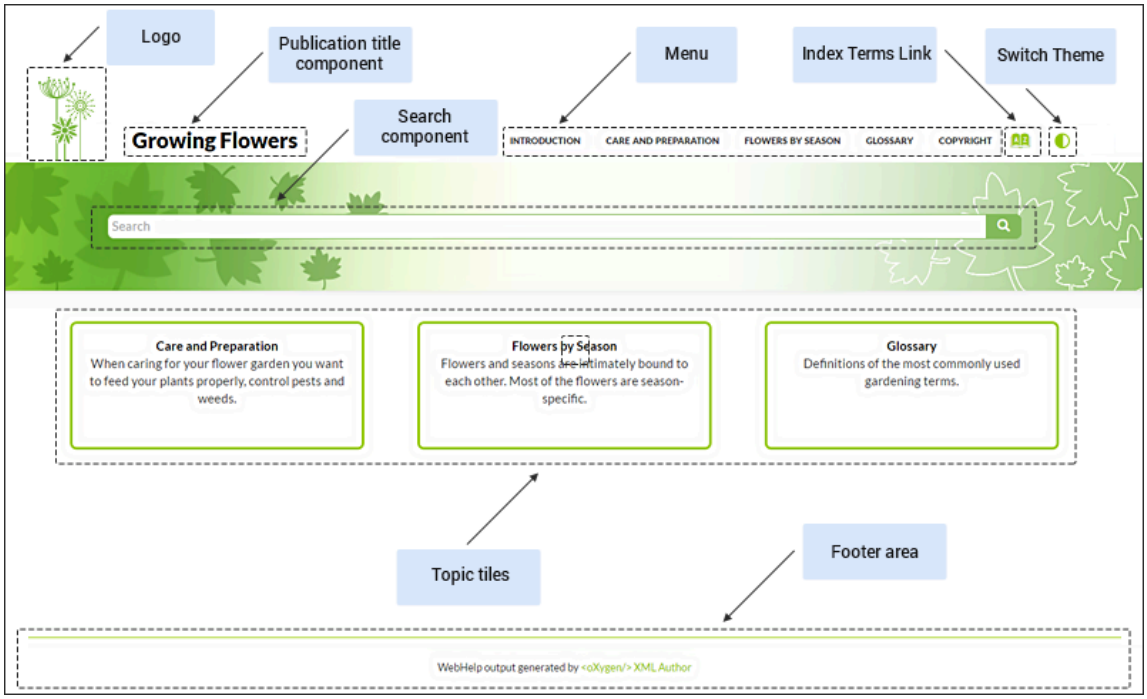
Main Page

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 385. Main Page - Tiles Layout

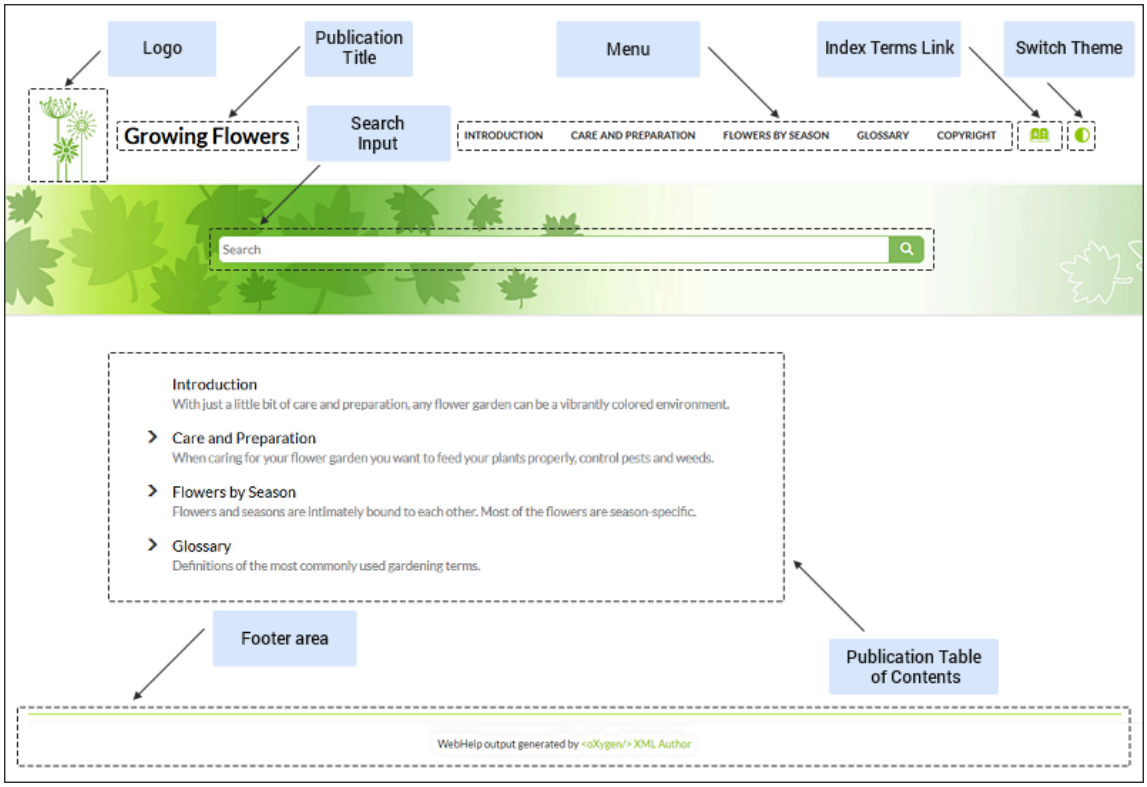


1. Logo Component *(on page 1225)*
2. Title Component *(on page 1225)*
3. Search Input Component *(on page 1226)*
4. Menu Component *(on page 1226)*
5. Index Terms Link Component *(on page 1226)*
6. Theme Button Component *(on page 1226)*
7. Topic Tiles Component *(on page 1226)*
8. Footer Component *(on page 1226)*

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 386. Main Page - Tree Layout



1. Logo Component *(on page 1225)*
2. Title Component *(on page 1225)*
3. Search Input Component *(on page 1226)*
4. Menu Component *(on page 1226)*
5. Index Terms Link Component *(on page 1226)*
6. Theme Button Component *(on page 1226)*
7. Table of Contents Component *(on page 1226)*
8. Footer Component *(on page 1226)*

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 1421)*. For the target URL, use the webhelp.logo.image.target.url parameter *(on page 1421)*.

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 1344)* 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 1431)*.

Switch Theme Button

Presents a button that opens a drop-down menu for selecting the theme. You can control if this component is displayed by using the webhelp.enable.dark.mode parameter *(on page 1431)*. For more information, see How to Create a Dark Mode-Ready Publishing Template *(on page 1312)*.

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 1344)* is done in the DITA map.

Tree Table of Contents

An area that contains first and second-level topic titles from your publication.

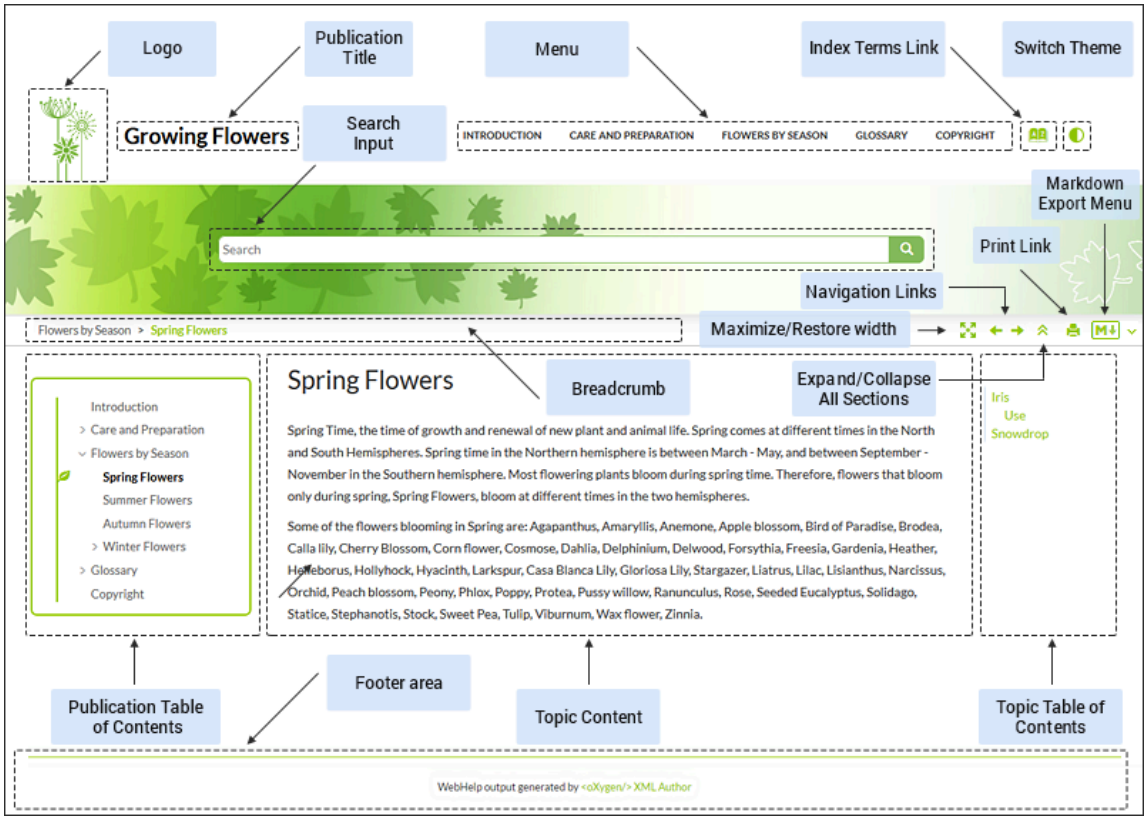
Page Footer

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 387. Topic Page



1. Logo Component *(on page 1228)*
2. Title Component *(on page 1227)*
3. Search Input Component *(on page 1228)*
4. Menu Component *(on page 1228)*
5. Index Terms Link Component *(on page 1228)*
6. Theme Button Component *(on page 1228)*
7. Maximize/Restore width Component *(on page 1228)*
8. Navigation Links Component *(on page 1228)*
9. Expand/Collapse All Sections Component *(on page 1228)*
10. Print Link Component *(on page 1229)*
11. Markdown Export Menu *(on page 1229)*
12. Breadcrumb Component *(on page 1229)*
13. Publication Table of Contents Component *(on page 1229)*
14. Topic Content Component *(on page 1229)*
15. Topic Table of Contents Component *(on page 1229)*
16. Footer Component *(on page 1229)*

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 1421)*. For the target URL, use the webhelp.logo.image.target.url parameter *(on page 1421)*.

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 1344)* 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 1431)*.

Switch Theme Button

Presents a button that opens a drop-down menu for selecting the theme. You can control if this component is displayed by using the webhelp.enable.dark.mode parameter *(on page 1431)*. For more information, see How to Create a Dark Mode-Ready Publishing Template *(on page 1312)*.

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`.

Maximize to full width/Restore to normal width Button

Icon that maximizes or restores the topic content to use the entire page space. Only visible when screen width exceeds 1600 pixels.

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.

Markdown Export Menu

Button that opens a menu with "Copy as Markdown" and "View as Markdown" actions.

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

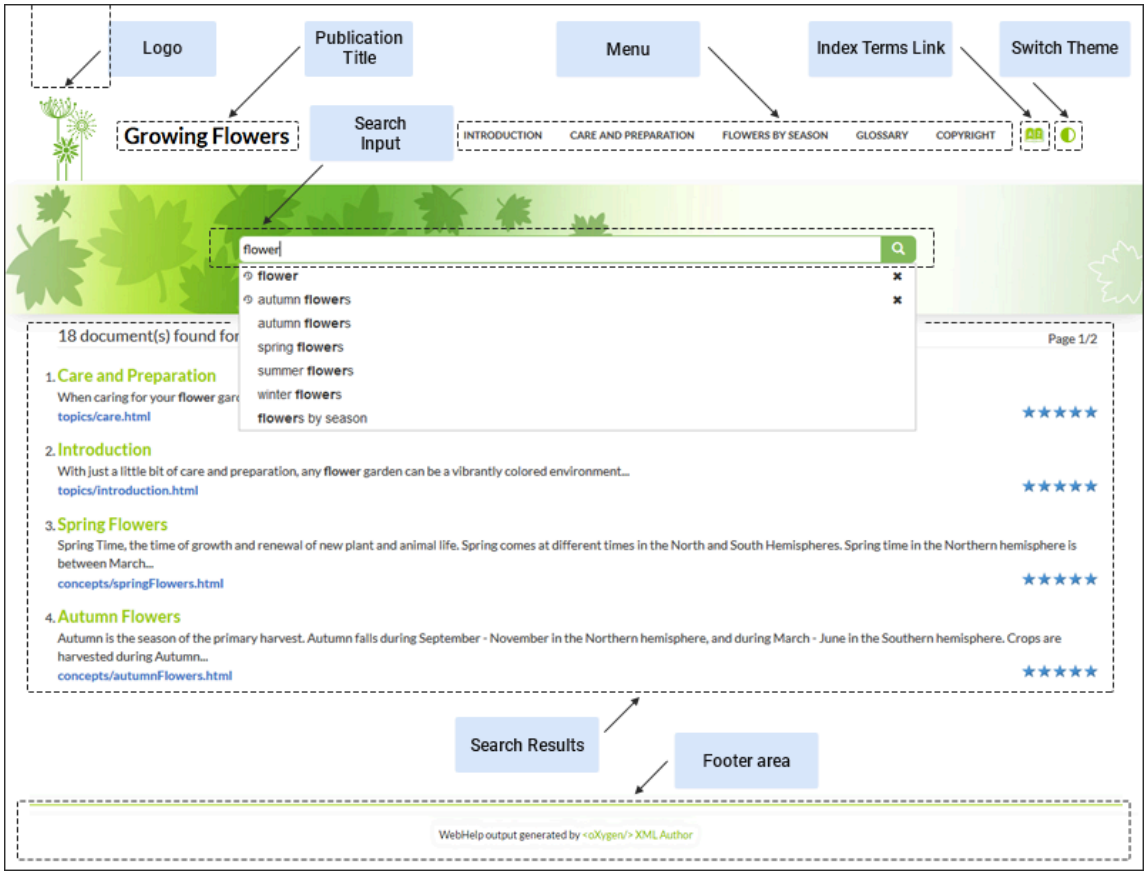
WebHelp Responsive output footer.

Search Results 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 388. Search Results Page



1. Logo Component *(on page 1230)*
2. Title Component *(on page 1230)*
3. Search Input Component *(on page 1231)*
4. Menu Component *(on page 1231)*
5. Index Terms Link Component *(on page 1231)*
6. Theme Button Component *(on page 1231)*
7. Search Results Component *(on page 1231)*
8. Footer Component *(on page 1231)*

Search Results 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 1421)*. For the target URL, use the webhelp.logo.image.target.url parameter *(on page 1421)*.

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 1344)* 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 1431)*.

Switch Theme Button

Presents a button that opens a drop-down menu for selecting the theme. You can control if this component is displayed by using the webhelp.enable.dark.mode parameter *(on page 1431)*. For more information, see How to Create a Dark Mode-Ready Publishing Template *(on page 1312)*.

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 Input Field

When you are typing in the search input 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:

```
<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

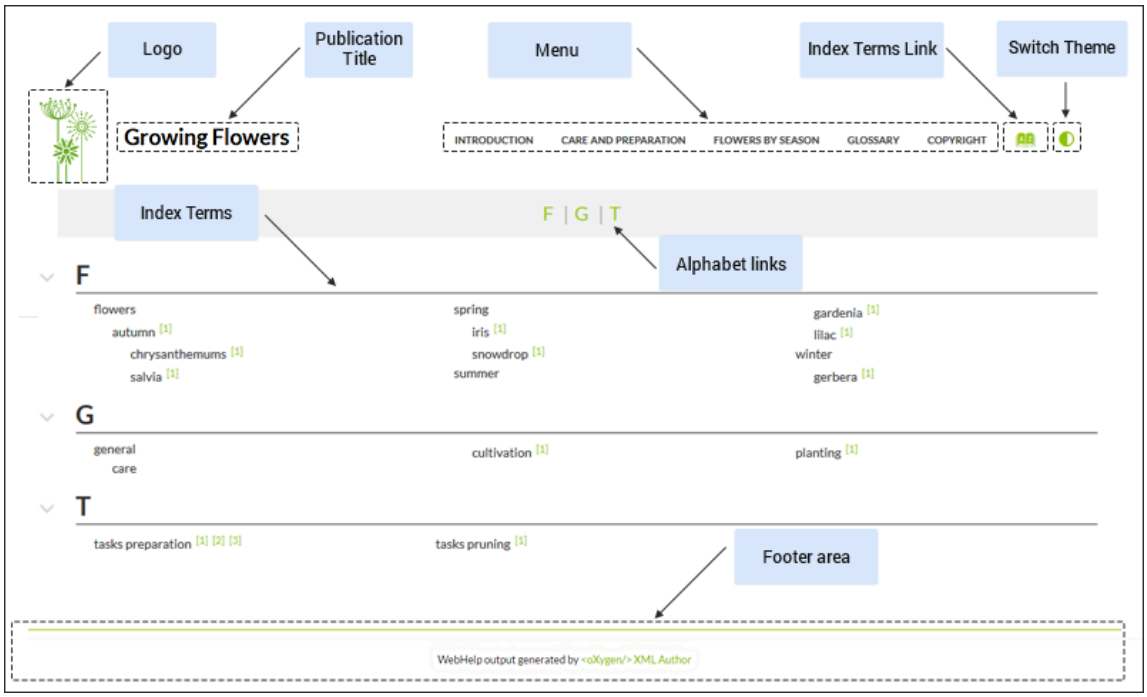
WebHelp Responsive Search Engine *(on page 1233)*

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 389. Index Terms Page



1. Logo Component *(on page 1233)*
2. Title Component *(on page 1233)*
3. Menu Component *(on page 1233)*
4. Index Terms Link Component *(on page 1233)*
5. Theme Button Component *(on page 1233)*
6. Index Terms Component *(on page 1233)*

7. Alphabet Links Component *(on page 1233)*
8. Footer Component *(on page 1233)*

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 *(on page 1421)*. For the target URL, use the webhelp.logo.image.target.url parameter *(on page 1421)*.

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 1344)* 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 1431)*.

Switch Theme Button

Presents a button that opens a drop-down menu for selecting the theme. You can control if this component is displayed by using the webhelp.enable.dark.mode parameter *(on page 1431)*. For more information, see How to Create a Dark Mode-Ready Publishing Template *(on page 1312)*.

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.

Built-in JavaScript-based Search Engine

The client-side JavaScript-based search engine comes preconfigured in the WebHelp responsive plugin. It is enabled by default when you run the WebHelp Responsive transformation.

Search Index

The search index is created when you publish your documentation to WebHelp by traverses all HTML pages (for DITA topics) from output folder 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.



Tip:

You can use the `searchQuery` URL parameter to perform a search operation when WebHelp is loaded. This opens the Search Results page with the specified search query processed. The URL should look something like this:

```
http://localhost/webhelp/search.html?searchQuery=deploying%20feedback
```

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 in the title of the page.
  - The search term is included in a meta keyword.
  - 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 1355)*.

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 multiple words in CJK (Chinese, Japanese, Korean) languages that often have them appear in strings without a space separator, you may 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.

  Note:
  Phrase searches (two or more consecutive words in an exact order) do not work for CJK (Chinese, Japanese, Korean) languages.

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 1229)*

Context-Sensitive Help System

Context-sensitive help systems assist users by offering relevant information about specific elements of a user interface, such as buttons or windows. This mechanism relies on a mapping between a unique ID and its associated HTML page.

Context-Sensitive Queries

Context-sensitive help is provided through the `cshelp.html` page:

```
http://localhost/webhelp/cshelp.html?contextId=topicID&appname=myApplication
```

Notes:

- The `contextId` parameter is not case-sensitive.
- The URL can include an *anchor (on page 2303)* to jump to a page section (e.g. `contextId=topicID#anchor`).

Attention:
Prior to version 24.1, context-sensitive help was accessible via `index.html`. This method still works but is deprecated and will be removed in a future release.

Use the following parameters in the URL:

contextId

Used by the WebHelp JavaScript engine to open the corresponding help page. Its value comes from either:

`<resourceid>` **(see DITA Specifications: <resourceid> for more details)**

Specified in the *DITA map* `<topicref>` or in the topic `<prolog>` using the following attributes (ordered by priority):

1. `@appid` - ID used by an application to identify the topic.
2. `@id` - ID used by a specific application to identify the topic, ignored if `@appid` is present.

Note:
Multiple `@appid` values can share a single `@appname`, and vice versa. However, each `@appid`/`@appname` combination must be unique within the context of a single *root map (on page 2309)*, as it identifies a specific application

ID. For example, two different functions of an application can point to 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>`.

```
<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.

```
<task id="app-help1">
  <title>My App Help</title>
  <prolog>
    <resourceid appname="myapp" appid="functionid1"/>
    <resourceid appname="myapp" appid="functionid2"/>
  </prolog>
...
</task>
```

`@id`

Specified in the topic root element, used if no `<resourceid>` is declared (in both *DITA map* and topic).

Important:
Ensure these IDs are unique within the entire DITA project. Duplicate IDs will trigger warning messages and the help system will not work properly.

Note:
The `webhelp.csh.disable.topicID.fallback` parameter *(on page 1421)* can be set to *true* to disable the topic `@id` fallback when `resourceid` information is unavailable when computing context-sensitive help mapping.

appname (optional)

Used in combination with `contextId` to match the corresponding *appname*. Its value comes from the `@appname` attribute of the `<resourceid>` element and is empty (`""`) if unspecified.

Generating Context-Sensitive WebHelp

To simplify the integration of WebHelp Responsive into an application, the `context-help-map.xml` file is generated in the output folder. This file maps IDs to HTML pages using `<appContext>` elements, as shown below:

```
<map productID="oxy-webhelp" productVersion="1.1">
  <appContext helpID="myapp-functionid1" path="tasks/app-help1.html"/>
  <appContext appname="myapp" helpID="myapp-functionid2" path="tasks/app-help1.html"/>
  ...
</map>
```

The `<appContext>` element contains the following attributes:

`@helpID`

A Unique ID that identifies the topic based on the `<resourceid>` element or the `@id` attribute (as described above).

`@appname` **(optional)**

A name for the external application that references the topic from the `@appname` attribute of the `<resourceid>` element (as described above).

`@path`

The path to the corresponding HTML page, relative to the location of the `context-help-map.xml` mapping file.

Note:
To prevent duplicate entries in `context-help-map.xml`, ensure that all repeated topics are assigned a unique `<resourceid>` in the DITA map.

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.

```
<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 `<longdescref>` element.

```
<image href="puffin.jpg">
  <alt>Puffin figure</alt>
  <longdescref 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.

```
<imagemap id="gear_pump_map">
  <image href="../images/Gear_pump_exploded.png" id="gear_pump_exploded">
    <alt>Gear Pump</alt>
```

```
  </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>circle</shape>
      <coords>324, 210, 14</coords>
      <xref href="parts/ports.dita#ports_topic/sucction_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 36. Example: Oxygen Events

Event	Date	Location
Evolution of TC 2018	May 31 - June 1, 2018	Sofia, Bulgaria
Markup UK	June 9 - 10, 2018	London, United Kingdom
Balisage 2018 - The Markup Conference	July 31 - August 3, 2018	Rockville, Maryland, USA

```
<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 37. Example: Oxygen Events

Event	Evolution of TC 2018	Markup UK	Balisage 2018 - The Markup Conference
Date	May 31 - June 1, 2018	June 9 - 10, 2018	July 31 - August 3, 2018

Table 37. Example: Oxygen Events (continued)

Location	Sofia, Bulgaria	London, United Kingdom	Rockville, Maryland, USA

```
<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 38. Example: Bus Timetable

	Mon-day	Tues-day	Wednes-day	Thurs-day	Friday
09:00 - 11:00	Closed	Open	Open	Closed	Closed
11:00 - 13:00	Open	Open	Closed	Closed	Closed
13:00 - 15:00	Open	Open	Open	Closed	Closed
15:00 - 17:00	Closed	Closed	Closed	Open	Open

```
<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"/>
     <thead>
       <row>
        <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>
      <row>
        <entry>11:00 - 13:00</entry>
        <entry>Open</entry>
        <entry>Open</entry>
```

```
        <entry>Closed</entry>
        <entry>Closed</entry>
        <entry>Closed</entry>
      </row>
      <row>
        <entry>13:00 - 15:00</entry>
        <entry>Open</entry>
        <entry>Open</entry>
        <entry>Open</entry>
        <entry>Closed</entry>
        <entry>Closed</entry>
      </row>
      <row>
        <entry>15:00 - 17:00</entry>
        <entry>Closed</entry>
        <entry>Closed</entry>
        <entry>Closed</entry>
        <entry>Open</entry>
        <entry>Open</entry>
      </row>
    </tbody>
  </tgroup>
</table>
```

WebHelp Responsive VPAT Accessibility Conformance Report

International Edition

VPAT® Version 2.5 – November 2023

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

February 2025

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 1238)*.

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:

Standard/Guideline	Included In Report
Web Content Accessibility Guidelines 2.0	Level A - Yes Level AA - Yes Level AAA - No
Web Content Accessibility Guidelines 2.1	Level A - Yes Level AA - Yes Level AAA - No
Web Content Accessibility Guidelines 2.2	Level A - Yes Level AA - Yes Level AAA - No
Revised Section 508 standards published January 18, 2017 and corrected January 22, 2018	Yes
EN 301 549 Accessibility requirements for ICT products and services - V3.1.1 (2019-11) AND EN 301 549 Accessibility requirements for ICT products and services - V3.2.1 (2021-03)	No

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 only be used in WCAG Level AAA criteria.

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 theWCAG 2.0 Conformance Requirements.

Table 1: Success Criteria, Level A

Criteria	Conformance Level	Remarks and Explanations
1.1.1 Non-text Content (Level A) Also applies to: Revised Section 508 • 501 (Web)(Software) • 504.2 (Authoring Tool) • 602.3 (Support Docs)	Partially Supports	Text alternatives are provided for many instances of non-text content, with a few exceptions.
1.2.1 Audio-only and Video-only (Prerecorded) (Level A) Also applies to: Revised Section 508 • 501 (Web)(Software) • 504.2 (Authoring Tool) • 602.3 (Support Docs)	Supports	The authors of the input DITA document are responsible for providing a transcript of the media content in the document.
1.2.2 Captions (Prerecorded) (Level A) Also applies to:	Supports	The product does not provide prerecorded media that requires captions.

Criteria	Conformance Level	Remarks and Explanations
Revised Section 508 • 501 (Web)(Software) • 504.2 (Authoring Tool) • 602.3 (Support Docs)		
1.2.3 Audio Description or Media Alternative (Prerecorded) (Level A) Also applies to: Revised Section 508 • 501 (Web)(Software) • 504.2 (Authoring Tool) • 602.3 (Support Docs)	Supports	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
1.3.1 Info and Relationships (Level A) Also applies to: Revised Section 508 • 501 (Web)(Software) • 504.2 (Authoring Tool) • 602.3 (Support Docs)	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:

Criteria	Conformance Level	Remarks and Explanations
		• 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.
1.3.2 Meaningful Sequence (Level A) Also applies to: Revised Section 508 • 501 (Web)(Software) • 504.2 (Authoring Tool) • 602.3 (Support Docs)	Supports	The product presents content in a meaningful sequence. Authors should use Unicode right-to-left mark (RLM) or left-to-right mark (LRM) to mix text direction inline.
1.3.3 Sensory Characteristics (Level A) Also applies to: Revised Section 508 • 501 (Web)(Software) • 504.2 (Authoring Tool) • 602.3 (Support Docs)	Supports	Authors should ensure that items are referenced in the content in ways that do not depend on sensory perception.
1.4.1 Use of Color (Level A) Also applies to: Revised Section 508 • 501 (Web)(Software) • 504.2 (Authoring Tool) • 602.3 (Support Docs)	Supports	(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.
1.4.2 Audio Control (Level A) Also applies to:	Supports	There is no sound that plays automatically.

Criteria	Conformance Level	Remarks and Explanations
Revised Section 508 • 501 (Web)(Software) • 504.2 (Authoring Tool) • 602.3 (Support Docs)		
2.1.1 Keyboard (Level A) Also applies to: Revised Section 508 • 501 (Web)(Software) • 504.2 (Authoring Tool) • 602.3 (Support Docs)	Partially Supports	Most of the content is operable through a keyboard interface, with exceptions that include: • The top-level links in the main page accordion cannot be accessed. • The facets component does not have full keyboard support.
2.1.2 No Keyboard Trap (Level A) Also applies to: Revised Section 508 • 501 (Web)(Software) • 504.2 (Authoring Tool) • 602.3 (Support Docs)	Supports	The product does not contain content that traps the keyboard focus.
2.1.4 Character Key Shortcuts (Level A 2.1 only) Also applies to: Revised Section 508 – Does not apply	Supports	The product does not include character key shortcuts.
2.2.1 Timing Adjustable (Level A) Also applies to: Revised Section 508	Supports	The product does not include time limits.

Criteria	Conformance Level	Remarks and Explanations
• 501 (Web)(Software) • 504.2 (Authoring Tool) • 602.3 (Support Docs)		
2.2.2 Pause, Stop, Hide (Level A) Also applies to: Revised Section 508 • 501 (Web)(Software) • 504.2 (Authoring Tool) • 602.3 (Support Docs)	Supports	The product does not include elements that move, blink, scroll, or auto-update.
2.3.1 Three Flashes or Below Threshold (Level A) Also applies to: Revised Section 508 • 501 (Web)(Software) • 504.2 (Authoring Tool) • 602.3 (Support Docs)	Supports	The product does not contain flashing content.
2.4.1 Bypass Blocks (Level A) 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	Supports	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.
2.4.2 Page Titled (Level A) Also applies to: Revised Section 508	Supports	Each page contains a non-empty `<title>` element in the `<head>` section.

Criteria	Conformance Level	Remarks and Explanations
• 501 (Web)(Software) • 504.2 (Authoring Tool) • 602.3 (Support Docs)		
2.4.3 Focus Order (Level A) Also applies to: Revised Section 508 • 501 (Web)(Software) • 504.2 (Authoring Tool) • 602.3 (Support Docs)	Supports	Focusable components receive focus in an order that preserves meaning and operability.
2.4.4 Link Purpose (In Context) (Level A) Also applies to: Revised Section 508 • 501 (Web)(Software) • 504.2 (Authoring Tool) • 602.3 (Support Docs)	Supports	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. 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).
2.5.1 Pointer Gestures (Level A 2.1 only) Also applies to: Revised Section 508 – Does not apply	Supports	The WebHelp Responsive output does not rely on path-based or multipoint gestures and does not provide controls that require complex gestures.
2.5.2 Pointer Cancellation (Level A 2.1 only) Also applies to: Revised Section 508 – Does not apply	Supports	The product has operations that are activated on the pointer up event.
2.5.3 Label in Name (Level A 2.1 only)	Supports	The names of the user interface components contain the text that is presented visually.

Criteria	Conformance Level	Remarks and Explanations
Also applies to: Revised Section 508 – Does not apply		
2.5.4 Motion Actuation (Level A 2.1 only) Also applies to: Revised Section 508 – Does not apply	Supports	The product does not contain functionality that can be operated by device or user motion.
3.1.1 Language of Page (Level A) Also applies to: Revised Section 508 • 501 (Web)(Software) • 504.2 (Authoring Tool) • 602.3 (Support Docs)	Supports	The web pages indicate the language of the content when the content language has been specified by authors.
3.2.1 On Focus (Level A) Also applies to: Revised Section 508 • 501 (Web)(Software) • 504.2 (Authoring Tool) • 602.3 (Support Docs)	Supports	No changes of context occur when any component receives focus.
3.2.2 On Input (Level A) Also applies to: Revised Section 508 • 501 (Web)(Software) • 504.2 (Authoring Tool) • 602.3 (Support Docs)	Supports	Changing the setting of any user interface component does not automatically cause a change of context.
3.3.1 Error Identification (Level A) Also applies to:	Partially Supports	If a search operation is performed leaving the search input empty, an error message is automatically dis-

Criteria	Conformance Level	Remarks and Explanations
Revised Section 508 • 501 (Web)(Software) • 504.2 (Authoring Tool) • 602.3 (Support Docs)		played to the user, but no *aria-invalid* information is provided.
3.3.2 Labels or Instructions (Level A) Also applies to: Revised Section 508 • 501 (Web)(Software) • 504.2 (Authoring Tool) • 602.3 (Support Docs)	Partially Supports	The search input does not have a visible label specified using a label element.
4.1.1 Parsing (Level A) Also applies to: Revised Section 508 • 501 (Web)(Software) • 504.2 (Authoring Tool) • 602.3 (Support Docs)	Partially Supports	Several HTML validation errors are reported by the W3C validator.
4.1.2 Name, Role, Value (Level A) Also applies to: Revised Section 508 • 501 (Web)(Software) • 504.2 (Authoring Tool) • 602.3 (Support Docs)	Partially Supports	The *Home* link from the breadcrumb does not have an associated *aria-label*.

Table 2: Success Criteria, Level AA

Criteria	Conformance Level	Remarks and Explanations
1.2.4 Captions (Live) (Level AA)	Supports	No live audio content is used.

Criteria	Conformance Level	Remarks and Explanations
Also applies to: Revised Section 508 • 501 (Web)(Software) • 504.2 (Authoring Tool) • 602.3 (Support Docs)		
1.2.5 Audio Description (Prerecorded) (Level AA) Also applies to: Revised Section 508 • 501 (Web)(Software) • 504.2 (Authoring Tool) • 602.3 (Support Docs)	Supports	The authors of the input DITA document can ensure that the output document meets this criterion.
1.3.4 Orientation (Level AA 2.1 and 2.2) Also applies to: Revised Section 508 – Does not apply	Supports	Content does not restrict its view and operation to a single display orientation.
1.3.5 Identify Input Purpose (Level AA 2.1 and 2.2) Also applies to: Revised Section 508 – Does not apply	Supports	The content does not contain input fields that collect information about the user.
1.4.3 Contrast (Minimum) (Level AA) Also applies to: Revised Section 508 • 501 (Web)(Software) • 504.2 (Authoring Tool) • 602.3 (Support Docs)	Partially Supports	The missing words element from the search results page does not have the contrast ratio 4.5:1.

Criteria	Conformance Level	Remarks and Explanations
1.4.4 Resize text (Level AA) Also applies to: Revised Section 508 • 501 (Web)(Software) • 504.2 (Authoring Tool) • 602.3 (Support Docs)	Partially Supports	Text can be resized up to 200 percent without loss of content or functionality and without using assistive technology. Some text content has dimensions specified in pixels rather that em units.
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)	Supports	The output does not contain images of text. The authors of the input DITA content can ensure that this criterion is met.
1.4.10 Reflow (Level AA 2.1 and 2.2) Also applies to: Revised Section 508 – Does not apply	Partially Supports	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.
1.4.11 Non-text Contrast (Level AA 2.1 and 2.2) Also applies to: Revised Section 508 – Does not apply	Supports	(Cobalt template) There is no contrast issue regarding user interface components or graphical objects.
1.4.12 Text Spacing (Level AA 2.1 and 2.2) Also applies to: Revised Section 508 – Does not apply	Supports	There is no loss of content or functionality that occurs by setting line height (line spacing), spacing following paragraphs, letter spacing, and word spacing.

Criteria	Conformance Level	Remarks and Explanations
1.4.13 Content on Hover or Focus (Level AA 2.1 and 2.2) Also applies to: Revised Section 508 – Does not apply	Partially Supports	Tooltips and submenus are not dismissible. Also, the tooltips are not hoverable.
2.4.5 Multiple Ways (Level AA) 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	Supports	There is a search form provided that will go to a page that contains the search term and links to the corresponding page. Also, a table of contents is provided. The authors of the input DITA document are responsible for providing links to all pages from the home page or providing links to navigate to related pages from the current page.
2.4.6 Headings and Labels (Level AA) Also applies to: Revised Section 508 • 501 (Web)(Software) • 504.2 (Authoring Tool) • 602.3 (Support Docs)	Supports	Headings and labels describe the topic or purpose. DITA authors can ensure that this criterion is met.
2.4.7 Focus Visible (Level AA) Also applies to: Revised Section 508 • 501 (Web)(Software) • 504.2 (Authoring Tool) • 602.3 (Support Docs)	Partially Supports	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.
2.4.11 Focus Not Obscured (Minimum) (Level AA 2.2 only)	Supports	Focused items are always visible even if the page contains sticky elements.

Criteria	Conformance Level	Remarks and Explanations
Revised Section 508 – Does not apply		
2.5.7 Dragging Movements (Level AA 2.2 only) Revised Section 508 – Does not apply	Supports	No user action relies on dragging.
2.5.8 Target Size (Minimum) (Level AA 2.2 only) Revised Section 508 – Does not apply	Does Not Support	Links in the breadcrumb, publication TOC, and topic TOC do not match the 24x24px minimum size. Navigation buttons, print button, and link to PDF button do not match the 24x24px minimun size.
3.1.2 Language of Parts (Level AA) Also applies to: Revised Section 508 • 501 (Web)(Software) • 504.2 (Authoring Tool) • 602.3 (Support Docs)	Supports	DITA authors can ensure that this criterion is met.
3.2.3 Consistent Navigation (Level AA) 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	Supports	Repeated components appear in the same relative in each page.
3.2.4 Consistent Identification (Level AA) Also applies to: Revised Section 508	Partially Supports	The output uses labels, names, and text alternatives consistently for items that have the same functionality.

Criteria	Conformance Level	Remarks and Explanations
• 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		Text alternatives are provided for many instances of non-text content, with exceptions that include: • Enlarge images action. The *Home* link from the breadcrumb does not have an associated *aria-label*.
3.3.3 Error Suggestion (Level AA) Also applies to: Revised Section 508 • 501 (Web)(Software) • 504.2 (Authoring Tool) • 602.3 (Support Docs)	Does Not Support	The Search input does not have the *aria-required* information set and does not contain a text description specifying that it is a required field.
3.3.4 Error Prevention (Legal, Financial, Data) (Level AA) Also applies to: Revised Section 508 • 501 (Web)(Software) • 504.2 (Authoring Tool) • 602.3 (Support Docs)	Supports	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.
3.3.8 Accessible Authentication (Minimum) (Level AA 2.2 only) Revised Section 508 – Does not apply	Not Applicable	
4.1.3 Status Messages(Level AA 2.1 and 2.2) Also applies to: Revised Section 508 – Does not apply	Supports	The pages do not contain status messages as defined by this criterion.

Table 3: Success Criteria, Level AAA

Criteria	Conformance Level	Remarks and Explanations
1.2.6 Sign Language (Prerecorded) (Level AAA) Revised Section 508 – Does not apply	Not Evaluated	
1.2.7 Extended Audio Description (Prerecorded) (Level AAA) Revised Section 508 – Does not apply	Not Evaluated	
1.2.8 Media Alternative (Prerecorded) (Level AAA) Revised Section 508 – Does not apply	Not Evaluated	
1.2.9 Audio-only (Live) (Level AAA) Revised Section 508 – Does not apply	Not Evaluated	
1.3.6 Identify Purpose (Level AAA 2.1 and 2.2) Revised Section 508 – Does not apply	Not Evaluated	
1.4.6 Contrast Enhanced (Level AAA) Revised Section 508 – Does not apply	Not Evaluated	
1.4.7 Low or No Background Audio (Level AAA) Revised Section 508 – Does not apply	Not Evaluated	
1.4.8 Visual Presentation (Level AAA) Revised Section 508 – Does not apply	Not Evaluated	
1.4.9 Images of Text (No Exception) Control (Level AAA)	Not Evaluated	

Criteria	Conformance Level	Remarks and Explanations
Revised Section 508 – Does not apply		
2.1.3 Keyboard (No Exception) (Level AAA) Revised Section 508 – Does not apply	Not Evaluated	
2.2.3 No Timing (Level AAA) Revised Section 508 – Does not apply	Not Evaluated	
2.2.4 Interruptions (Level AAA) Revised Section 508 – Does not apply	Not Evaluated	
2.2.5 Re-authenticating (Level AAA) Revised Section 508 – Does not apply	Not Evaluated	
2.2.6 Timeouts (Level AAA 2.1 and 2.2) Revised Section 508 – Does not apply	Not Evaluated	
2.3.2 Three Flashes (Level AAA) Revised Section 508 – Does not apply	Not Evaluated	
2.3.3 Animation from Interactions (Level AAA 2.1 and 2.2) Revised Section 508 – Does not apply	Not Evaluated	
2.4.8 Location (Level AAA) Revised Section 508 – Does not apply	Not Evaluated	
2.4.9 Link Purpose (Link Only) (Level AAA) Revised Section 508 – Does not apply	Not Evaluated	
2.4.10 Section Headings (Level AAA)	Not Evaluated	

Criteria	Conformance Level	Remarks and Explanations
Revised Section 508 – Does not apply		
2.4.12 Focus Not Obscured (Enhanced) (Level AAA 2.2 only) Revised Section 508 – Does not apply	Not Evaluated	
2.4.13 Focus Appearance (Level AAA 2.2 only) Revised Section 508 – Does not apply	Not Evaluated	
2.5.5 Target Size (Level AAA 2.1 and 2.2) Revised Section 508 – Does not apply	Not Evaluated	
2.5.6 Concurrent Input Mechanisms (Level AAA 2.1 and 2.2) Revised Section 508 – Does not apply	Not Evaluated	
3.1.3 Unusual Words (Level AAA) Revised Section 508 – Does not apply	Not Evaluated	
3.1.4 Abbreviations (Level AAA) Revised Section 508 – Does not apply	Not Evaluated	
3.1.5 Reading Level (Level AAA) Revised Section 508 – Does not apply	Not Evaluated	
3.1.6 Pronunciation (Level AAA) Revised Section 508 – Does not apply	Not Evaluated	
3.2.5 Change on Request (Level AAA) Revised Section 508 – Does not apply	Not Evaluated	
3.3.5 Help (Level AAA)	Not Evaluated	

Criteria	Conformance Level	Remarks and Explanations
Revised Section 508 – Does not apply		
3.3.6 Error Prevention (All) (Level AAA) Revised Section 508 – Does not apply	Not Evaluated	
3.3.9 Accessible Authentication (Enhanced) (Level AAA 2.2 only) Revised Section 508 – Does not apply	Not Evaluated	

Revised Section 508 Report

N/A

Chapter 3: Functional Performance Criteria (FPC)

Criteria	Conformance Level	Remarks and Explanations
302.1 Without Vision	Partially Supports	Most of the content is accessible without vision with exceptions that include: • Some components do not have text alternatives or labels. • 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.
302.2 With Limited Vision	Partially Supports	Most of the content is accessible with limited vision with exceptions that include: • Some components do not have text alternatives or labels. • Some landmarks are not marked with the corresponding

Criteria	Conformance Level	Remarks and Explanations
		role or do not have an associated label. • Some link groups are not structured using lists or are not marked as navigation regions.
302.3 Without Perception of Color	Supports	(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.
302.4 Without Hearing	Supports	The authors can create content that does not require hearing abilities for use.
302.5 With Limited Hearing	Supports	The authors can create content that does not require hearing abilities for use.
302.6 Without Speech	Supports	The output does not require speech for use.
302.7 With Limited Manipulation	Supports	The WebHelp Responsive output does not rely on path-based or multipoint gestures and does not provide controls that require complex gestures.
302.8 With Limited Reach and Strength	Supports	The WebHelp Responsive output does not rely on path-based or multipoint gestures and does not provide controls that require complex gestures.
302.9 With Limited Language, Cognitive, and Learning Abilities	Supports	The authors can create content that can be used by users with limited language, cognitive, and learning abilities.

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

Criteria	Conformance Level	Remarks and Explanations
501.1 Scope – Incorporation of WCAG 2.0 AA	See WCAG 2.x section *(on page 1246)*	See information in WCAG section

502 Interoperability with Assistive Technology

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

502.3 Accessibility Services

Criteria	Conformance Level	Remarks and Explanations
502.3.1 Object Information	Partially Supports	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*.
502.3.2 Modification of Object Information	Supports	States and properties that can be set by the user can be set programmatically.
502.3.3 Row, Column, and Headers	Supports	The headers associated with the rows or columns of a table can be programmatically determined.
502.3.4 Values	Supports	The current values of an object can be programmatically determined.

Criteria	Conformance Level	Remarks and Explanations
502.3.5 Modification of Values	Supports	Values that can be set by the user are capable of being set programmatically.
502.3.6 Label Relationships	Partially Supports	Information, structure, and relationships conveyed through presentation can be programmatically determined or are available in text. See WCAG 1.3.1 *(on page 1247)*.
502.3.7 Hierarchical Relationships	Supports	The content is hierarchically structured using language-specific elements and their relationships can be programmatically determined.
502.3.8 Text	Supports	The content of text objects, text attributes, and the boundary of text rendered to the screen shall be programmatically determinable.
502.3.9 Modification of Text	Supports	The editable text (search input) can be set programmatically.
502.3.10 List of Actions	Not Applicable	There are no custom actions available that can be executed on the content.
502.3.11 Actions on Objects	Not Applicable	There are no custom actions available that can be executed on the content.
502.3.12 Focus Cursor	Not Applicable	The product is a web application and is isolated from the underlying platform software (web browser).
502.3.13 Modification of Focus Cursor	Not Applicable	The product is a web application and is isolated from the underlying platform software (web browser).
502.3.14 Event Notification	Not Applicable	There are no automatic focus changes, caret movement, selection changes, or added components in the content.

Criteria	Conformance Level	Remarks and Explanations
502.4 Platform Accessibility Features	Not Applicable	This product is not platform software.

503 Applications

Criteria	Conformance Level	Remarks and Explanations
503.2 User Preferences	Not Applicable	This section does not apply to web applications.
503.3 Alternative User Interfaces	Not Applicable	The application does not provide an alternative user interface that functions as assistive technology.

503.4 User Controls for Captions and Audio Description

Criteria	Conformance Level	Remarks and Explanations
503.4.1 Caption Controls	Not Applicable	The product does not provide controls for volume adjustment.
503.4.2 Audio Description Controls	Not Applicable	The product does not provide controls for program selection.

504 Authoring Tools

Criteria	Conformance Level	Remarks and Explanations
504.2 Content Creation or Editing (if not authoring tool, enter “not applicable”)	Not Applicable See the WCAG 2.x section *(on page 1246)*	The product is not an authoring tool. See information in WCAG section
504.2.1 Preservation of Information Provided for Accessibility in Format Conversion	Not Applicable	The product is not an authoring tool.
504.2.2 PDF Export	Not Applicable	The product is not an authoring tool.
504.3 Prompts	Not Applicable	The product is not an authoring tool.
504.4 Templates	Not Applicable	The product is not an authoring tool.

Chapter 6: Support Documentation and Services

601.1 Scope

602 Support Documentation

Criteria	Conformance Level	Remarks and Explanations
602.2 Accessibility and Compatibility Features	Partially Supports	The product documentation is distributed in the WebHelp Responsive format. See the Chapter 3 *(on page 1262)* and Chapter 5 *(on page 1263)* sections.
602.3 Electronic Support Documentation	See the WCAG 2.x section *(on page 1246)*	See information in the WCAG section.
602.4 Alternate Formats for Non-Electronic Support Documentation	Not Applicable	Documentation is not provided in non-electronic formats.

603 Support Services

Criteria	Conformance Level	Remarks and Explanations
603.2 Information on Accessibility and Compatibility Features	Supports	The support services cover the accessibility features.
603.3 Accommodation of Communication Needs	Supports	Support services are available by phone or e-mail.

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.

Tip:
You can start creating publishing templates by using the **Oxygen Styles Basket**. https://styles.oxygenxml.com

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 390. Oxygen Publishing Template Package (WebHelp Responsive)

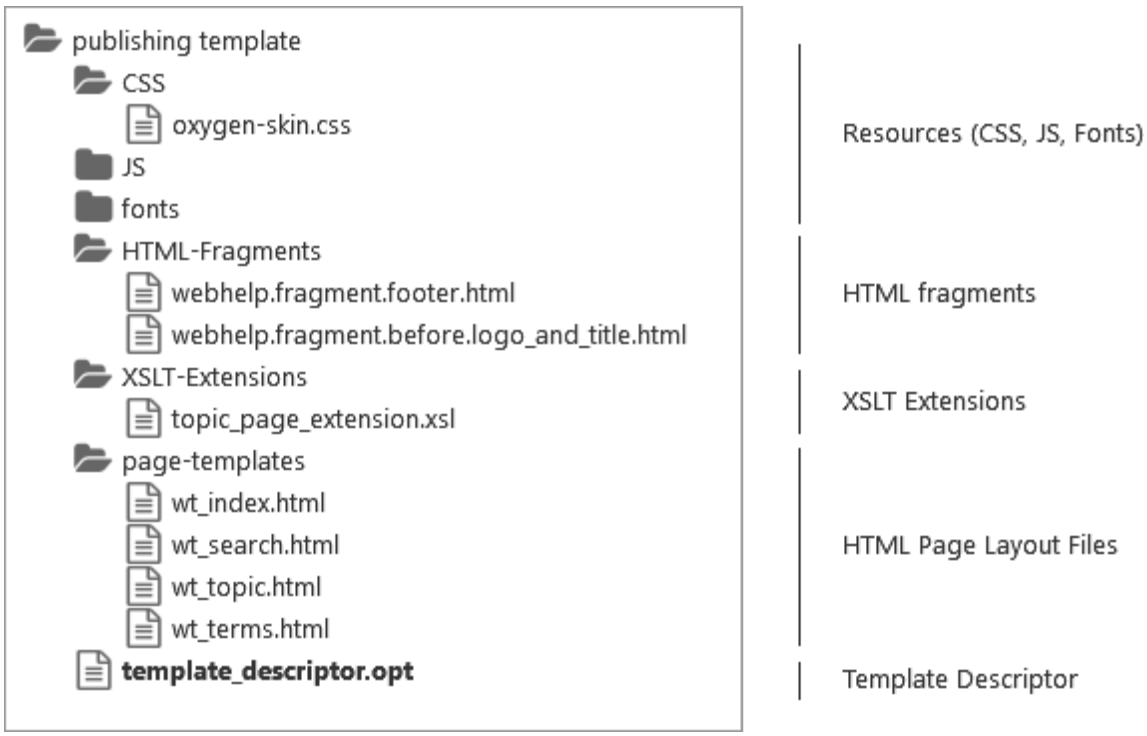


Figure 391. Oxygen Publishing Template Package (PDF)



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 1309)*
How to Edit a Packed Publishing Template *(on page 1324)*
How to Add a Publishing Template to the Publishing Templates Gallery *(on page 1324)*
How to Share a Publishing Template *(on page 1469)*

Publishing Templates Gallery

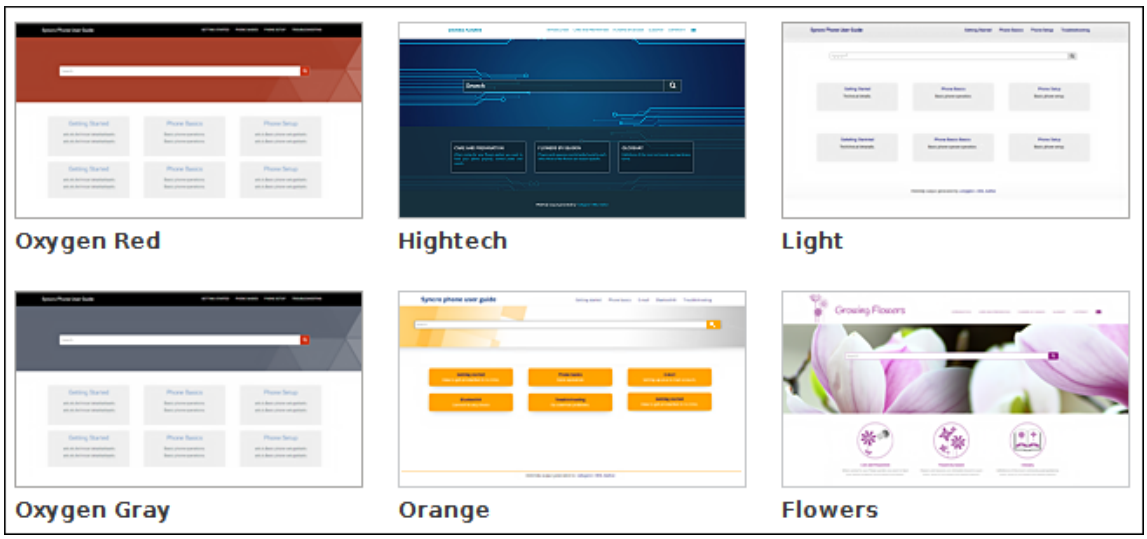
Oxygen XML Developer comes bundled with a variety of built-in templates. You can use one of them to publish your documentation or as a starting point for a new publishing template.

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.

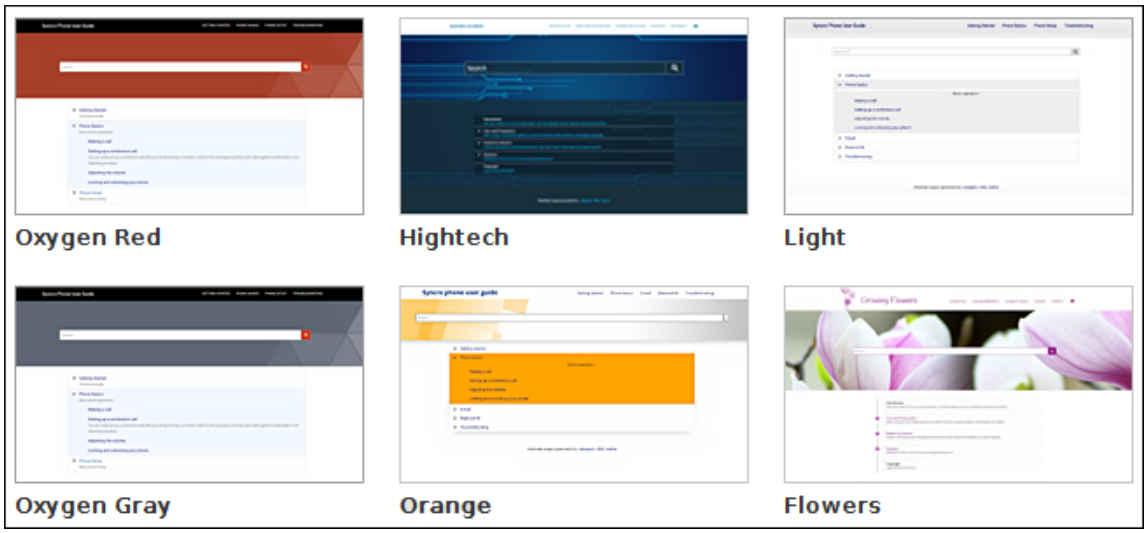
Tiles Templates

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 1464)*. 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:

```
<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>
            <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.

```
<publishing-template>
    <name>Lorem Ipsum</name>
    <description>Lorem ipsum dolor sit amet, consectetur adipiscing elit</description>
    ...
```

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.

```
<publishing-template>
    ...
    <author>
      <name>John Doe</name>
      <email>jdoe@example.com</email>
      <organization>ACME</organization>
      <organizationUrl>http://www.example.com/jdoe</organizationUrl>
```

```
    </author>
    ...
```

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 1273)*, template preview image *(on page 1274)*, resources *(on page 1274)* (such as CSS, JS, fonts, logos), transformation parameters *(on page 1276)*, HTML fragment extensions *(on page 1278)* (used to add fragments to placeholders), XSLT extensions *(on page 1277)*, or HTML page layout files *(on page 1287)*.

```
    <webhelp>
        <tags>
          ...
        </tags>
        <preview-image file="MyPreview.png"/>

        <resources>
          ...
        </resources>

        <html-page-layouts>
          ...
        </html-page-layouts>

        <parameters>
          ...
        </parameters>
    </webhelp>
```

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>
```

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.

```
<publishing-template>
    ...
    <webhelp>
      ...
      <preview-image file="ashes/ashes-tree.png"/>
      <online-preview-url>https://www.example.com/samples/tiles/ashes</online-preview-url>
```

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` - Redirects the user to the specified URL if they click the logo in the output.
    - `@new-tab` - Opens the target URL in a new browser tab.
    - `@alt` - Provides an alternate text for the logo image.
- **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.

```
<publishing-template>
    ...
    <webhelp>
      ...
      <resources>
          <css file="css/custom_styles.css"/>
          <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"
           new-tab="true"/>

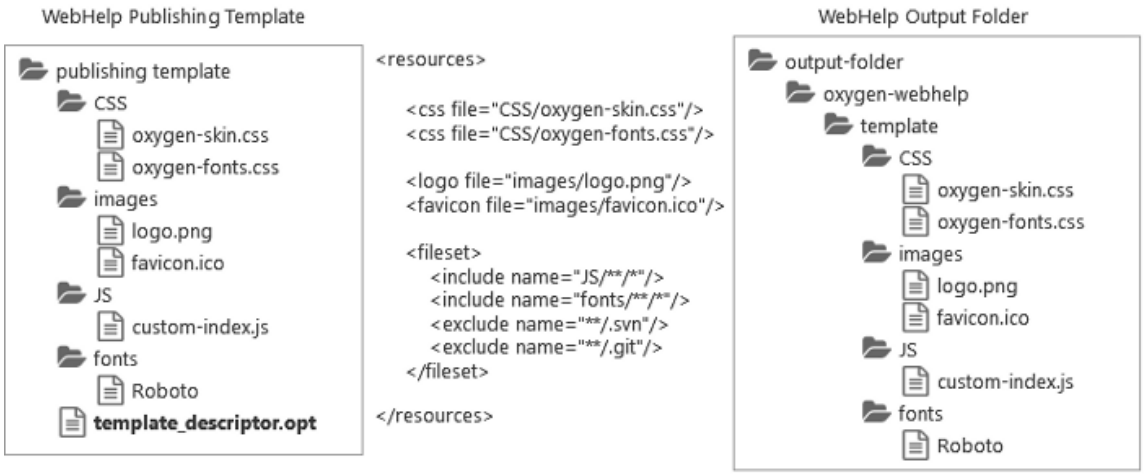
          <js-amd-module file="js/template-main.js"/>

          <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 392. Template Resources Mapping



Related Information:

How to Add a Favicon in WebHelp Systems *(on page 1352)*

Transformation Parameters

You can also set one or more WebHelp transformation parameters in the descriptor file.

```
<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>
```

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 1420)* 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 1464)* and adding it to the templates gallery *(on page 1324)*, 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 1435)*. 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>
```

For a full list of the supported extension points, see: XSLT-Import and XSLT-Parameter Extension Points *(on page 1435)*.

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 1334)*).

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>
      ...
      <html-fragments>
        <fragment
          file="html-fragments/webhelp_fragment_welcome.html"
          placeholder="webhelp.fragment.welcome"/>
        <fragment
          file="html-fragments/webhelp_fragment_footer.html"
          placeholder="webhelp.fragment.footer"/>
      </html-fragments>
```

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 393. Global Predefined Placeholders Diagram



1. Header *(on page 1279)*
2. After Header *(on page 1279)*
3. Before Body *(on page 1279)*
4. Before Logo and Title *(on page 1279)*
5. After Logo and Title *(on page 1280)*
6. Beforer Top Menu *(on page 1280)*
7. After Top Menu *(on page 1280)*
8. Before Search Input *(on page 1280)*
9. After Search Input *(on page 1280)*
10. Before Main Content *(on page 1280)*
11. After Main Content *(on page 1280)*
12. Footer *(on page 1280)*
13. After Body *(on page 1280)*

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:

- **1 = webhelp.fragment.header** - Displays the specified XHTML fragment in the header section.
- **2 = webhelp.fragment.after.header** - Displays the specified XHTML fragment after the header section.
- **3 = webhelp.fragment.before.body** - Displays the specified XHTML fragment before the body.
- **4 = webhelp.fragment.before.logo_and_title** - Displays the specified XHTML fragment before the logo and title.

- **5 = webhelp.fragment.after.logo_and_title** - Displays the specified XHTML fragment after the logo and title.
- **6 = webhelp.fragment.before.top_menu** - Displays the specified XHTML fragment before the top menu.
- **7 = webhelp.fragment.after.top_menu** - Displays the specified XHTML fragment after the top menu.
- **8 = webhelp.fragment.before.search.input** - Displays the specified XHTML fragment before the search input component.
- **9 = webhelp.fragment.after.search.input** - Displays the specified XHTML fragment after the search input component.
- **10 = webhelp.fragment.before.main.content.area** - Displays the specified XHTML fragment before the main content area
- **11 = webhelp.fragment.after.main.content.area** - Displays the specified XHTML fragment after the main content area.
- **12 = webhelp.fragment.footer** - Displays the specified XHTML fragment in the footer section.
- **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:

- **webhelp.fragment.header.main.page** - Displays the specified XHTML fragment in the header section.
- **webhelp.fragment.after.header.main.page** - Displays the specified XHTML fragment after the header section.
- **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.header.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.header.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.header.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.
- **webhelp.fragment.before.body.terms.page** - Displays the specified XHTML fragment before the body.
- **webhelp.fragment.before.search.input.terms.page** - Displays the specified XHTML fragment before the search input component.
- **webhelp.fragment.after.search.input.terms.page** - Displays the specified XHTML fragment after the search input component.
- **webhelp.fragment.before.main.content.area.terms.page** - Displays the specified XHTML fragment before the main content area
- **webhelp.fragment.after.main.content.area.terms.page** - Displays the specified XHTML fragment after the main content area.
- **webhelp.fragment.after.body.terms.page** - Displays the specified XHTML fragment after the body.

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 1334)*

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

```
${macro-name(macro-parameter)}
```

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.

```
${i18n(string.id)}
```

param

Returns the value of a transformation parameter.

```
${param(webhelp.show.main.page.tiles)}
```

env

Returns the value of an environment variable.

```
${env(JAVA_HOME)}
```

system-property

Returns the value of a system property.

```
${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.

```
${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.

```
${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`).

```
${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.

```
<!-- 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:choose>
        <xsl:when test="$contextNode instance of attribute()">
            <xsl:value-of select="$matchedString"/>
        </xsl:when>
        <xsl:otherwise>
            <xsl:message>Cannot expand macro:
                [<xsl:value-of select="$matchedString"/>]</xsl:message>
            <xsl:copy-of select="$contextNode"/>
        </xsl:otherwise>
    </xsl:choose>
</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.

```
<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 PDF Customizations *(on page 1459)*

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 1271)*.

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.

```
<publishing-template>
      ...
      <webhelp>
      ...
      <!-- HTML page layout files -->
      <html-page-layout-files>
        <page-layout-file page="main" file="page-templates/wt_index.html"/>
        <page-layout-file page="search" file="page-templates/wt_search.html"/>
        <page-layout-file page="topic" file="page-templates/wt_topic.html"/>
        <page-layout-file page="index-terms" file="page-templates/wt_terms.html"/>
      </html-page-layout-files>
```

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 2304)*.

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:

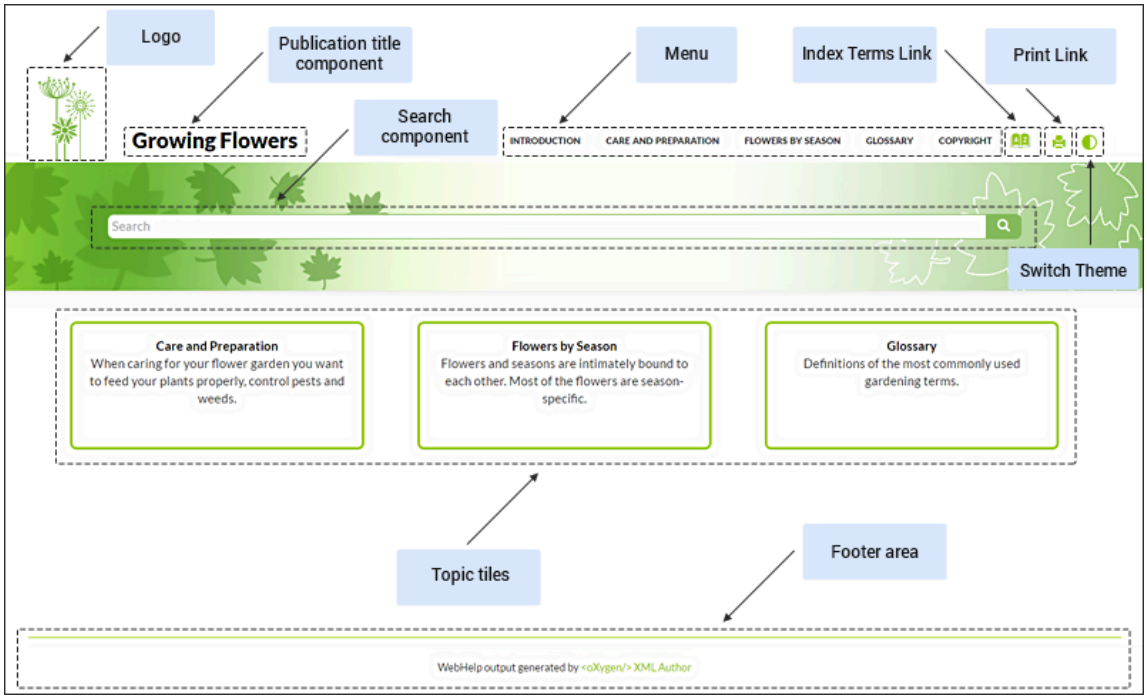
```
<whc:webhelp_search_input class="navbar-form wh_main_page_search"
    role="form" >
    <div class="custom-content-before">Enter search terms here:</div>
    <div class="custom-wrapper">
        <whc:component_content/>
    </div>
    <div class="custom-content-after">Results will be displayed in 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`.

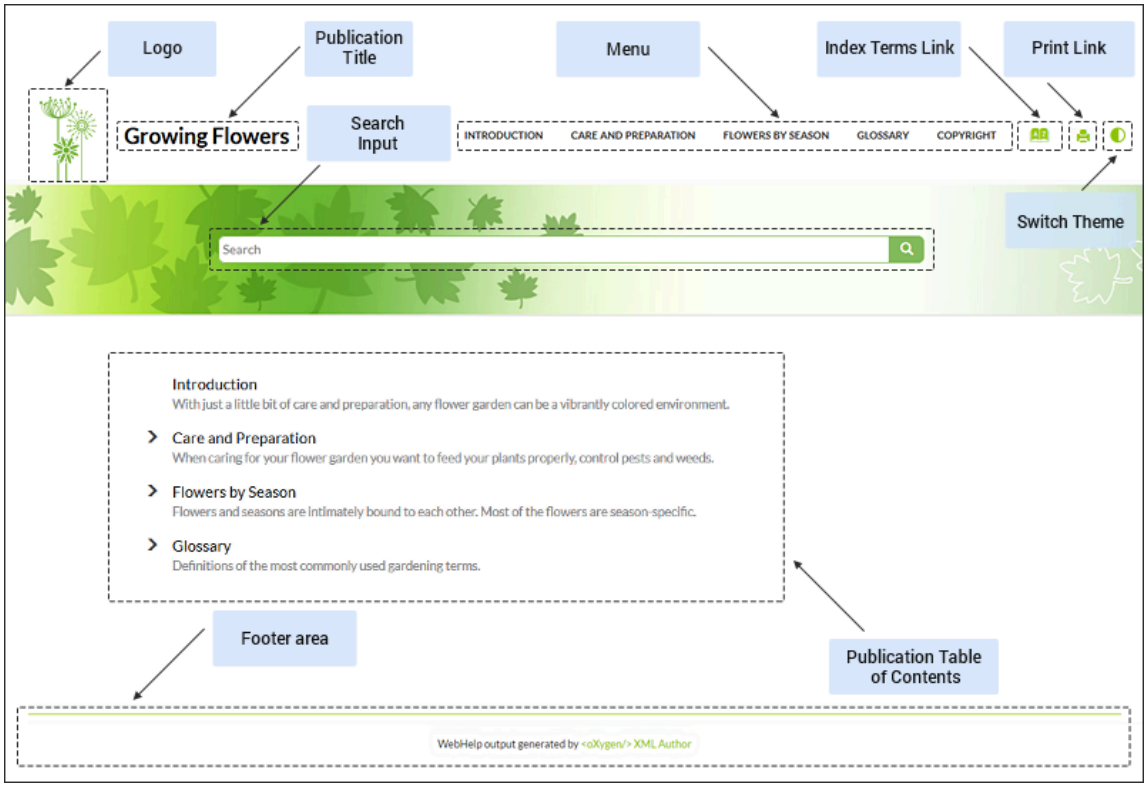
The main function of the home page is to display top-level information and provide links that help you easily navigate to any of the top-level topics of the publication. These links can be rendered in either a *Tiles* or *Tree* style of layout. The HTML page produced for the home page also consists of various other components, such as a logo, title, menu, search field, or index link.

Figure 394. Example of Main Page Components for a Tiles Style of Layout



1. Publication Logo *(on page 1290)*
2. Publication Title *(on page 1290)*
3. Search Input *(on page 1291)*
4. Main Menu *(on page 1291)*
5. Index Terms Link *(on page 1293)*
6. Print Link *(on page 1291)*
7. Switch Theme Button *(on page 1291)*
8. Topic Tiles *(on page 1292)*

Figure 395. Example of Main Page Components for a Tree Style of Layout



1. Publication Logo *(on page 1290)*
2. Publication Title *(on page 1290)*
3. Search Input *(on page 1291)*
4. Main Menu *(on page 1291)*
5. Index Terms Link *(on page 1293)*
6. Table of Contents *(on page 1292)*
7. Switch Theme Button *(on page 1291)*
8. Print Link *(on page 1291)*

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:

```
<whc:webhelp_publication_title
    xmlns:whc="http://www.oxygenxml.com/webhelp/components"/>
```

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:

```
<whc:webhelp_logo
    xmlns:whc="http://www.oxygenxml.com/webhelp/components"/>
```

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
    xmlns:whc="http://www.oxygenxml.com/webhelp/components"/>
```

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
    xmlns:whc="http://www.oxygenxml.com/webhelp/components"/>
```

In the output, you will find an element with the class: `wh_print_link`.

Switch Theme Button (`webhelp_theme_button`)

This component is used to generate a button that opens a drop-down menu for selecting the theme. To generate this component, the `<whc:webhelp_theme_button>` element must be specified in the HTML file as in the following example:

```
<whc:webhelp_theme_button
    xmlns:whc="http://www.oxygenxml.com/webhelp/components"/>
```

In the output, you will find an element with the class: `wh_theme`.

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
    xmlns:whc="http://www.oxygenxml.com/webhelp/components"/>
```

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 1344)*.

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
     xmlns:whc="http://www.oxygenxml.com/webhelp/components"/>
```

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 1348)*.

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

    xmlns:whc="http://www.oxygenxml.com/webhelp/components"/>
```

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:

```
<whc:webhelp_indexterms_link

    xmlns:whc="http://www.oxygenxml.com/webhelp/components"/>
```

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/>
```

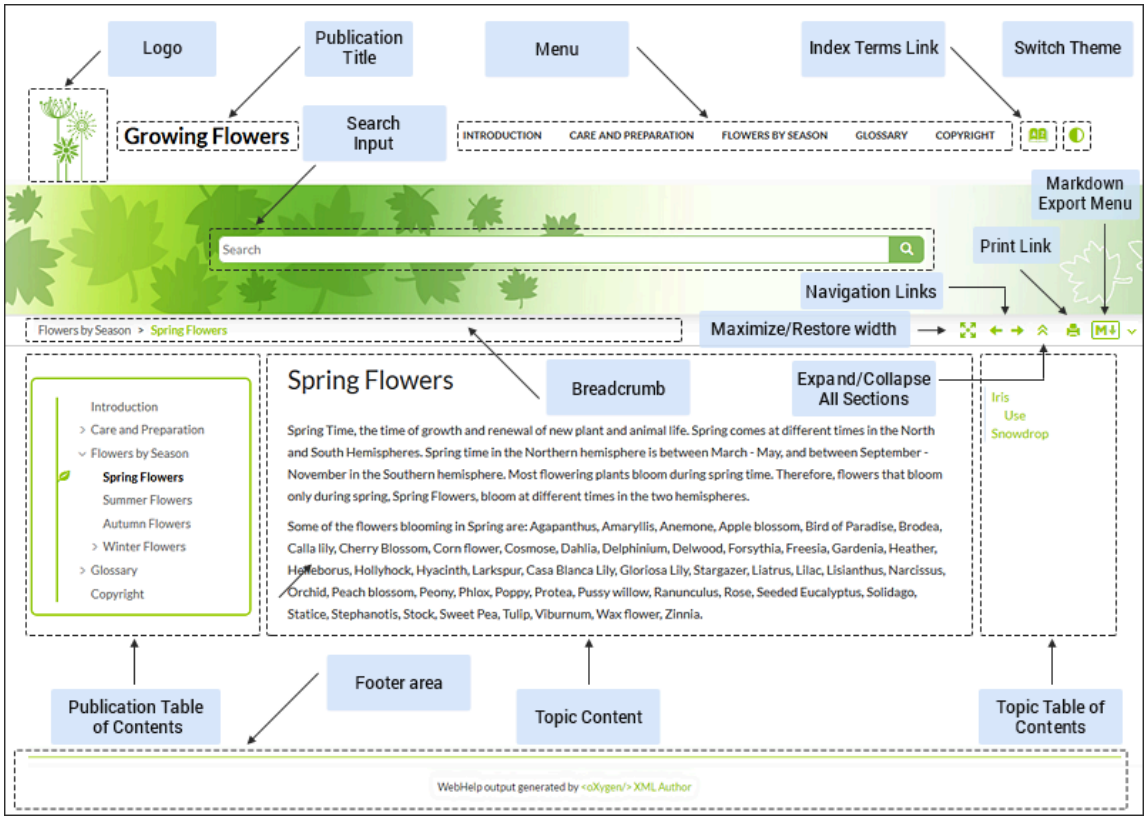
In the output, you will find a link to the skin resources.

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 396. Example of Topic Page Components



1. Publication Logo *(on page 1295)*
2. Publication Title *(on page 1294)*
3. Search Input *(on page 1295)*
4. Main Menu *(on page 1296)*
5. Index Terms Link *(on page 1295)*
6. Switch Theme Button *(on page 1295)*
7. Maximize/Restore width Component *(on page 1296)*
8. Navigation Links *(on page 1296)*
9. Expand/Collapse All Sections *(on page 1298)*
10. Print Link *(on page 1297)*
11. Markdown Export Menu *(on page 1297)*
12. Breadcrumb *(on page 1296)*
13. Publication Table of Contents *(on page 1297)*
14. Topic Content *(on page 1297)*
15. Topic Table of Contents *(on page 1298)*

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:

```
<whc:webhelp_publication_title
    xmlns:whc="http://www.oxygenxml.com/webhelp/components"/>
```

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:

```
<whc:webhelp_logo
    xmlns:whc="http://www.oxygenxml.com/webhelp/components"/>
```

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`.

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
    xmlns:whc="http://www.oxygenxml.com/webhelp/components"/>
```

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.

Switch Theme Button (`webhelp_theme_button`)

This component is used to generate a button that opens a drop-down menu for selecting the theme. To generate this component, the `<whc:webhelp_theme_button>` element must be specified in the HTML file as in the following example:

```
<whc:webhelp_theme_button
    xmlns:whc="http://www.oxygenxml.com/webhelp/components"/>
```

In the output, you will find an element with the class: `wh_theme`.

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
    xmlns:whc="http://www.oxygenxml.com/webhelp/components"/>
```

In the output, you will find an element with the class: `wh_search_input`.

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
    xmlns:whc="http://www.oxygenxml.com/webhelp/components"/>
```

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 1344)*.

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:

```
<whc:webhelp_breadcrumb
    xmlns:whc="http://www.oxygenxml.com/webhelp/components"/>
```

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.

Maximize to full width/Restore to normal width Link (`webhelp_full_width_link`)

This component is used to maximizes or restores the topic content to use the entire page space. To generate this component, the `<whc:webhelp_full_width_link>` element must be specified in the HTML file as in the following example:

```
<whc:webhelp_full_width_link
    xmlns:whc="http://www.oxygenxml.com/webhelp/components"/>
```

In the output, you will find an element with the class: `webhelp_full_width_link`.

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:

```
<whc:webhelp_navigation_links
    xmlns:whc="http://www.oxygenxml.com/webhelp/components"/>
```

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:

```
<whc:webhelp_print_link
    xmlns:whc="http://www.oxygenxml.com/webhelp/components"/>
```

In the output, you will find an element with the class: `wh_print_link`.

Markdown Export Menu (`whc:webhelp_md_menu`)

This component is used to generate a menu with "Copy as Markdown" and "View as Markdown" actions. To generate this component, the `<whc:webhelp_md_menu>` element must be specified in the HTML file as in the following example:

```
<whc:webhelp_md_menu
    xmlns:whc="http://www.oxygenxml.com/webhelp/components"/>
```

In the output, you will find an element with the class: `wh-md-menu`.

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:

```
<whc:webhelp_topic_content
    xmlns:whc="http://www.oxygenxml.com/webhelp/components"/>
```

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:

```
<whc:webhelp_publication_toc
    xmlns:whc="http://www.oxygenxml.com/webhelp/components"/>
```

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:

```
<whc:webhelp_topic_toc
    xmlns:whc="http://www.oxygenxml.com/webhelp/components"/>
```

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:

```
<whc:webhelp_expand_collapse_sections
    xmlns:whc="http://www.oxygenxml.com/webhelp/components"/>
```

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:

```
<whc:webhelp_feedback
    xmlns:whc="http://www.oxygenxml.com/webhelp/components"/>
```

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:

```
<whc:webhelp_child_links

    xmlns:whc="http://www.oxygenxml.com/webhelp/components"/>
```

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:

```
<whc:webhelp_related_links

    xmlns:whc="http://www.oxygenxml.com/webhelp/components"/>
```

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/>
```

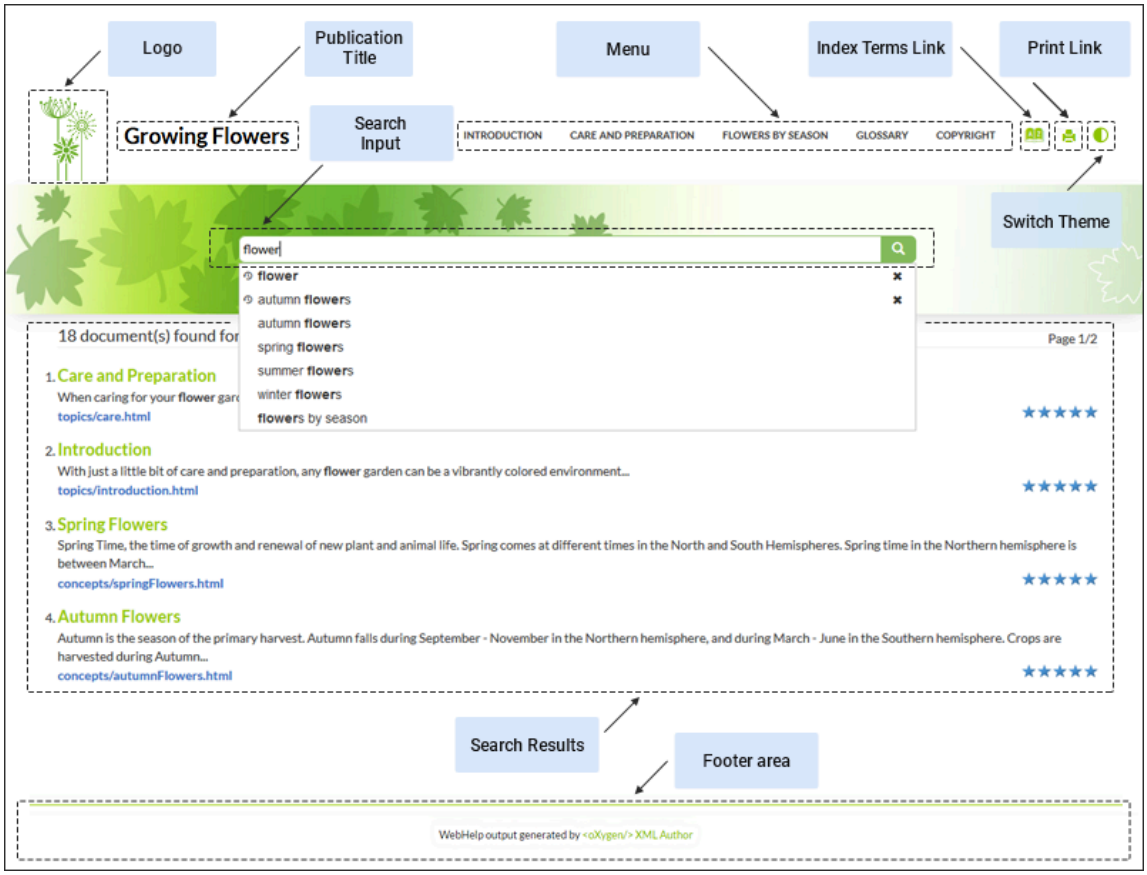
In the output, you will find a link to the skin resources.

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.

Figure 397. Example of Search Results Page Components



1. Publication Logo *(on page 1300)*
2. Publication Title *(on page 1300)*
3. Search Input *(on page 1301)*
4. Main Menu *(on page 1302)*
5. Index Terms Link *(on page 1302)*
6. Search Results *(on page 1301)*
7. Print Link *(on page 1301)*
8. Switch Theme Button *(on page 1301)*

The following components can be referenced in the *Search Results Page* (`wt_search.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:

```
<whc:webhelp_publication_title
    xmlns:whc="http://www.oxygenxml.com/webhelp/components"/>
```

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:

```
<whc:webhelp_logo
    xmlns:whc="http://www.oxygenxml.com/webhelp/components"/>
```

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
    xmlns:whc="http://www.oxygenxml.com/webhelp/components"/>
```

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:

```
<whc:webhelp_search_results
    xmlns:whc="http://www.oxygenxml.com/webhelp/components"/>
```

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:

```
<whc:webhelp_print_link
    xmlns:whc="http://www.oxygenxml.com/webhelp/components"/>
```

In the output, you will find an element with the class: `wh_print_link`.

Switch Theme Button (`webhelp_theme_button`)

This component is used to generate a button that opens a drop-down menu for selecting the theme. To generate this component, the `<whc:webhelp_theme_button>` element must be specified in the HTML file as in the following example:

```
<whc:webhelp_theme_button
    xmlns:whc="http://www.oxygenxml.com/webhelp/components"/>
```

In the output, you will find an element with the class: `wh_theme`.

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
    xmlns:whc="http://www.oxygenxml.com/webhelp/components"/>
```

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 1344)*.

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
    xmlns:whc="http://www.oxygenxml.com/webhelp/components"/>
```

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.

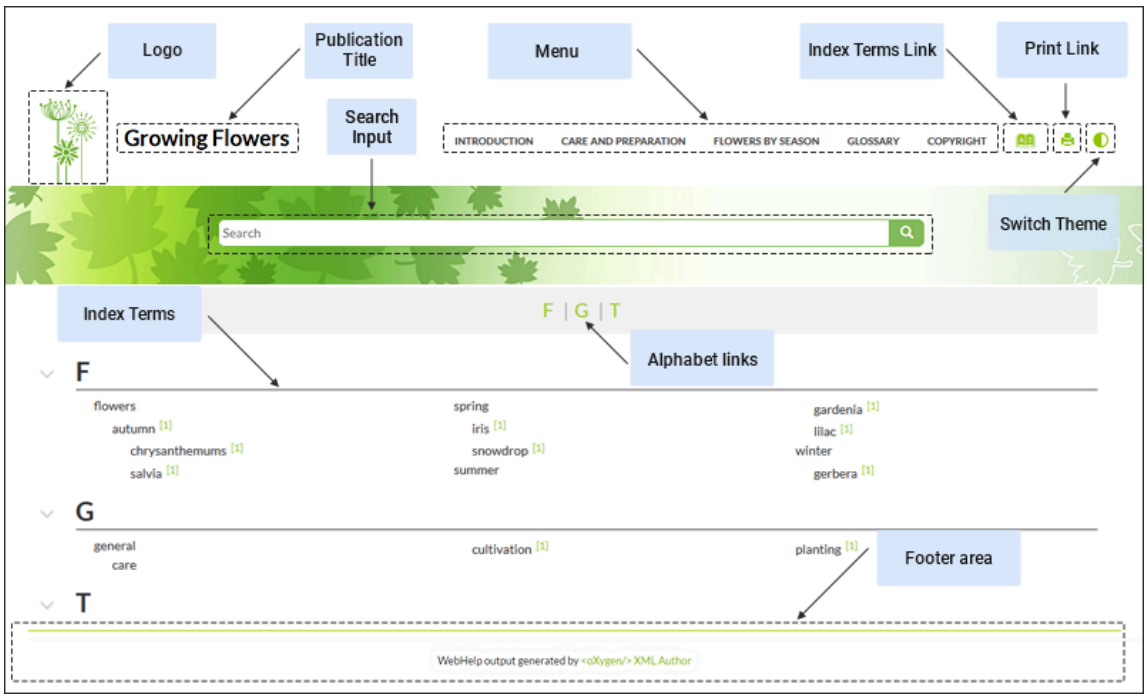
Index Terms Page

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.

Figure 398. Example of Index Terms Page Components



1. Publication Logo *(on page 1303)*
2. Publication Title *(on page 1303)*
3. Search Input *(on page 1304)*
4. Main Menu *(on page 1304)*
5. Index Terms Link *(on page 1305)*
6. Print Link *(on page 1304)*
7. Switch Theme Button *(on page 1304)*

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:

```
<whc:webhelp_publication_title
    xmlns:whc="http://www.oxygenxml.com/webhelp/components"/>
```

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:

```
<whc:webhelp_logo
    xmlns:whc="http://www.oxygenxml.com/webhelp/components"/>
```

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
    xmlns:whc="http://www.oxygenxml.com/webhelp/components"/>
```

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
    xmlns:whc="http://www.oxygenxml.com/webhelp/components"/>
```

In the output, you will find an element with the class: `wh_print_link`.

Switch Theme Button (`webhelp_theme_button`)

This component is used to generate a button that opens a drop-down menu for selecting the theme. To generate this component, the `<whc:webhelp_theme_button>` element must be specified in the HTML file as in the following example:

```
<whc:webhelp_theme_button
    xmlns:whc="http://www.oxygenxml.com/webhelp/components"/>
```

In the output, you will find an element with the class: `wh_theme`.

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
    xmlns:whc="http://www.oxygenxml.com/webhelp/components"/>
```

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 1344)*.

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
    xmlns:whc="http://www.oxygenxml.com/webhelp/components"/>
```

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.

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 2304)* as **WebHelp Responsive** output, follow these steps:

1. Select the  **Configure Transformation Scenario(s)** action from the 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**.

Adding Oxygen Feedback to WebHelp Responsive Documentation

You can add Oxygen Feedback in WebHelp Responsive for DITA output to benefit from a modern commenting system, advanced ready-to-use search engine, administrative interface, and **Oxygen XML Editor/Author** integration.

Comments Component

A comments component is presented 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.

External Search Engine

Oxygen Feedback can be configured as an external search engine for the **Oxygen WebHelp Responsive** output. This function be enabled from the *Content Indexing and Search* section that is available in the *Version Settings* page.

Administration Interface

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)*.

Oxygen XML Editor/Author Integration

An add-on is 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.

Deploying the Oxygen Feedback Comments Component

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). 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.
5. [Optional] To configure Oxygen Feedback as an external search engine check the **Enable content indexing** option.
6. Click **Continue**.
7. 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
```

8. [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 Developer 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 2306)* 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 1271)*

How to Create a Publishing Template

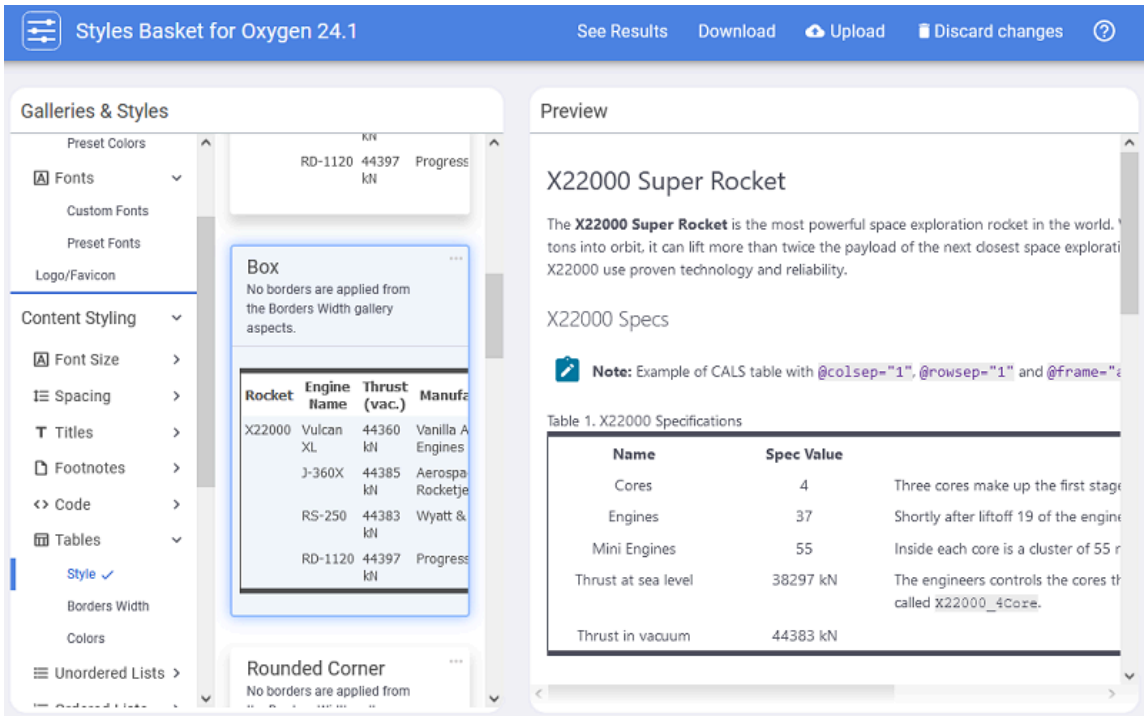
To create a customization, you can use the **Oxygen Styles Basket**, start from scratch or from an existing template, and then adapt it according to your needs.

Creating a Publishing Template Using the Oxygen Styles Basket

Oxygen Styles Basket is a handy free-to-use web-based visual tool that helps you create your own *Publishing Template Package* to customize your **DITA Map WebHelp Responsive** and/or **DITA Map PDF - based on HTML5 & CSS** transformation scenarios.

To create an *Oxygen Publishing Template*, simply visit the galleries and pick styling aspects. Various different types of styles can be selected (such as fonts, tables, lists, spacing, code) and all changes can be seen in the **Preview** pane. You can also click the **See Results** button to generate a preview of either PDF or WebHelp output.

Figure 399. Oxygen Styles Basket Interface



After creating your custom style, you can **Download** the current template, then back in **Oxygen**, click **Choose Custom Publishing Template** and select the downloaded archive in the transformation dialog to apply it.

At any time you can **Upload** a previously generated template for further customization.

Tip:
For more information about the **Oxygen Styles Basket**, see the following resources:

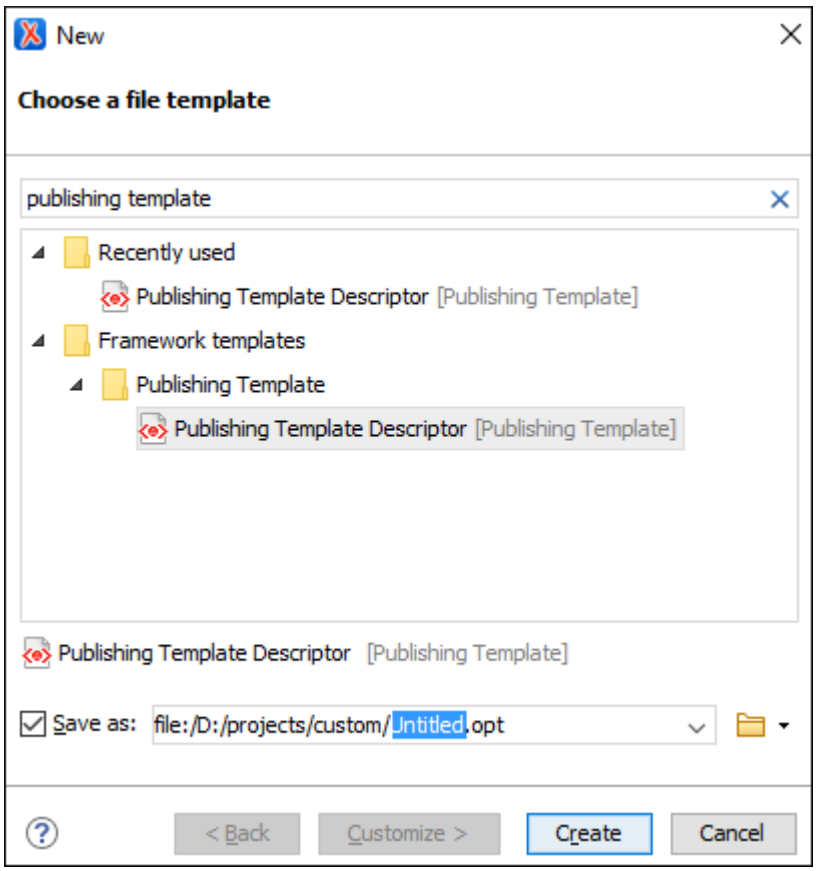
- **Video: Introducing the New Oxygen Styles Basket**
- **Webinar: Using Oxygen Styles Basket to Create CSS Customization from Scratch**

Creating a Publishing Template Starting from Scratch

To create a new *Oxygen Publishing Template (on page 2306)*, 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 400. Choosing the Publishing Template Descriptor Document 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 **DITA Map WebHelp Responsive** or **DITA Map PDF - based on HTML5 & CSS** transformation, the easiest way to create a new *Oxygen Publishing Template (on page 2306)* 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 1287)* 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/zNmXfKWXwO8

Related information

Publishing Template Package Contents for PDF Customizations *(on page 1459)*
Publishing Template Package Contents for WebHelp Responsive Customizations *(on page 1271)*

How to Create a Dark Mode-Ready Publishing Template

How to Create a Dark Mode-Ready Publishing Template From a Default One

By default, two templates supports both light and dark mode: **Light** and **Oxygen**. The following example uses the **Light** template:

1. Copy the existing Publishing Template (see Creating a Publishing Template Starting from an Existing Template *(on page 1466)*).
2. Edit the `light-theme.css` stylesheet with the colors you want to add/modify:

```
:root {
  --wh-primary-bg: #e3e3e3;
  ...
  --wh-title-color: #333;
}

@media screen {
  :root[data-wh-theme="dark"] {
    color-scheme: dark;
    --wh-primary-bg: #293136;
    ...
    --wh-title-color: #bfbfbf;
  }
}
```

3. If you added new color variables, add CSS rules using them (in `light.css` for example):

```
*[class ~= "topic/title"] {
  color: var(--wh-title-color);
}
```

4. Duplicate 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, then run the transformation scenario.

How to Create a Dark Mode-Ready Publishing Template From Scratch

1. Create a Publishing Template, see Creating a Publishing Template Starting from Scratch *(on page 1465)*.
2. Create a custom CSS stylesheet (`theme.css` for example) and add the light colors:

> **Note:**
> Built-in CSS rules will automatically use the variables defined here and in the next step.

```
:root {
  /* General */
  --wh-primary-bg: #fff;
  --wh-primary-color: #000;
  --wh-secondary-bg: #000;
  --wh-secondary-color: #fff;
  --wh-strong-color: #888;
  --wh-soft-color: #ccc;
  --wh-subtle-color: #f2f2f2;

  --wh-top-menu-bg: #e6e6e6;
  --wh-main-page-toc-bg: #f2f8ff;
  --wh-main-page-toc-border: #c0c0c0;
  --wh-main-page-shortdesc-color: #0d1a2b;
  --wh-close-toc-icon-border: #343a40;
  --wh-close-toc-button-shadow: rgba(0,0,0,.16);

  --wh-link-color: #337ab7;
  --wh-label-color: #666;
  --wh-pre-bg: #eee;
  --wh-tooltip-color: #5185cb;
  --wh-code-color: #800000;
  --wh-boolean-color: #008000;
  --wh-state-color: #ff0000;
  --wh-syntaxdiagram-color: #800000;
  --wh-markup-color: #639;
  --wh-fig-border: #c0c0c0;
  --wh-caution-bg: #ffd100;
  --wh-danger-bg: #c8102e;
  --wh-warning-bg: #ff8200;
  --wh-hazardstatement-bg: #0072ce;
  --wh-indexterm-bg: #fdf;
  --wh-go2top-bg: #444;
  --wh-go2top-border: #999;
  --wh-toggle-border: #347bb7;

  /* Search */
  --wh-search-color: #c93416;
  --wh-search-breadcrumb-color: #ccc;
  --wh-search-loader-base-bg: #eaecec;
  --wh-search-loader-glow-bg: #f3f3f3;
  --wh-search-facets-bg: #e3ebf3;
```

```
  --wh-search-facets-disabled-color: #b7b6b6;
  --wh-search-facets-border: #ebebeb;
  --wh-search-results-facets-bg: #f5f5f5;
  --wh-search-results-facets-color: #212c37;
  --wh-highlight-bg: #ffff88;
  --wh-scroll-bg: #d6d6d6;

  /* Change Tracking */
  --wh-oxy-delete-color: #ff0000;
  --wh-oxy-insert-color: #0000ff;
  --wh-oxy-change-done-color: #808080;
  --wh-oxy-range-color: #800000;
  --wh-oxy-comment-border: #ffa500;
  --wh-oxy-change-bg: #ffff00;
  --wh-oxy-attributes-footnote-bg: #00ffff;
  --wh-oxy-comment-footnote-bg: #ffffaa;
  --wh-oxy-delete-footnote-bg: #ffaaaa;
  --wh-oxy-insert-footnote-bg: #ddddff;

  /* Syntax Highlight */
  --wh-hl-json_key-color: #7263b9;
  --wh-hl-keyword-color: #7f0055;
  --wh-hl-string-color: #2a00ff;
  --wh-hl-comment-color: #006400;
  --wh-hl-directive-color: #8b26c9;
  --wh-hl-tag-color: #000096;
  --wh-hl-attribute-color: #ff7935;
  --wh-hl-value-color: #993300;
  --wh-hl-html-color: #000080;
  --wh-hl-xsl-color: #0066ff;
  --wh-hl-annotation-color: #808080;
  --wh-hl-tag-doctype-comment-color: #3f5fbf;
  --wh-hl-tag-doctype-color: #0000ff;

  /* OpenAPI */
  --wh-openapi-border: #c5c5c5;
  --wh-openapi-post-bg: #c2f2cb;
  --wh-openapi-get-bg: #ccebff;
  --wh-openapi-delete-bg: #ffceca;
  --wh-openapi-put-bg: #ffe7c7;
  --wh-openapi-head-bg: #f4e6ff;
}
```

3. In the same stylesheet, add the dark colors:



Note:
If you are using the same color in both light and dark modes, you only need to define it for the light mode.

```
@media screen {
  :root[data-wh-theme="dark"] {
    color-scheme: dark;
    /* General */
    --wh-primary-bg: #212529;
    --wh-primary-color: #e3e3e3;
    --wh-secondary-bg: #212529;
    --wh-secondary-color: #e3e3e3;
    --wh-strong-color: #e0e0e0;
    --wh-subtle-color: #3a3b3c;

    --wh-top-menu-bg: #2b3035;
    --wh-main-page-toc-bg: #22303c;
    --wh-main-page-shortdesc-color: #d5d5d5;
    --wh-close-toc-icon-border: var(--wh-primary-color);
    --wh-close-toc-button-shadow: rgba(255,255,255,.16);

    --wh-link-color: #76a5e0;
    --wh-label-color: #bbb;
    --wh-pre-bg: #444;
    --wh-code-color: #ffa500;
    --wh-boolean-color: #66ff66;
    --wh-state-color: #ff6666;
    --wh-syntaxdiagram-color: #ff4d4d;
    --wh-markup-color: #d1b3ff;
    --wh-caution-bg: #7f6000;
    --wh-danger-bg: #62002b;
    --wh-warning-bg: #993500;
    --wh-hazardstatement-bg: #0025cb;

    /* Search */
    --wh-search-color: #a2311a;
    --wh-search-breadcrumb-color: #555555;
    --wh-search-loader-base-bg: #3a3f44;
    --wh-search-loader-glow-bg: #4b525a;
    --wh-search-facets-bg: #2d333b;
```

```
    --wh-search-facets-disabled-color: var(--wh-label-color);
    --wh-search-facets-border: #444;
    --wh-search-results-facets-bg: #2c2f35;
    --wh-search-results-facets-color: #d8dee9;
    --wh-highlight-bg: #777700;
    --wh-scroll-bg: #444;

    /* Change Tracking */
    --wh-oxy-delete-color: #ff6b6b;
    --wh-oxy-insert-color: #66aaff;
    --wh-oxy-change-done-color: #ccc;
    --wh-oxy-range-color: #ff4d4d;
    --wh-oxy-comment-border: #995f00;
    --wh-oxy-change-bg: #666600;
    --wh-oxy-attributes-footnote-bg: #006666;
    --wh-oxy-comment-footnote-bg: #363600;
    --wh-oxy-delete-footnote-bg: #994d4d;
    --wh-oxy-insert-footnote-bg: #666699;

    /* Syntax Highlight */
    --wh-hl-json_key-color: #b5adda;
    --wh-hl-keyword-color: #ff89d8;
    --wh-hl-string-color: #b7a8ff;
    --wh-hl-comment-color: #00cf00;
    --wh-hl-directive-color: #d0a0ed;
    --wh-hl-tag-color: #aaaaff;
    --wh-hl-attribute-color: #ff9763;
    --wh-hl-value-color: #ff9661;
    --wh-hl-html-color: #a9a9ff;
    --wh-hl-xsl-color: #80b2ff;
    --wh-hl-annotation-color: #b0b0b0;
    --wh-hl-tag-doctype-comment-color: #a2b2e0;
    --wh-hl-tag-doctype-color: #a8a8ff;

    /* OpenAPI */
    --wh-openapi-border: #8a8a8a;
    --wh-openapi-post-bg: #7eae7a;
    --wh-openapi-get-bg: #7a9fcf;
    --wh-openapi-delete-bg: #b2746e;
    --wh-openapi-put-bg: #bfae86;
    --wh-openapi-head-bg: #7a70b2;
```

```
  }
}
```

4. Add custom variables for elements that require two color or image variants, such as:

```
:root {
  ...
  --wh-breadcrumb-color: #878787;
  --wh-related-links-color: #5185cb;
  --wh-publication-toc-color: #7b7a7a;

  --wh-search-image: url("resources/images/custom.png");
}

@media screen {
  :root[data-wh-theme="dark"] {
    color-scheme: dark;
    ...
    --wh-breadcrumb-color: #6ab1e3;
    --wh-related-links-color: #0d6efd;
    --wh-publication-toc-color: var(--wh-breadcrumb-color);

    --wh-search-image: url("resources/images/custom_dark.png");
  }
}
```

If you added new color variables, add CSS rules using them (in `custom.css` for example):

```
.wh_breadcrumb ol{
  color: var(--wh-breadcrumb-color);
}
.related-links a,
.related-links a:visited {
  color: var(--wh-related-links-color);
  padding-bottom: .3em;
}
.wh_publication_toc a,
.wh_publication_toc a:visited,
.wh_publication_toc a:hover {
  color: var(--wh-publication-toc-color);
}
.wh_search_input{
  padding: 115px 0;
  background: var(--wh-search-image) right no-repeat var(--wh-search-input-bg);
}
```

5. Open the template descriptor file *(on page 1271)* associated with your publishing template, add your custom CSS stylesheets in the `<resources>` section and declare the parameter in the `<parameters>` section:

```
<publishing-template>
  ...
  <webhelp>
    ...
    <resources>
      <css file="theme.css"/>
      <css file="custom.css"/>
    </resources>
    <parameters>
      ...
      <parameter name="webhelp.enable.dark.mode" value="yes"/>
    </parameters>
    ...
```

6. Duplicate 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, then run the transformation scenario.

Restriction:
If you open the output locally in **Firefox**, the theme settings may not persist between pages. This happens because Firefox treats each local page as a separate origin, preventing shared storage of theme preferences. If you deploy the output on an HTTP server or use another browser, theme settings will persist.

Related information
Using CSS Inspector to Identify Content for Custom CSS File *(on page 1332)*

How to Migrate a Publishing Template to be Dark Mode-Ready

If you already have a publishing template (created manually or via the **Oxygen Styles Basket**) prior to version 28.0, you can migrate it to support dark mode by following these steps:

1. Create a custom CSS stylesheet (`theme.css` for example) and add the default colors:

Note:
Built-in CSS rules will automatically use the variables defined here and in the next step.

```
:root {
  /* General */
```

```
--wh-primary-bg: #fff;
--wh-primary-color: #000;
--wh-secondary-bg: #000;
--wh-secondary-color: #fff;
--wh-strong-color: #888;
--wh-soft-color: #ccc;
--wh-subtle-color: #f2f2f2;

--wh-top-menu-bg: #e6e6e6;
--wh-main-page-toc-bg: #f2f8ff;
--wh-main-page-toc-border: #c0c0c0;
--wh-main-page-shortdesc-color: #0d1a2b;
--wh-close-toc-icon-border: #343a40;
--wh-close-toc-button-shadow: rgba(0,0,0,.16);

--wh-link-color: #337ab7;
--wh-label-color: #666;
--wh-pre-bg: #eee;
--wh-tooltip-color: #5185cb;
--wh-code-color: #800000;
--wh-boolean-color: #008000;
--wh-state-color: #ff0000;
--wh-syntaxdiagram-color: #800000;
--wh-markup-color: #639;
--wh-fig-border: #c0c0c0;
--wh-caution-bg: #ffd100;
--wh-danger-bg: #c8102e;
--wh-warning-bg: #ff8200;
--wh-hazardstatement-bg: #0072ce;
--wh-indexterm-bg: #fdf;
--wh-go2top-bg: #444;
--wh-go2top-border: #999;
--wh-toggle-border: #347bb7;

/* Search */
--wh-search-color: #c93416;
--wh-search-breadcrumb-color: #ccc;
--wh-search-loader-base-bg: #eaecec;
--wh-search-loader-glow-bg: #f3f3f3;
--wh-search-facets-bg: #e3ebf3;
--wh-search-facets-disabled-color: #b7b6b6;
--wh-search-facets-border: #ebebeb;
```

```
  --wh-search-results-facets-bg: #f5f5f5;
  --wh-search-results-facets-color: #212c37;
  --wh-highlight-bg: #ffff88;
  --wh-scroll-bg: #d6d6d6;

  /* Change Tracking */
  --wh-oxy-delete-color: #ff0000;
  --wh-oxy-insert-color: #0000ff;
  --wh-oxy-change-done-color: #808080;
  --wh-oxy-range-color: #800000;
  --wh-oxy-comment-border: #ffa500;
  --wh-oxy-change-bg: #ffff00;
  --wh-oxy-attributes-footnote-bg: #00ffff;
  --wh-oxy-comment-footnote-bg: #ffffaa;
  --wh-oxy-delete-footnote-bg: #ffaaaa;
  --wh-oxy-insert-footnote-bg: #ddddff;

  /* Syntax Highlight */
  --wh-hl-json_key-color: #7263b9;
  --wh-hl-keyword-color: #7f0055;
  --wh-hl-string-color: #2a00ff;
  --wh-hl-comment-color: #006400;
  --wh-hl-directive-color: #8b26c9;
  --wh-hl-tag-color: #000096;
  --wh-hl-attribute-color: #ff7935;
  --wh-hl-value-color: #993300;
  --wh-hl-html-color: #000080;
  --wh-hl-xsl-color: #0066ff;
  --wh-hl-annotation-color: #808080;
  --wh-hl-tag-doctype-comment-color: #3f5fbf;
  --wh-hl-tag-doctype-color: #0000ff;

  /* OpenAPI */
  --wh-openapi-border: #c5c5c5;
  --wh-openapi-post-bg: #c2f2cb;
  --wh-openapi-get-bg: #ccebff;
  --wh-openapi-delete-bg: #ffceca;
  --wh-openapi-put-bg: #ffe7c7;
  --wh-openapi-head-bg: #f4e6ff;
}

@media screen {
```

```
:root[data-wh-theme="dark"] {
  color-scheme: dark;
  /* General */
  --wh-primary-bg: #212529;
  --wh-primary-color: #e3e3e3;
  --wh-secondary-bg: #212529;
  --wh-secondary-color: #e3e3e3;
  --wh-strong-color: #e0e0e0;
  --wh-subtle-color: #3a3b3c;

  --wh-top-menu-bg: #2b3035;
  --wh-main-page-toc-bg: #22303c;
  --wh-main-page-shortdesc-color: #d5d5d5;
  --wh-close-toc-icon-border: var(--wh-primary-color);
  --wh-close-toc-button-shadow: rgba(255,255,255,.16);

  --wh-link-color: #76a5e0;
  --wh-label-color: #bbb;
  --wh-pre-bg: #444;
  --wh-code-color: #ffa500;
  --wh-boolean-color: #66ff66;
  --wh-state-color: #ff6666;
  --wh-syntaxdiagram-color: #ff4d4d;
  --wh-markup-color: #d1b3ff;
  --wh-caution-bg: #7f6000;
  --wh-danger-bg: #62002b;
  --wh-warning-bg: #993500;
  --wh-hazardstatement-bg: #0025cb;

  /* Search */
  --wh-search-color: #a2311a;
  --wh-search-breadcrumb-color: #555555;
  --wh-search-loader-base-bg: #3a3f44;
  --wh-search-loader-glow-bg: #4b525a;
  --wh-search-facets-bg: #2d333b;
  --wh-search-facets-disabled-color: var(--wh-label-color);
  --wh-search-facets-border: #444;
  --wh-search-results-facets-bg: #2c2f35;
  --wh-search-results-facets-color: #d8dee9;
  --wh-highlight-bg: #777700;
  --wh-scroll-bg: #444;
```

```
    /* Change Tracking */
    --wh-oxy-delete-color: #ff6b6b;
    --wh-oxy-insert-color: #66aaff;
    --wh-oxy-change-done-color: #ccc;
    --wh-oxy-range-color: #ff4d4d;
    --wh-oxy-comment-border: #995f00;
    --wh-oxy-change-bg: #666600;
    --wh-oxy-attributes-footnote-bg: #006666;
    --wh-oxy-comment-footnote-bg: #363600;
    --wh-oxy-delete-footnote-bg: #994d4d;
    --wh-oxy-insert-footnote-bg: #666699;

    /* Syntax Highlight */
    --wh-hl-json_key-color: #b5adda;
    --wh-hl-keyword-color: #ff89d8;
    --wh-hl-string-color: #b7a8ff;
    --wh-hl-comment-color: #00cf00;
    --wh-hl-directive-color: #d0a0ed;
    --wh-hl-tag-color: #aaaaff;
    --wh-hl-attribute-color: #ff9763;
    --wh-hl-value-color: #ff9661;
    --wh-hl-html-color: #a9a9ff;
    --wh-hl-xsl-color: #80b2ff;
    --wh-hl-annotation-color: #b0b0b0;
    --wh-hl-tag-doctype-comment-color: #a2b2e0;
    --wh-hl-tag-doctype-color: #a8a8ff;

    /* OpenAPI */
    --wh-openapi-border: #8a8a8a;
    --wh-openapi-post-bg: #7eae7a;
    --wh-openapi-get-bg: #7a9fcf;
    --wh-openapi-delete-bg: #b2746e;
    --wh-openapi-put-bg: #bfae86;
    --wh-openapi-head-bg: #7a70b2;
  }
}
```

2. If your custom stylesheets include color definitions, move them into the file created in the previous step:

Note:
If you are using the same color in both light and dark modes, you only need to define it for the light mode.

```
.topictitle1,
.topictitle2,
.topictitle3,
.topictitle4,
.topictitle5,
.topictitle6 {
  color: #070d59;
}
```

```
:root {
  ...
  --wh-title-color: #070d59;
}

@media screen {
  :root[data-wh-theme="dark"] {
    --wh-title-color: #5ca6e4;
  }
}

.topictitle1,
.topictitle2,
.topictitle3,
.topictitle4,
.topictitle5,
.topictitle6 {
  color: var(--wh-title-color);
}
```

3. Open your publishing template descriptor file *(on page 1271)* and add the `theme.css` stylesheet in the `<resources>` section and declare the parameter in the `<parameters>` section:

```
<publishing-template>
  ...
  <webhelp>
    ...
    <resources>
      <css file="theme.css"/>
      <css file="styles.css"/>
```

```
    <css file="custom.css"/>
  </resources>
  <parameters>
    ...
    <parameter name="webhelp.enable.dark.mode" value="yes"/>
  </parameters>
  ...
```

4. Run the transformation scenario with the updated template.

How to Edit a Packed Publishing Template

To edit an existing *Oxygen Publishing Template (on page 2306)* 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 1459)*
Publishing Template Package Contents for WebHelp Responsive Customizations *(on page 1271)*

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 1271)*. 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 1420)*. 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 1420)* and webhelp.publishing.template.descriptor *(on page 1421)* parameters.

The webhelp.publishing.template *(on page 1420)* parameter specifies the path to the ZIP archive (or root folder) while the webhelp.publishing.template.descriptor *(on page 1421)* 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.

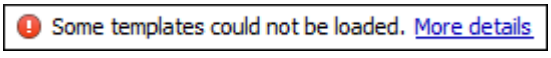
How to Share a Publishing Template

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 1271)* 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:



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 1271)*.

Also, if a template could be loaded, but certain elements could not be found in the descriptor file *(on page 1271)*, 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 1274)* cannot be found.

Converting Old Templates to Newer Versions

WebHelp templates that were created in older versions of Oxygen XML Developer can be converted to the Publishing Template format that was introduced in Oxygen XML Developer version 20.0. This section contains procedures for converting old templates depending on the version they were created in.

Important:
You must convert each version sequentially, one at a time (for example, you cannot convert version 27.0 directly to version 28.0, you must first convert it to 27.1 and then to 28.0). Make sure you follow the procedure in each migration topic to ensure a smooth transition and avoid compatibility issues.

Convert Version 27.1 Publishing Templates to Version 28

If you have a custom Publishing Template that was created in Oxygen XML Developer version 27.1, the following conversion procedure is required for the template to be compatible with Oxygen XML Developer version 28.0:

1. In the **Project** view, add the root directory for your custom Publishing Template (you can use a linked folder *(on page 333)* 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 v28**, 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 28.0.

Related information

Convert Version 25 - 27.0 Publishing Templates to Version 27.1 *(on page 1328)*
Convert Version 24.1 Publishing Templates to Version 25 *(on page 1328)*
Convert Version 24.0 Publishing Templates to Version 24.1 *(on page 1329)*
Convert Version 22 - 23 Publishing Templates to Version 24 *(on page 1330)*
Convert Version 21 Publishing Templates to Version 22 *(on page 1330)*
Convert Version 20 Publishing Templates to Version 21 *(on page 1331)*

Convert Version 25 - 27.0 Publishing Templates to Version 27.1

If you have a custom Publishing Template that was created in Oxygen XML Developer version 25, 26, or 27.0, the following conversion procedure is required for the template to be compatible with Oxygen XML Developer version 27.1:

1. In the **Project** view, add the root directory for your custom Publishing Template (you can use a linked folder *(on page 333)* 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 v27.1**, 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 27.1.

Related information

Convert Version 24.1 Publishing Templates to Version 25 *(on page 1328)*
Convert Version 24.0 Publishing Templates to Version 24.1 *(on page 1329)*
Convert Version 22 - 23 Publishing Templates to Version 24 *(on page 1330)*
Convert Version 21 Publishing Templates to Version 22 *(on page 1330)*
Convert Version 20 Publishing Templates to Version 21 *(on page 1331)*

Convert Version 24.1 Publishing Templates to Version 25

If you have a custom Publishing Template that was created in Oxygen XML Developer version 24.1, the following conversion procedure is required for the template to be compatible with Oxygen XML Developer version 25.0:

1. In the **Project** view, add the root directory for your custom Publishing Template (you can use a linked folder *(on page 333)* 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 v25**, 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 25.0.

Related information

Convert Version 22 - 23 Publishing Templates to Version 24 *(on page 1330)*
Convert Version 21 Publishing Templates to Version 22 *(on page 1330)*
Convert Version 20 Publishing Templates to Version 21 *(on page 1331)*

Convert Version 24.0 Publishing Templates to Version 24.1

If you have a custom Publishing Template that was created in Oxygen XML Developer version 24.0, the following conversion procedure is required for the template to be compatible with Oxygen XML Developer 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 333)* 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.

Related information

Convert Version 22 - 23 Publishing Templates to Version 24 *(on page 1330)*
Convert Version 21 Publishing Templates to Version 22 *(on page 1330)*
Convert Version 20 Publishing Templates to Version 21 *(on page 1331)*

Convert Version 22 - 23 Publishing Templates to Version 24

If you have a custom Publishing Template that was created in Oxygen XML Developer version 23.0 or 23.1, the following conversion procedure is required for the template to be compatible with Oxygen XML Developer version 24:

1. In the **Project** view, add the root directory for your custom Publishing Template (you can use a linked folder *(on page 333)* 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**, 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.

Related information

Convert Version 24.0 Publishing Templates to Version 24.1 *(on page 1329)*
Convert Version 21 Publishing Templates to Version 22 *(on page 1330)*
Convert Version 20 Publishing Templates to Version 21 *(on page 1331)*

Convert Version 21 Publishing Templates to Version 22

If you have a custom Publishing Template that was created in Oxygen XML Developer version 21.0 or 21.1, the following conversion procedure is required for the template to be compatible with Oxygen XML Developer version 22:

1. In the **Project** view, add the root directory for your custom Publishing Template (you can use a linked folder *(on page 333)* 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 1329)*
Convert Version 22 - 23 Publishing Templates to Version 24 *(on page 1330)*
Convert Version 20 Publishing Templates to Version 21 *(on page 1331)*

Convert Version 20 Publishing Templates to Version 21

If you have a custom Publishing Template that was created in Oxygen XML Developer version 20.0 or 20.1, the following conversion procedure is required for the template to be compatible with Oxygen XML Developer 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 *(on page 333)* 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 *(on page 1331)*
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 *(on page 1329)*
Convert Version 22 - 23 Publishing Templates to Version 24 *(on page 1330)*
Convert Version 21 Publishing Templates to Version 22 *(on page 1330)*

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 2306)* 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 changes 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:

```
.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:

```
.wh_footer {

  background-color: #255890;

}
```

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 1333)* or the `args.css` parameter *(on page 1334)*.



Fastpath:
Regenerating the output to see the changes made in the CSS is not required. Instead, you can directly edit the files in *`WebHelp Output Directory`*`/oxygen-webhelp/template` and reload the page in your browser. Once you obtained the desired output, simply copy the stylesheet back to your publishing template folder.

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 1464)*.
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 1271)* associated with your publishing template and add your custom CSS in the *resources* section.

```
<publishing-template>

    ...

    <webhelp>

        ...

        <resources>

            ...

            <css file="resources/MyCustom.css"/>
```

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 (on page 2306)* or using one of the HTML fragment placeholder parameters *(on page 1422)*). 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 (on page 515)*, 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 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.

  > **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>
  <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 *(on page 1267)*, 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 1398)*.

- **Inline JavaScript or CSS Content**:

JavaScript:

```
<script type="text/javascript">
    /* Include JavaScript code here. */

    function myFunction() {
      return true;
    }
</script>
```

CSS:

```
<style>
    /* Include CSS style rules here. */

    *{
      color:red
    }
</style>
```

Note:

If you have special characters (e.g. `&`, `<`) 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.

```
<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:

```
<script type="text/javascript" src="${path(oxygen-webhelp-template-dir)}/"></script>
```

- **In text content** - Using the `<whc:macro>` template component:

```
<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.

```
${i18n(string.id)}
```

param

Returns the value of a transformation parameter.

```
${param(webhelp.show.main.page.tiles)}
```

env

Returns the value of an environment variable.

```
${env(JAVA_HOME)}
```

system-property

Returns the value of a system property.

```
${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.

```
${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.

```
${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`).

```
${topic-xpath(string-join(//shortdesc//text(), ' '))}
```

oxygen-webhelp-build-number

Returns the current WebHelp distribution ID (build number).

```
${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 1308)*.
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 1271)* associated with your publishing template and add a reference to the custom HTML fragment in the *html-fragments* section.

```
<publishing-template>
    ...
    <webhelp>
      ...
      <html-fragments>
        <fragment
          file="html-fragments/custom-html.xml"
          placeholder="webhelp.fragment.head"/>
```

> **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 1278)*.

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 1278)*.

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 1278)*
Publishing Template Package Contents for WebHelp Responsive Customizations *(on page 1271)*

How to Change Ordered Lists Layout and Styles

How to Continue Numbering in Ordered Lists

Ordered lists (`<ol>`) typically begin numbering at 1 in XHTML output. However, in certain situations you may need to start at a given index. You can control the starting number using the `@outputclass="start-x"` attribute (where X is the number you want the list to start with) on the `<ol>` element. This is especially useful when lists do not have the same parent:

```
<table frame="all">
  <title>Table with nested order lists</title>
  <tgroup cols="1">
    <tbody>
```

```
      <row>
        <entry>
          <ol>
            <li>First Item</li>
            <li>Second Item</li>
          </ol>
        </entry>
      </row>
      <row>
        <entry>
          <ol outputclass="start-3">
            <li>Third Item</li>
            <li>Fourth Item</li>
          </ol>
        </entry>
      </row>
    </tbody>
  </tgroup>
</table>
```

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:

   ```
   <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:

   ```
   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 1341)* or the `args.css` parameter *(on page 1341)*.

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 1464)*.
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 1271)* associated with your publishing template and add your custom CSS in the *resources* section.

```
<publishing-template>
    ...
    <webhelp>
        ...
        <resources>
            ...
            <css file="resources/MyCustom.css"/>
```

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 Add 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 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-json
- language-yaml
- language-xml
- language-bourne
- language-c
- language-cmd
- language-cpp
- language-csharp
- language-css
- language-dtd
- language-ini
- language-java
- language-javascript
- language-lua
- language-perl
- language-powershell
- language-php
- language-python
- language-ruby
- language-sql
- language-xquery

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:

The following codeblock with the `@outputclass` set as `language-css`:

```
<codeblock outputclass="language-css" id="codeblock_1">@page preface-page {
 background-color:silver;
 @top-center{
  content: "Custom Preface Header";
 }
}
*[class ~= "topic/topic"][@topicrefclass ~= "bookmap/preface"] {
 page: preface-page;
}</codeblock>
```

would like this in WebHelp output:

```
@page preface-page {
 background-color:silver;
 @top-center{
  content: "Custom Preface Header";
 }
}
*[class ~= "topic/topic"][@topicrefclass ~= "bookmap/preface"] {
 page: preface-page;
}
```

How to Show or Hide Navigation Links in Topic Pages

The topic pages *(on page 1226)* 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).

```
<map id="example_map" title="Example Map">
 <topicref href="../topics/ParentTopic.dita" collection-type="sequence">
  <topicref href="../topics/Childtopic.dita"/>
 </topicref>
```

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 2304)* 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 Developer.
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:

```
<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:

```
.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 1333)* or the `args.css` parameter *(on page 1334)*.

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 1267)*, follow this procedure:

1. If you have not already created a Publishing Template, see How to Create a Publishing Template *(on page 1464)*.
2. Open the template descriptor file *(on page 1271)* associated with your publishing template and add the `webhelp.show.top.menu` parameter in the *parameters* section with its value set to `no`.

```
<publishing-template>

    ...

    <webhelp>

        ...

        <parameters>

            <parameter name="webhelp.show.top.menu" value="no"/>

        </parameters>

    </webhelp>
```

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 1278)* 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 1267)*, follow this procedure:

1. If you have not already created a Publishing Template, see How to Create a Publishing Template *(on page 1464)*.
2. Open the template descriptor file *(on page 1271)* 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.

```
<publishing-template>
    ...
    <webhelp>
        ...
        <parameters>
            <parameter name="webhelp.fragment.welcome"
                value="path/to/welcome.xhtml" type="filePath"/>
        </parameters>
    </webhelp>
```

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.

Related information

How to Insert Custom HTML Content *(on page 1334)*

How to Create a Custom Footer

The main page of the WebHelp Responsive output contains a set of empty placeholders *(on page 1278)* 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 1267)*, follow this procedure:

1. If you have not already created a Publishing Template, see How to Create a Publishing Template *(on page 1464)*.
2. Open the template descriptor file *(on page 1271)* associated with your publishing template and add the `webhelp.fragment.footer` parameter in the *html-fragments* section with its value set to a path of a file that contains well-formed XHTML content.

```
<publishing-template>
  ...
  <webhelp>
    ...
    <html-fragments>
      <fragment file="html/footer.xhtml" placeholder="webhelp.fragment.footer"/>
```

```
    </html-fragments>
</webhelp>
```

Important:
This parameter should only be used if you are using a valid, purchased license of Oxygen XML Developer (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).
- 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 2304)*. 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 Developer.
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:

```
<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:

```
.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 1333)* or the `args.css` parameter *(on page 1334)*.

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 Developer.
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:

```
<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:

```
.wh_tile[data-id='growing-flowers']> div {
    background-image:url('resources/flower.png');
}
```

3. Reference the CSS file in a *WebHelp Responsive* transformation using an *Oxygen Publishing Template (on page 1333)* or the `args.css` parameter *(on page 1334)*.

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 1267)*, follow this procedure:

1. If you have not already created a Publishing Template, see How to Create a Publishing Template *(on page 1464)*.
2. Open the template descriptor file *(on page 1271)* 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.

```
<publishing-template>
    ...
    <webhelp>
        ...
        <resources>
            <logo
                file="images/logo.png"
                target-url="http://www.example.com"
                alt="Alternate text for the logo image"
                new-tab="true"/>
        </resources>
    </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.

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 1267)*, follow this procedure:

1. If you have not already created a Publishing Template, see Working with Publishing Templates *(on page 1308)*.
2. Open the template descriptor file *(on page 1271)* 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.

```
<publishing-template>
    ...
    <webhelp>
      ...
      <resources>
          ...
          <favicon file="images/favicon.png"/>
```

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 by adding an `<object>` element, as in one of the following examples:

   ```
   <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:

   ```
   <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.

```
<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 by adding an `<object>` element, as in one of the following examples:

   ```
   <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:

   ```
   <object outputclass="audio">
     <param name="src" value="audio/MyClip.mp3"/>
   </object>
   ```

2. Apply a **DITA to WebHelp** transformation to obtain the output.

Result: The transformation converts the `<object>` element to an HTML5 `<audio>` element.

```
<audio controls="controls"><source type="audio/mpeg" src="MyClip.mp3"></source>
</audio>
```

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 manually adding an `<object>` element, as in one of the following examples:

   ```
   <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:

   ```
   <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**:

   ```
   <object outputclass="iframe" data="https://www.youtube.com/embed/m_vv2s5Trn4"/>
     <param name="allowfullscreen" value="true"/>
   </object>
   ```

3. Apply a **DITA to WebHelp** transformation to obtain the output.

Result: The transformation converts the `<object>` element to an HTML5 `<iframe>` element.

```
<iframe controls="controls" src="https://www.youtube.com/embed/m_vv2s5Trn4">
</iframe>
```

How to Add MathML Equations in WebHelp Output

Currently, the majority of modern browsers have native support to render **MathML** equations embedded in the **HTML** code. If your browser that does not have support for MathML,MathJax is a solution to properly view MathML equations embedded in **HTML** content in a variety of browsers.

If you have DITA 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:

```
<script type="text/javascript" id="MathJax-script" async
  src="https://cdnjs.cloudflare.com/ajax/libs/mathjax/3.0.0/es5/latest?tex-mml-chtml.js">
</script>
```

Result: The equation should now be properly rendered in the WebHelp output for other browsers.

Related information

How to Insert Custom HTML Content *(on page 1334)*

Getting Started with MathJax Components

Searching the Output

This section contains topics that explain how to use some of the search features in WebHelp Responsive output.

Built-in JS Based Search Engine Customizations

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 following values 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

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 1267)*, follow this procedure:

1. If you have not already created a Publishing Template, see How to Create a Publishing Template *(on page 1464)*.
2. Open the template descriptor file *(on page 1271)* associated with your publishing template and add the `default.language` parameter in the *parameters* section with its value set to `ja-jp`.

```
<publishing-template>
    ...
    <webhelp>
        ...
        <parameters>
            <parameter name="default.language" value="ja-jp"/>
        </parameters>
    </webhelp>
```

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 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`.

   > **Note:**
   > Alternatively, you could set the `@xml:lang` attribute on the root of the *DITA map (on page 2304)* and the referenced topics to `ja-jp`. Another alternative for DITA output is to use the `webhelp.search.japanese.dictionary` parameter to specify a path to a Japanese dictionary that will be used by the *Kuromoji* morphological engine (note that the encoding for the dictionary must be **UTF8**).

2. Run the WebHelp transformation scenario to generate the output.

How to Implement a Custom Search Filter

It is possible to implement a custom search filter (search input component) in your WebHelp Responsive output. The search input component is where users enter search queries to locate certain content within the WebHelp output.

To integrate a custom search filter, follow these steps:

1. If you have not already created a Publishing Template, see How to Create a Publishing Template *(on page 1464)*.
2. Create the following items in the folder that contains your publishing descriptor file (the `.opt` file):
   - A folder named `js`.
   - A folder named `fragments`.
3. In the **js** folder, create a file named **search-filter.js**.
4. As a starting point, you can copy the following content to the **search-filter.js** file:

```
/**
 * Object that implements the methods required by WebHelp to run a search filter.
 */
function CustomSearchFilter() {


    /**
     * Method required to run the search filter in webhelp. It is called when the users
     * executes the query in the search page.
     *
     * @param {WebHelpAPI.SearchResult} searchResult The search result for the executed query.
     *
     * @return A list of WebHelpAPI.SearchResult objects
     */
    this.filterResults = function (searchResult) {
        // implement filter
        return filteredResults;
```

```
    }
}


// Set the Search Filter to WebHelp
WebHelpAPI.setCustomSearchFilter(new CustomSearchFilter());
...
```

Note:
See the API Search Objects section *(on page 1368)* for details on how to create a `WebHelpAPI.SearchResult` object.

5. Implement your custom search filter.
6. In the **fragments** folder, create a file named **search-filter-script-fragment.xml**.
7. In the **search-filter-script-fragment.xml** file, define the scripts that are required for your custom search filter to run. For example:

```
<div>
    <script src="${oxygen-webhelp-template-dir}/js/search-filter.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 1274)* for more details):

```
<fileset>
  <include name="js/**"/>
</fileset>
```

9. Set the transformation parameters needed to enable the custom search filter. For this, open the `.opt` file and add the following content inside the `<webhelp>` element:

```
<html-fragments>
  <fragment file="fragments/search-filter-script-fragment.xml"
      placeholder="webhelp.fragment.head.search.page"/>
</html-fragments>
```

10. Run the transformation with this publishing template selected.

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 2304)* 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 1276)* 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:

   ```
   <topicref href="../topics/internal-topic1.dita" search="no"/>
   ```

2. Save your changes to the *DITA map*.
3. Run your WebHelp system transformation.

Oxygen Feedback Search Engine

How to Configure Faceted Search in WebHelp Output

A *faceted search* is a powerful tool that allows users to refine search results by selecting filters or facets. *Facets* are predefined categories that are associated with search results. By selecting one or more facets, users can narrow down their search results to a specific category or set of categories.

Configure Oxygen Feedback as an External Search Engine

To enable faceted searches, you need to have a search engine that supports this functionality. The **Oxygen Feedback** search engine implements faceted searches and can be easily configured as a search engine for

WebHelp, see Adding Oxygen Feedback to WebHelp Responsive Documentation *(on page 1306)* for more details.

> **Attention:**
> The default search engine that comes embedded in the WebHelp Responsive output does not support faceted searches.

Defining Facets Using a DITA Subject Scheme Map

A subject scheme map can be used to define controlled values and subject definitions. *Subject definitions* are classifications and sub-classifications that compose a tree. Subject definitions provide semantics that can be used in conjunction with taxonomies and ontologies.

The `<subjectdef>` element is used to define both a subject category and a list of controlled values. The parent `<subjectdef>` element defines the category, and the children `<subjectdef>` elements define the controlled values.

The following example defines the "Operating system" category, with "Linux" and "Windows" sub-categories. The controlled values (facet values) are: "RedHat Linux", "SUSE Linux", "Windows 7", and "Windows 10".

```
<subjectScheme>
    ...
    <hasInstance>
        <subjectdef keys="os" navtitle="Operating system">
            <subjectdef keys="linux" navtitle="Linux">
                <subjectdef keys="redhat" navtitle="RedHat Linux"/>
                <subjectdef keys="suse" navtitle="SUSE Linux"/>
            </subjectdef>
            <subjectdef keys="windows" navtitle="Windows">
                <subjectdef keys="win7" navtitle="Windows 7"/>
                <subjectdef keys="win10" navtitle="Windows 10"/>
            </subjectdef>
        </subjectdef>
    </hasInstance>
    ...
</subjectScheme>
```

Associating Faceted Values With a Topic Using a DITA Classification Map

The classification domain provides elements that enable map authors to indicate information about the subject matter of DITA topics. The subjects are defined in subject scheme maps, and the subjects are referenced using the `@keyref` attribute.

The following example shows you how to associate a faceted value with a topic:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE map PUBLIC "-//OASIS//DTD DITA Classification Map//EN" "classifyMap.dtd">
```

```
<map>
    <title>Classification map</title>
    <topicref keyref="how-to-install-on-suse.dita">
      <topicsubject keyref="linux">
        <subjectref keyref="suse"/>
      </topicsubject>
    </topicref>

    <topicref keyref="how-to-install-win7.dita">
      <topicsubject keyref="windows">
        <subjectref keyref="win7"/>
      </topicsubject>
    </topicref>
</map>
```

Note:
The facet information cascades into child `<topicref>` elements.

Refining the Search Results by Using Facets in the Search Page

The configured facets are displayed in the search page, allowing you to narrow down the results.

When a user selects a facet, the search results are updated to only include the topics that match the selected facets. If multiple facet values are selected from the same category/facet, the search results display all topics with at least one facet. On the other hand, if multiple facet values from distinct facets are selected, the search results display all topics with all selected facet values.

Related information
Adding Oxygen Feedback to WebHelp Responsive Documentation *(on page 1306)*

How to Add Searchable Labels in WebHelp Output

It is possible to add *searchable labels* in WebHelp Responsive output that can be clicked to search for topics with that exact same label. Labels are textual words attached to a DITA topic that enables it to be easily found using the search function. These labels can help you organize your topics, making it more accessible to retrieve topics for a specific text.

Configure Oxygen Feedback as an External Search Engine

To enable *searchable labels*, you need to have a search engine that supports this functionality. The **Oxygen Feedback** search engine implements *searchable labels* and can be easily configured as a search engine for WebHelp. See Adding Oxygen Feedback to WebHelp Responsive Documentation *(on page 1306)* for more details.

Attention:
The default search engine that comes embedded in the WebHelp Responsive output does not support searchable labels. It will simply perform a standard search using the content within the label.

How to Add Searchable Labels in a DITA Topic

The generation of *searchable labels* in the WebHelp Responsive output is activated by default. You need to insert the desired text (to be displayed in the label in the output) in a `<keyword>` element with an `@outputclass` attribute set to **label** within the prolog of each topic that you want to have that label displayed in the output.

Note:
You can right-click anywhere within the topic in **Author** mode and select **Insert > Insert Label** to quickly insert the needed structure in the prolog.

For example:

```
<prolog>
    <metadata>
        <keywords>
            <keyword outputclass="label">Customization</keyword>
        </keywords>
    </metadata>
</prolog>
```

This would add a label that contains the text "Customization" in the output for the particular topic. If the user clicks that label, the search engine will search for all topics that have this same label defined.

Transformation Parameters for Generating Searchable Labels

You can have more control over how the labels are generated in the WebHelp Responsive output by using the **webhelp.labels.generation.mode** transformation parameter. The possible values for this parameter are:

- **keywords-label** - Generates labels for each defined `<keyword>` element that has the `@outputclass` attribute value set to **label**.
- **keywords** - Generates labels for each defined `<keyword>` element. If the topic contains `<keyword>` elements with the `@outputclass` attribute value set to **label**, then only these elements will have labels generated for them in the output.
- **disable** - Disables the generation of labels in the WebHelp Responsive output.

Note:
The default value for the **webhelp.labels.generation.mode** transformation parameter is **keywords-label**.

Searchable Labels in WebHelp Responsive Output

The WebHelp Responsive transformation will generate a component that renders the text value of the `<keyword>` element. When the user clicks that component, they will be redirected to the search page with the search query populated for them and the search engine will display all topics that have the same text value defined in the prolog.

Custom Search Engine

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 1267)*, 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:

   ```
   <script async src="https://cse.google.com/cse.js?cx=xxxxxxxxxxxxxx"></script>
   <div class="gcse-searchresults-only"></div>
   ```

5. Create a folder named `html-fragments`, and in that folder, create a file named `google-script.html`.
6. Inside `google-script.html`, add only the `<script>` element. The file should look like this:

   ```
   <script async src="https://cse.google.com/cse.js?cx=xxxxxxxxxxxxxx"></script>
   ```

7. In the `html-fragments` folder, create another file named `google-results.html`.
8. Inside *google-results.html*, add only the * * element and include the `data-autoSearchOnLoad="true" data-queryParameterName="searchQuery"` attributes. The file should look like this:

   ```
   <div class="gcse-searchresults-only" data-autoSearchOnLoad="true"
     data-queryParameterName="searchQuery"></div>
   ```

> **Note:**
> For more information about other supported attributes, see Google Custom Search: Supported Attributes.

9. Open the template descriptor file *(on page 1271)* associated with your publishing template and add the `webhelp.google.search.script` parameter in the parameters section, setting its value to reference `google-script.html`. Then add the `webhelp.google.search.results` parameter, with its value set to `google-results.html`.

```
<publishing-template>
  ...
  <webhelp>
    ...
    <parameters>
    ...
    <parameter
      name="webhelp.google.search.script"
      value="html-fragments/google-script.html"
      type="filePath"/>
    <parameter
      name="webhelp.google.search.results"
      value="html-fragments/google-results.html"
      type="filePath"/>
    </parameters>
  </webhelp>
</publishing-template>
```

10. Open the *DITA Map WebHelp Responsive* transformation scenario.
11. Click the **Choose Custom Publishing Template** link and select your template.
12. Click **OK** to save the changes to the transformation scenario.
13. Run the transformation scenario.

> **Important:**
> The Google search feature will not work locally, it must be deployed on an HTTP server.

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:

```
<script async src="https://cse.google.com/cse.js?cx=xxxxxxxxxxxxxx"></script>
<div class="gcse-searchresults-only"></div>
```

5. Create a file named `google-script.html`, and add only the `<script>` element. The file should look like this:

```
<script async src="https://cse.google.com/cse.js?cx=xxxxxxxxxxxxxx"></script>
```

6. Create a file named `google-results.html`, add only the * * element and include the `data-autoSearchOnLoad="true" data-queryParameterName="searchQuery"` attributes. The file should look like this:

```
<div class="gcse-searchresults-only" data-autoSearchOnLoad="true"
  data-queryParameterName="searchQuery"></div>
```

> **Note:**
> For more information about other supported attributes, see Google Custom Search: Supported Attributes.

7. Edit the *DITA Map WebHelp Responsive* transformation scenario and choose a *template*.
8. Switch to the **Parameters** tab and edit the `webhelp.google.search.script` parameter to reference the `google-script.html` file. Then edit the `webhelp.google.search.results` parameter to reference the `google-results.html` file.
9. Click **Ok** and run the transformation scenario.

> **Important:**
> The Google search feature will not work locally, it must be deployed on an HTTP server.

Replacing the Search Engine Only

It is possible to replace the internal search engine that is used by **Oxygen XML WebHelp** by using a custom JavaScript file. This customization method allows you to replace the search engine but keep the search results presentation.

To replace WebHelp's internal search engine, follow this procedure:

1. If you have not already created a Publishing Template, see How to Create a Publishing Template *(on page 1464)*.
2. Create the following items in the folder that contains your publishing descriptor file (the **.opt** file):

- A folder named `js`.
- A folder named `fragments`.

3. In the `js` folder, create a file named `search-engine.js`.
4. As a starting point, you can copy the following content to the `search-engine.js` file:

```
/**
 * Object that implements the methods required by WebHelp to run a search engine.
 */
function CustomSearchEngine() {


  /**
   * Method required to run the search engine in webhelp. Handler when the users
   * executes the query in the search page.
   *
   * @param {String} query            The search input string from the user.
   * @param {Function} successHandler Needs to be called if the search operation is executed
   *                                  successfully. The parameter needs to have the type of
   *                                  WebHelpAPI.SearchResult
   * @param {Function} errorHandler   Needs to be called if the search operation fails to
   *                                  execute successfully. It needs to have the type
   *                                  of String.
   */
  this.performSearchOperation = function(query, successHandler, errorHandler) {
      // implement search engine
      // const searchRestult = externalSearchEngine(query);


      // convert the result to WebHelpApi.SearchResult
      // const formattedResult = convert(searchRestult);


      // call successHanlder with the converted result.
      // successHandler(formattedResult)
  }


  /**
   * Method required to run the search engine in webhelp. Handler when the
   * page is changed in the search page.
   *
   * @param {Integer} pageToShow      The page to be dispalyed.
   * @param {Integer} maxItemsPerPage The maximum # of items that can be displayed on a page.
   * @param {String} query            The search input string from the user.
   * @param {Function} successHandler Needs to be called if the search operation is executed
   *                                  successfully. The parameter needs to have the type of
   *                                  WebHelpAPI.SearchResult
```

```
   * @param {Function} errorHandler   Needs to be called if the search operation fails to
   *                                  execute successfully. It needs to have the type
   *                                  of String.
   */
  this.onPageChangedHandler = function(pageToShow, maxItemsPerPage, query, successHandler,
errorHandler) {
      // implement search engine
      // const searchRestult = externalSearchEngine(pageToShot, maxItemsPerPage, query);

      // convert the result to WebHelpApi.SearchResult
      // const formattedResult = convert(searchRestult);

      // call successHanlder with the converted result.
      // successHandler(formattedResult)
  }
}

// Set the Search Engine to WebHelp
WebHelpAPI.setCustomSearchEngine(new CustomSearchEngine());
```

Note:
See the API Search Objects section *(on page 1368)* for details on how to convert your custom search engine results to `WebHelpAPI.SearchResult`.

5. Implement your search engine.
6. In the `fragments` folder, create a file named `search-engine-script-fragment.xml`.
7. In the `search-engine-script-fragment.xml` file, define the scripts that are required for your search engine to run. For example:

```
<div>
    <script src="${oxygen-webhelp-template-dir}/js/search-engine.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 1274)* for more details):

```
<fileset>
  <include name="js/**"/>
</fileset>
```

9. Set the transformation parameters needed to enable the search filter. For this, open the `.opt` file and add the following content inside the `<webhelp>` element:

```
<html-fragments>
  <fragment file="fragments/search-engine-script-fragment.xml"
```

```
      placeholder="webhelp.fragment.head.search.page"/>
</html-fragments>
```

API Search Objects

To replace the WebHelp Search Engine, you will need to convert your custom search result into WebHelp API Objects that WebHelp will use to render your search result on the search page. To convert your custom search result, you will have to create the following objects:

1. *WebHelpAPI.SearchMeta* is a JavaScript object used to hold additional information for the search result. To create such an object, the following fields are required:
    - **String: searchEngineName** - The name of the search engine used to retrieve the search result.
    - **Integer: totalSearchItems** - The total number of search items the search engine returned.
    - **Integer: currentPage** - The current page to display.
    - **Integer: maxItemsPerPage** - The maximum number of items that can be displayed on a page.
    - **Integer: totalPages** - The number of total pages for the search result.
    - **String: originalSearchExpression** - The query string the user typed in the search input field.

```
conse searchMeta = new WebHelpAPI.SearchMeta(searchEngineName, totalSearchItems, currentPage,
maxItemsPerPage, totalPages, origianlSearchExpresion);
```

2. *WebHelpAPI.SearchDocument* is a JavaScript object used to hold the search result for a single topic/HTML page. To create such an object, the following fields are required:
    - **String: linkLocation** - The URL to the topic.
    - **String: title** - The topic title.
    - **String: shortDescription** - The topic short description.

```
const searchDocument = new WebHelpAPI.SearchDocument(linkLocation, title, shortDescription);
```

3. *WebHelpAPI.SearchResult* is a JavaScript object used to display the search results in the search page. To create such an object, the following fields are required:
    - **WebHelpAPI.SearchMeta: searchMeta** - Contains additional information for the search result.
    - **Array[WebHelpAPI.SearchDocument]: documents** - An array with the matching documents (HTML pages) for the search result.

```
conse searchMeta = new WebHelpAPI.SearchMeta(searchEngineName, totalSearchItems, currentPage,
maxItemsPerPage, totalPages, origianlSearchExpresion);
const searchDocument = new WebHelpAPI.SearchDocument(linkLocation, title, shortDescription);
const documents = [searchDocument]; // An array with one element.
const searchResult = new WebHelpAPI.SearchResult(searchMeta, documents);
```

Replacing the Search Engine and Results Presentation

It is possible to integrate a custom search engine and replace the search results area 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 1464)*.
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:

```
<div id="cumstom-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:

```
document.addEventListener('DOMContentLoaded', (event) => {
    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:

```
<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 1274)* for more details):

```
<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:

```
<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.

Tip:
A sample publishing template that overrides WebHelp's default search engine is available to download here. You can use it as a starting point for your customization.

How to Display Custom Title in Search Results

It is possible to display a custom title for topics in the search results page. This can be achieved by adding the `<searchtitle>` element inside the particular DITA topic (or within the topic reference in the DITA map). The `<searchtitle>` element is used to specify the title that is displayed by search tools that locate the topic. This is useful when the topic has a title that makes sense in the context of a single information set, but may be too general in a list of search results. If the `<searchtitle>` is specified, then the search results page will display the contents inside the `<searchtitle>` as the topic title.

For details about the `<searchtitle>` element (including an example), see https://docs.oasis-open.org/dita/v1.2/os/spec/langref/searchtitle.html.

How to Trigger a Search Query When WebHelp is Loaded

You can use the `searchQuery` URL parameter to perform a search operation when WebHelp is loaded. This opens the search results page with the specified search query processed. The URL should look something like this:

```
http://localhost/webhelp/search.html?searchQuery=deploying%20feedback
```

Configuring the Search Engine Optimization

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:

```
<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>
  <url>
    <loc>http://www.example.com/topics/care.html#care</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 Developer 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.

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 Developer 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 1356)*

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:

```
<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:

```
<langlist>
    <lang xml:lang="en"    filename="strings-en-us.xml"/>
    <lang xml:lang="en-us"    filename="strings-en-us.xml"/>
</langlist>
```

4. Create an empty `strings-en-us.xml` file and copy the strings you want to change from the translation files *(on page 1373)* into it. Make sure you leave the `@id` attribute unchanged because this is the key used to look up the string. Then replace their values:

```
<variables>
  <variable id="Figure">Fig</variable>
  <variable id="Draft comment">ADDRESS THIS DRAFT COMMENT</variable>
</variables>
```

5. Use the **Integrate/Install DITA-OT Plugins** transformation scenario *(on page 1096)* found in the **DITA Map** section in the **Configure Transformation Scenario(s)** dialog box *(on page 1210)*.

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:

```
<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:

```
<langlist>

    <lang xml:lang="pl"    filename="strings-pl-pl.xml"/>

    <lang xml:lang="pl-PL"    filename="strings-pl-pl.xml"/>

</langlist>
```

4. Copy the WebHelp strings file (*`DITA-OT-DIR`*`/plugins/com.oxygenxml.webhelp.responsive/oxygen-webhelp/resources/localization/strings-en-us.xml`) to your plugin directory, and rename it as `strings-pl-pl.xml`.
5. Copy the common DITA-OT strings defined in the *`DITA-OT-DIR`*`/plugins/org.dita.base/xsl/common/strings-en-us.xml` file into the file created at step 4. It defines a set generated text available for HTML-based transformations (such as `<note>`, `<fig>`, and `<table>` elements). Translate the content of each `<variable>` element (make sure to leave the `@id` attribute unchanged).

```
<variables>

  ...

  <variable id="webhelp.content" js="true" php="false">Polish word for 'Content'.</variable>

  <variable id="webhelp.search" js="true" php="false">Polish word for 'Search'</variable>

  ...

  <variable id="Figure">Polish word for 'Figure'</variable>

  <variable id="Table">Polish word for 'Table'</variable>

  ...

</variables>
```

6. Use the **Integrate/Install DITA-OT Plugins** transformation scenario *(on page 1096)* found in the **DITA Map** section in the **Configure Transformation Scenario(s)** dialog box *(on page 1210)*.

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 2304)* 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 1267)*, 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:

```
<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 1271)* associated with your publishing template.

5. Use one of the parameters that begin with *webhelp.fragment (on page 1278)* 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.

```
<publishing-template>

    ...

    <webhelp>

        ...

      <html-fragments>

        <fragment

          file="HTML-fragments/facebook-widget.xml"

          placeholder="webhelp.fragment.after.toc_or_tiles"/>

      </html-fragments>

    </webhelp>
```

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:

```
<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. 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 1278)*. 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 X™ (formerly known as 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 X™ *Tweet* widget to your WebHelp Responsive output using an *Oxygen Publishing Template (on page 1267)*, 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:

```
<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 1271)* associated with your publishing template.
5. Use one of the parameters that begin with *webhelp.fragment (on page 1278)* 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.

```
<publishing-template>
    ...
    <webhelp>
        ...
      <html-fragments>
        <fragment
          file="HTML-fragments/tweet-button.xml"
          placeholder="webhelp.fragment.after.toc_or_tiles"/>
      </html-fragments>
    </webhelp>
```

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 X™ *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:

```
<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. 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 1278)*. 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 1267)*, 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 1271)* associated with your publishing template.
6. Use the webhelp.fragment.after.body parameter *(on page 1422)* 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"/>

      </html-fragments>

    </webhelp>
```

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:

```
<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 1422)* 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.

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.

whr-sitemap-post

Runs a custom Ant target after the `whr-sitemap` processing stage.

whr-copy-resources-pre

Runs a custom Ant target before the `whr-copy-resources` processing stage.

whr-copy-resources-post

Runs a custom Ant target after the `whr-copy-resources` processing stage.

whr-create-topic-pages-pre

Runs a custom Ant target before the `whr-create-topic-pages` processing stage.

whr-create-topic-pages-post

Runs a custom Ant target after the `whr-create-topic-pages` processing stage.

whr-create-main-page-pre

Runs a custom Ant target before the `whr-create-main-page` processing stage.

whr-create-main-page-post

Runs a custom Ant target after the `whr-create-main-page` processing stage.

whr-create-search-page-pre

Runs a custom Ant target before the `whr-create-search-page` processing stage.

whr-create-search-page-post

Runs a custom Ant target after the `whr-create-search-page` processing stage.

whr-create-indexterms-page-pre

Runs a custom Ant target before the `whr-create-indexterms-page` processing stage.

whr-create-indexterms-page-post

Runs a custom Ant target after the `whr-create-indexterms-page` processing stage.

whr-search-index-pre

Runs a custom Ant target before the `whr-search-index` processing stage.

whr-search-index-post

Runs a custom Ant target after the `whr-search-index` 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:

```
<plugin id="com.oxygenxml.webhelp.responsive.custom.ant.extensions">

     <require plugin="com.oxygenxml.webhelp.responsive"/>

    <feature extension="ant.import" file="custom_build_file.xml"/>

</plugin>
```

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:

```
<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 *(on page 2305)*, run one of the following scripts, depending on your operating system:
   - Windows: `DITA-OT-DIR/bin/dita.bat --install`
   - Linux/macOS: `sh DITA-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 2306)*.

  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.

Related information

WebHelp Responsive XSLT-Import and XSLT-Parameter Extension Points *(on page 1435)*

How to Use XSLT Extension Points from a Publishing Template

It is possible to declare a WebHelp XSLT-Import Extension Points *(on page 1436)* in an *Oxygen Publishing Template (on page 1457)* to override or add additional XSLT processing.

Example:

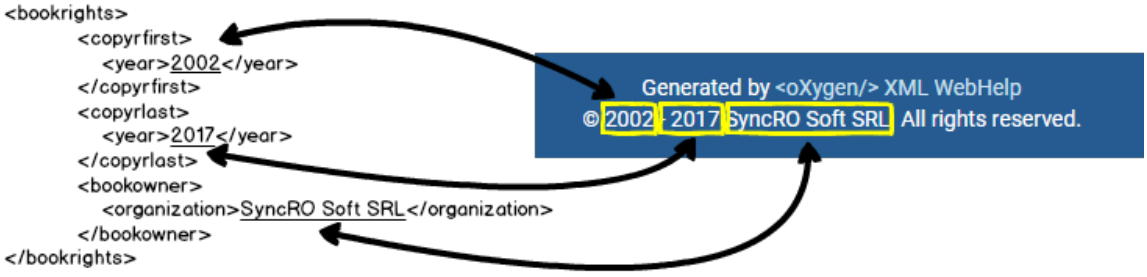
```
<publishing-template>
    ...
    <webhelp>
        ...
        <xslt>
            <extension
              id="com.oxygenxml.webhelp.xsl.createMainPage"
              file="xsl/customMainPage.xsl"/>
        </xslt>
```

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:

```
<bookrights>
  <copyrfirst>
    <year>2002</year>
  </copyrfirst>
  <copyrlast>
    <year>2017</year>
  </copyrlast>
  <bookowner>
    <organization>SyncRO Soft SRL</organization>
  </bookowner>
</bookrights>
```

Figure 401. 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 1288)* to expand its components. The main page template declares a component for the footer section that looks like this:

```
<div class=" footer-container text-center ">
  <whc:include_html href="${webhelp.fragment.footer}"/>
</div>
```

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 1464)*.
2. Link the folder associated with the publishing template to your current project in the **Project** view.

   Step Result: You should have the `custom_footer_template` folder linked in your project.
3. Using the **Project** view, create an `xsl` folder inside the project root folder.

   Step Result: You should have the `custom_footer_template/xsl` folder in your project.
4. Create your customization stylesheet (for example, **customMainPage.xsl**) in the `custom_footer_template/xsl` folder. Edit it to override the template that produces the footer section:

```
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns:oxygen="http://www.oxygenxml.com/functions"
  exclude-result-prefixes="xs"
  version="3.0">

  <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">
            &#xa9;<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($toc/*:topicmeta/*:bookrights/*:copyrfirst)">
          <span class="copyright_years">
            &#xa9;<xsl:value-of select="$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>


</xsl:stylesheet>
```

5. Open the template descriptor file *(on page 1271)* 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.

```
<publishing-template>
    ...
    <webhelp>
        ...
        <xslt>
            <extension
            file="xslt/customMainPage.xsl"
            id="com.oxygenxml.webhelp.xsl.createMainPage"/>
```

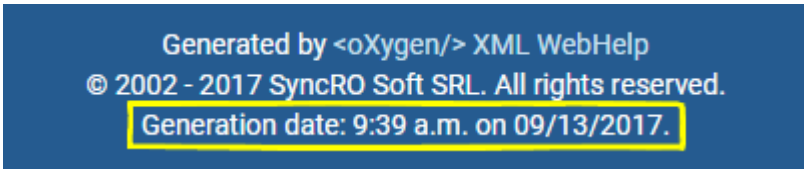
6. Open the *DITA Map WebHelp Responsive* transformation scenario.
7. Click the **Choose Custom Publishing Template** link and select your template.
8. Click **OK** to save the changes to the transformation scenario.
9. Run the transformation scenario.

Bonus: Add Generation Time in the Output Footer

An additional 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.

Note:
This can also be done directly by using an HTML fragment *(on page 1278)*. You can see a sample project on GitHub that uses an HTML fragment to show the generation time.

Figure 402. Generation Time Added in the WebHelp Footer



To add this functionality, follow these steps:

1. In the customization stylesheet that you just created (for example, **customMainPage.xsl**), modify the template by adding the following XSLT code at the end (before the closing `</div>` element).

```
<xsl:if test="oxygen: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:
   The `oxygen:getParameter(param.name)` function can be called to get the value of any WebHelp transformation parameter. Just make sure the `xmlns:oxygen="http://www.oxygenxml.com/functions"` namespace is declared in the stylesheet.

2. Open the template descriptor file *(on page 1271)* associated with your publishing template and set the `webhelp.footer.add.generation.time` parameter to the default value.

```
<publishing-template>
    ...
    <webhelp>
      ...
      <parameters>
        <parameter
          name="webhelp.footer.add.generation.time"
          value="yes"/>
```

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.

Use Case 2: Display Footnotes Below Tables

Suppose you want to separate the footnotes displayed in the WebHelp Responsive pages so that all footnotes are displayed at the end of the page except those within tables that are displayed just after them.

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 1464)*.
2. Link the folder associated with the publishing template to your current project in the **Project** view.

   Step Result: You should have the `custom_template` folder linked in your project.
3. Using the **Project** view, create an `xsl` folder inside the project root folder.

   Step Result: You should have the `custom_template/xsl` folder in your project.
4. Create your customization stylesheet (for example, **tablesFootnotes.xsl**) in the `custom_template/xsl` folder with the following content:

```
<?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"
  xmlns:oxy="http://www.oxygenxml.com/extensions/author"
  exclude-result-prefixes="#all"
  version="3.0">

  <!--
    Match only top level tables (i.e tables that are not nested in other tables),
    that contains some footnotes.
  -->
  <xsl:template match="
      *[contains(@class, 'topic/table')]
      [not(ancestor::*[contains(@class, 'topic/table')])]
      [//*[contains(@class, 'topic/fn')]]">
    <xsl:variable name="top-level-table" select="."/>
    <xsl:next-match>
      <xsl:with-param name="top-level-table" select="$top-level-table" tunnel="yes"/>
    </xsl:next-match>
    <!-- Create a list with all the footnotes from the current table. -->
    <div outputclass="table-fn-container">
      <xsl:for-each select=".//*[contains(@class, 'topic/fn')]">
        <!--
          Try to preserve the footnote ID, if available, so that the xrefs will have a target.
```

```
      -->
      <div id="{if(@id) then @id else generate-id(.)}">
        <xsl:variable name="unique-id" select="generate-id($top-level-table)"/>
        <xsl:variable name="fn-id">
          <xsl:number from="$top-level-table" level="any"/>
        </xsl:variable>

        <xsl:call-template name="commonattributes"/>
        <a name="tbl_{$unique-id}_fntarg_{$fn-id}" href="#tbl_{$unique-id}_fnsrc_{$fn-id}">
          <sup>
            <xsl:value-of select="$fn-id"/>
          </sup>
        </a>
        <xsl:text>  </xsl:text>

        <xsl:apply-templates/>
      </div>
    </xsl:for-each>
  </div>
</xsl:template>

<!--
  Process footnotes both inside and outside tables (based on the top-level-table parameter).
-->
<xsl:template match="*[contains(@class, ' topic/fn ')]" name="topic.fn">
  <xsl:param name="xref"/>
  <xsl:param name="top-level-table" tunnel="yes"/>
  <!-- Footnotes with IDs must be ignored, they are accessible only through xrefs. -->
  <xsl:if test="not(@id) or $xref = 'yes'">
    <xsl:variable name="unique-id" select="
        if ($top-level-table) then
          generate-id($top-level-table)
        else
          ()"/>
    <xsl:variable name="fn-id" select="
        if ($top-level-table) then
          index-of($top-level-table//*[contains(@class, 'topic/fn')], .)
        else
          index-of(//*[contains(@class, 'topic/fn')]
          [not(ancestor::*[contains(@class, 'topic/entry')])], .)"/>
    <xsl:variable name="callout" select="@callout"/>
    <xsl:variable name="converged-callout" select="
```

```
        if (string-length($callout) > 0) then
          $callout
        else
          $fn-id"/>
    <a>
      <!-- Generate different attributes based on the presence of a parent table. -->
      <xsl:attribute name="name" select="
          if ($top-level-table) then
            concat('tbl_', $unique-id,'_fnsrc_', $fn-id)
          else
            concat('fnsrc_', $fn-id)"/>
      <xsl:attribute name="href" select="
          if ($top-level-table) then
            concat('#tbl_', $unique-id,'_fntarg_', $fn-id)
          else
            concat('#fntarg_', $fn-id)"/>
      <sup>
        <xsl:value-of select="$converged-callout"/>
      </sup>
    </a>
  </xsl:if>
</xsl:template>

<!--
  The xrefs to footnotes with IDs inside table-cells. We need to recalculate
  their indexes if their referenced footnote is also in the table.
-->
<xsl:template match="
    *[contains(@class, 'topic/xref')][@type = 'fn']
    [ancestor::*[contains(@class, 'topic/entry')]]">
  <xsl:param name="top-level-table" tunnel="yes"/>

  <xsl:variable name="topic-id" select="substring-after(@href, '#')"/>
  <xsl:variable name="element-id" select="substring-after($topic-id, '/')"/>
  <xsl:variable name="destination">
    <xsl:choose>
      <xsl:when test="$element-id = ''">
        <xsl:value-of select="$topic-id"/>
      </xsl:when>
      <xsl:otherwise>
        <xsl:value-of select="$element-id"/>
      </xsl:otherwise>
```

```
    </xsl:choose>
  </xsl:variable>

  <xsl:variable name="fn" select="
      $top-level-table//*[contains(@class, 'topic/fn')][@id = $destination]"/>
  <xsl:choose>
    <xsl:when test="$fn">
      <xsl:variable name="unique-id" select="generate-id($top-level-table)"/>
      <!-- There is a reference in the table, recalculate index. -->
      <xsl:variable name="fn-id" select="
          index-of($top-level-table//*[contains(@class, 'topic/fn')], $fn)"/>
      <a name="tbl_{$unique-id}_fnsrc_{$fn-id}" href="#tbl_{$unique-id}_fntarg_{$fn-id}">
        <sup>
          <xsl:value-of select="$fn-id"/>
        </sup>
      </a>
    </xsl:when>
    <xsl:otherwise>
      <!-- There is no reference in the table, keep original index. -->
      <xsl:next-match/>
    </xsl:otherwise>
  </xsl:choose>
</xsl:template>

<!--
  Output footnotes below tables and at the end of the topic.
-->
<xsl:template match="*[contains(@class, ' topic/fn ')]" mode="genEndnote">
  <!-- Do not process footnotes from tables. -->
  <xsl:if test="not(node()/ancestor::*[contains(@class, 'topic/table')])">
    <div>
      <!-- Do not number footnotes from tables. -->
      <xsl:variable name="fn-id" select="index-of(//*[contains(@class, 'topic/fn')]
        [not(ancestor::*[contains(@class, 'topic/entry')])], .)"/>
      <xsl:variable name="callout" select="@callout"/>
      <xsl:variable name="converged-callout" select="
          if (string-length($callout) > 0) then
            $callout
          else
            $fn-id"/>

      <xsl:call-template name="commonattributes"/>
```

```
      <xsl:choose>
        <xsl:when test="@id and not(@id = '')">
          <xsl:variable name="topic-id" select="
            ancestor::*[contains(@class, ' topic/topic ')][1]/@id"/>
          <xsl:variable name="ref-id" select="concat($topic-id, '/', @id)"/>
          <xsl:choose>
            <xsl:when test="key('xref', $ref-id)">
              <a>
                <xsl:call-template name="setid"/>
                <sup>
                  <xsl:value-of select="$converged-callout"/>
                </sup>
              </a>
              <xsl:text>  </xsl:text>
            </xsl:when>
            <xsl:otherwise>
              <sup>
                <xsl:value-of select="$converged-callout"/>
              </sup>
              <xsl:text>  </xsl:text>
            </xsl:otherwise>
          </xsl:choose>
        </xsl:when>
        <xsl:otherwise>
          <a name="fntarg_{$fn-id}" href="#fnsrc_{$fn-id}">
            <sup>
              <xsl:value-of select="$converged-callout"/>
            </sup>
          </a>
          <xsl:text>  </xsl:text>
        </xsl:otherwise>
      </xsl:choose>

      <xsl:apply-templates/>
    </div>
  </xsl:if>
 </xsl:template>

</xsl:stylesheet>
```

5. Open the template descriptor file *(on page 1271)* associated with your publishing template and set the XSLT stylesheet created in the previous step with the *com.oxygenxml.webhelp.xsl.dita2webhelp* XSLT extension point.

```
<publishing-template>
    ...
    <webhelp>
        ...
        <xslt>
            <extension
            file="xslt/tablesFootnotes.xsl"
            id="com.oxygenxml.webhelp.xsl.dita2webhelp"/>
```

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.

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.



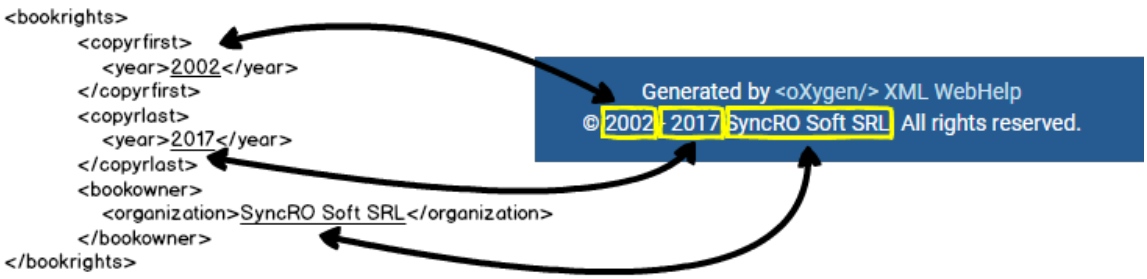
Note:
This customization is available as a GitHub project at: https://github.com/oxygenxml/com.oxygenxml.webhelp.responsive.custom.footer.

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:

```
<bookrights>
  <copyrfirst>
    <year>2002</year>
  </copyrfirst>
  <copyrlast>
    <year>2017</year>
  </copyrlast>
  <bookowner>
    <organization>SyncRO Soft SRL</organization>
  </bookowner>
</bookrights>
```

Figure 403. 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 1288)* declares a component for the footer section that looks like this:

```
<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.

```
<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">
                 &#xa9;<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($toc/*:topicmeta/*:bookrights/*:copyrfirst)">
              <span class="copyright_years">
                 &#xa9;<xsl:value-of select="$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>
```

You can implement this functionality with a WebHelp extension plugin that uses the **com.oxygenxml.webhelp.xsl.createMainPage** extension point *(on page 1436)*. 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:

   ```
   <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:

   ```
   <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">
                    &#xa9;<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($toc/*:topicmeta/*:bookrights/*:copyrfirst)">
            <span class="copyright_years">
               &#xa9;<xsl:value-of select="$toc/*:topicmeta/*:bookrights/*:copyrfirst"/>
            </span>
        </xsl:when>
    </xsl:choose>



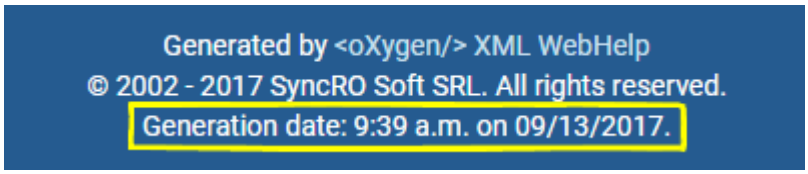
    <xsl:if test="exists($toc/*:topicmeta/*:bookrights/*:bookowner/*:organization)">
        <span class="organization">
            <xsl:text> </xsl:text><xsl:value-of
                select="$toc/*:topicmeta/*:bookrights/*:bookowner/*:organization"/>
            <xsl:text>. All rights reserved.</xsl:text>
        </span>
    </xsl:if>
  </div>
</xsl:template>
```

4. Use the **Integrate/Install DITA-OT Plugins** transformation scenario *(on page 1096)* found in the **DITA Map** section in the **Configure Transformation Scenario(s)** dialog box *(on page 1210)*.
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 1437)*.

Figure 404. 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 1394)*.
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.

```
<xsl:if test="$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>
```

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:

```
<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:

```
<dummy>
    <param name="webhelp.footer.add.generation.time"
        expression="${webhelp.footer.add.generation.time}"
        if="webhelp.footer.add.generation.time"/>
</dummy>
```

5. Use the **Integrate/Install DITA-OT Plugins** transformation scenario *(on page 1096)* found in the **DITA Map** section in the **Configure Transformation Scenario(s)** dialog box *(on page 1210)*.
6. Edit a **DITA Map WebHelp Responsive** transformation scenario and in the **Parameters** tab *(on page)*, 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 2306)*, the `webhelp.custom.resources` parameter or by referencing it in the DITA Map .

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 1464)*.
2. Add a new `<fileset>` element in the resources section of the template descriptor file *(on page 1274)*.

```
<publishing-template>
    ...
    <webhelp>
        ...
        <resources>
            <fileset>
                <include name="custom-resources/**/*"/>
                <exclude name="**/*.git"/>
            </fileset>
```

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.

Copying Additional Resources to the Output Directory using DITA Map reference

1. Place the resources in the ditamap directory or in a folder under it.
2. Open your ditamap file and declare the resource to be copied:

```
<topicref href="path/to/manual.pdf" format="pdf" processing-role="resource-only"/>
```

3. Run the transformation scenario.

Result: All files referenced this way will be copied at the path indicated by `@href` in the ditamap.

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 1267)*, follow this procedure:

1. If you have not already created a Publishing Template, see Working with Publishing Templates *(on page 1308)*.
2. Open the template descriptor file *(on page 1271)* 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 or Content Fusion 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*).

```
<publishing-template>
    ...
    <webhelp>
        ...
        <parameters>
            <parameter name="editlink.ditamap.edit.url"
                       value="webdav-https://dav.box.com/dav/my.ditamap"/>
        </parameters>
    </webhelp>
```

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`dita.bat -i c:\mySample.ditamap -f webhelp-responsive -Deditlink.ditamap.edit.url=webdav-https://dav.box.com/dav/my.ditamap`
- macOS/ Linux`dita -i /mySample.ditamap -f webhelp-responsive -Deditlink.ditamap.edit.url=webdav-https://dav.box.com/dav/my.ditamap`

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

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 2303)* (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 1267)*, 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 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 1271)* 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.

```
<publishing-template>
    ...
    <webhelp>
        ...
        <parameters>
            <parameter name="args.filter" value="resources/myFile.ditaval"/>
        </parameters>
    </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.

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 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:

```
<script type="text/javascript"
  src="https://cdnjs.cloudflare.com/ajax/libs/mathjax/2.7.1/MathJax.js?config=TeX-AMS-MML_HTMLorMML">
</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.
   - **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 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 Responsive output, follow this procedure:

1. Create a new well-formed XML file and add the following code snippet:

```
<meta http-equiv="Pragma" content="no-cache" />
<meta http-equiv="Expires" content="-1" />
```



Note:
The code should look like this:

```
<!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 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.

Important:
The external PDF must already be created beforehand. This procedure does not generate the PDF, just the component that links to a specified PDF resource.

Add a Link to a PDF Using a Publishing Template

To embed a component that links to an external PDF resource in the DITA Map WebHelp Responsive output using an *Oxygen Publishing Template (on page 1267)*, follow this procedure:

1. If you have not already created a Publishing Template, see Working with Publishing Templates *(on page 1308)*.
2. Open the template descriptor file *(on page 1271)* associated with your publishing template and add the following parameters:

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.named.destination.enabled

Specifies whether or not the current topic ID should be appended as named destination at the end of the PDF link. Allowed values are: **yes** (default) and **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.

Result: In the WebHelp output, the link component is added in the header/breadcrumb stripe, next to the navigation links.

Add a Link to a PDF by Directly Setting Parameters in a Transformation Scenario

To embed a component that links to an external PDF resource 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:

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.named.destination.enabled

Specifies whether or not the current topic ID should be appended as named destination at the end of the PDF link. Allowed values are: **yes** (default) and **no**.

3. Run the transformation scenario.

Result: In the WebHelp output, the link component is added in the header/breadcrumb stripe, next to the navigation links.

Related information

How to Copy Additional Resources to Output Directory *(on page 1398)*

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 1287)* 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 1222)*.

For example, the page component that is used to define the Search Input field in the WebHelp HTML pages is defined as follows:

```
<!-- 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:

```
<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 1332)* to change the default layout and styles.
- Insert additional HTML content *(on page 1334)* using one of the available HTML Fragment Placeholder parameters *(on page 1278)*.
- Extend the default processing using XSLT Extension Points *(on page 1277)*.
- Configure available Transformation Parameters *(on page 1420)*.

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:

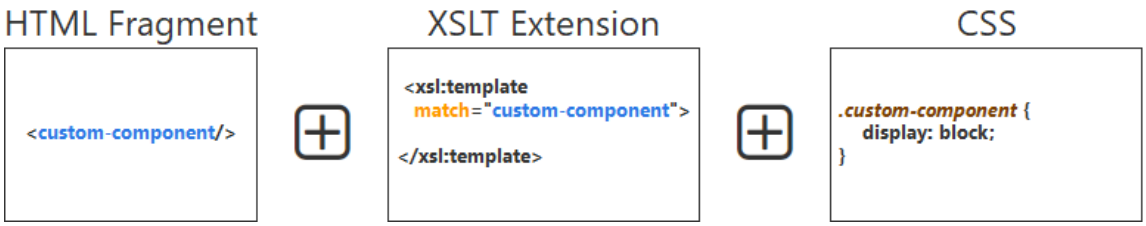
```
<prolog>
   <metadata>
      <othermeta name="video-link" content="https://www.youtube.com/watch?v=zNmXfKWXwO8"/>
   </metadata>
</prolog>
```

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 405. Custom Component



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:

   ```
   <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 1271)* and associate it with an HTML Fragment placeholder that is available for the topic pages (`webhelp.fragment.before.topic.toolbar` in this case):

```
<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:

```
<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:

```
<div class="video-link-container">
    <a href="https://www.youtube.com/watch?v=zNmXfKWXwO8"
      class="video-link" target="_blank"
        aria-label="Video">
        <span>Video</span>
    </a>
</div>
```

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:

```
<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:

```
@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;
}
```

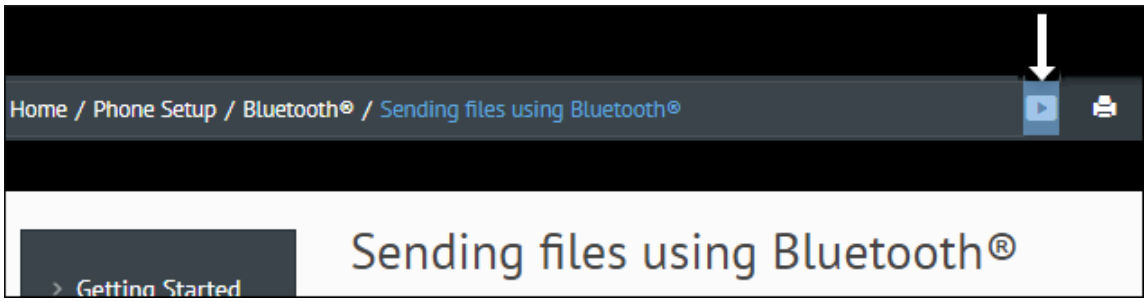
```
.wh_right_tools {
    padding: 0;
}
```

7. Reference the above CSS file in your Publishing Template's descriptor file:

```
<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 406. Custom Link to Video Component



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.

How to Generate Google Structured Data

It is possible to generate Google Structured Data (`<script>` elements that contain a JSON-LD object) in the DITA WebHelp Responsive output. Google uses this JSON-LD object to better understand the contents of the page and display special search results in a Google Search.

Tip:
For more details, see Google Search Central: Understand how structured data works.

To generate Google Structured Data in WebHelp output, use the following transformation parameter:

google.structured.data

Specifies whether or not Google Structured Data will be generated in the output. If set to **yes**, the transformation automatically generates Google Structured Data for **Questions and Answers** topics, DITA Task topics, and from `<data>` elements found inside a topic that has the `@name="oxy:question"` construct. If set to **no** (default value), the transformation will not generate Google Structured Data.

Generating Google Structured Data for DITA Tasks Topics

When Google Structured Data is enabled, the DITA Task `<title>`, `<shordesc>`, and `<step>` elements are mapped to the HowTo JSON-LD object. For example, the following DITA Task topic:

```
<task id="task_id">
  <title>My task</title>
  <shortdesc>Task description</shortdesc>
  <steps>
    <step>
      <cmd>Step 1 content.</cmd>
    </step>
    <step>
      <cmd>Step 2 content.</cmd>
    </step>
  </steps>
</task>
```

will generate the following structure in the output:

```
<script type="application/ld+json" id="jsonld-howto">
  {
    "@context": "https://schema.org",
    "@type": "HowTo",
    "name": "My task",
    "description": "Task description",
    "supply": [],
    "tool": [],
    "step":[
      {
        "@type": "HowToStep",
        "text": "<span class=\"topic/ph task/cmd ph cmd\">Step 1 content.</span>"
      }
      ,
      {
        "@type": "HowToStep",
        "text": "<span class=\"topic/ph task/cmd ph cmd\">Step 2 content.</span>"
      }
    ]
  }
</script>
```

Generating for Questions and Answers Topics

When Google Structured Data is enabled, the QA topic `<qagroup>` elements are mapped to the FAQPage JSON-LD object. For example, the following QA topic:

```
<qatopic id="qa_id">
  <title>Faq Page 1</title>
  <qabody>
    <qagroup>
      <question>What is a car engine?</question>
      <answer>The car engine is a device that uses fuel to create mechanical power that can
              turn the car's wheels.</answer>
    </qagroup>
  </qabody>
</qatopic>
```

will generate the following structure in the output:

```
<script type="application/ld+json" id="jsonld-faq">
  {
    "@context": "https://schema.org",
    "@type": "FAQPage",
    "mainEntity": [
      {
        "@type": "Question",
        "name": "What is a car engine?",
        "acceptedAnswer": {
          "@type": "Answer",
          "text": "<div class=\"- topic/div qatopic/answer div answer\">The car engine is a
device that uses fuel to create mechanical power that can turn the car's wheels.</div>"
        }
      }
    ]
  }
</script>
```

Generating from `data` elements found inside a topic

When Google Structured Data is enabled, the WebHelp Responsive transformation will map the `<data>` elements found inside a topic to a FAQPage JSON-LD object. There are 2 different use cases depending on where the `<data>` element is found in the document:

- In the `<prolog>` element. For example, this content:

  ```
  <concept id="lawnmowerconcept">
    <title>Lawnmower</title>
  ```

```
  <shortdesc>The lawnmower is a machine used to cut grass in the yard.</shortdesc>
  <prolog>
    <metadata>
      <data name="oxy:question">What tools are necessary to cut the grass?</data>
    </metadata>
  </prolog>
  <conbody>
    <p>Lawnmowers can be electric, gas-powered, or manual.</p>
  </conbody>
</concept>
```

will generate the following structure in the output:

```
<script type="application/ld+json" id="jsonld-faq">
  {
    "@context": "https://schema.org",
    "@type": "FAQPage",
    "mainEntity": [
      {
        "@type": "Question",
        "name": "What tools are necessary to cut the grass?",
        "acceptedAnswer": {
          "@type": "Answer",
          "text": "<div class=\"- topic/body concept/conbody body conbody\">
                     <p class=\"- topic/shortdesc shortdesc\">The lawnmower is a machine
used to cut grass in the yard.</p> <p class=\"- topic/p p\">Lawnmowers can be electric,
gas-powered, or manual.</p> </div>"
        }
      }
    ]
  }
</script>
```

Important:

The answer represents the HTML result of the entire content inside the topic.

- Inside the topic body elements. For example, content:

```
<topic id="concept-id">
  <title>Morning</title>
  <shortdesc>In the morning we have breakfast.</shortdesc>
  <body>
    <ul>
      <data name="oxy:question">What do people drink in the morning?</data>
```

```
      <li>Tea</li>
      <li>Milk</li>
    </ul>
  </body>
</topic>
```

will generate the following structure in the output:

```
<script type="application/ld+json" id="jsonld-faq">
  {
    "@context": "https://schema.org",
    "@type": "FAQPage",
    "mainEntity": [
      {
        "@type": "Question",
        "name": "What do people drink in the morning?",
        "acceptedAnswer": {
          "@type": "Answer",
          "text": "<div class=\"- topic/body body\"><ul class=\"- topic/ul ul\"></ul>
<li class=\"- topic/li li\">Tea</li> <li class=\"- topic/li li\">Milk</li> </div>"
        }
      }
    ]
  }
</script>
```

Important:
The answer represents the HTML result of the entire block where the `<data>` element is located inside.

How to Hide Related Link Descriptions

Starting with **Oxygen 27.0**, by default, if the target of a related link includes a short description (`<shortdesc>`), that description will appear as text beneath the link. It is possible to remove this text by adding the following rule in a custom CSS stylesheet:

```
.wh_related_links .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:

```
Related information

  Target Topic

  Target Concept

  Target Task
```

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:

```
Related concepts

  Target Concept

Related tasks

  Target Task

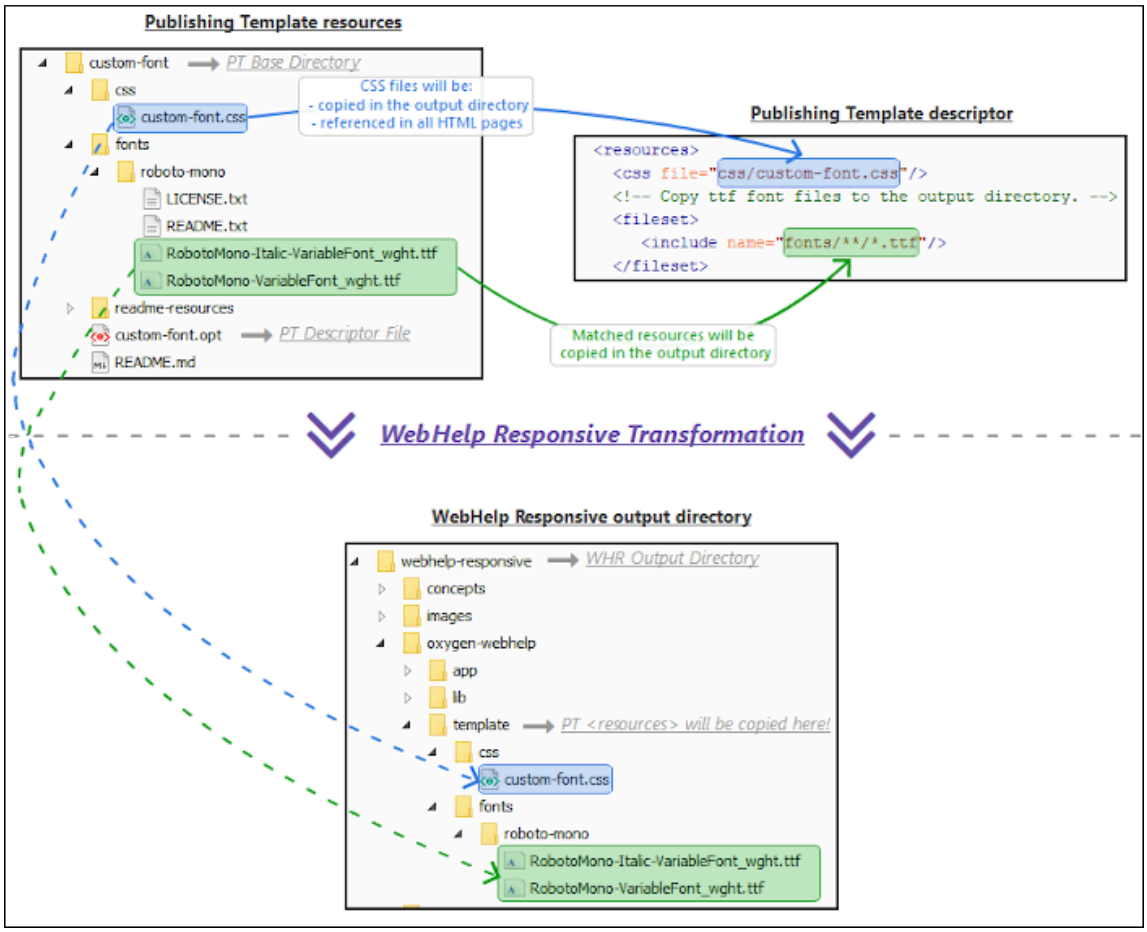
Related information

  Target Topic
```

How to Use a Local Font in WebHelp Responsive Output

It is possible to use a local fonts in WebHelp Responsive output by copying the local font file to the output directory through a Publishing Template and referencing the font files using `@font-face` rules within a custom CSS.

Figure 407. Referencing Local Fonts in a Publishing Template



To use a local font in 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 1464)*.
2. Add the local font files to the `fonts` folder within your Publishing Template directory structure. For example:

   ```
   fonts/roboto-mono/RobotoMono-Italic-VariableFont_wght.ttf

   fonts/roboto-mono/RobotoMono-VariableFont_wght.ttf
   ```

3. Configure WebHelp Responsive to copy the font file to the output directory. Define a `<fileset>` that matches the location of the font files in the `<resources>` section of your Publishing Template's descriptor file.

   ```
   <resources>
     <!-- Copy ttf font files to the output directory. -->
     <fileset>
        <include name="fonts/**/*.ttf"/>
     </fileset>
   </resources>
   ```

   All the files matched by this fileset will be copied to the output directory. The additional resources will be copied in the following subfolder of the output directory:

   ```
   {OUTPUT-DIR}/oxygen-webhelp/template/
   ```

4. Create a custom CSS file in your Publishing Template directory.

   ```
   css/custom-font.css
   ```

5. Reference the CSS file in the `<resources>` section of the Publishing Template's descriptor file. This means that the CSS file will be referenced in each HTML page within the WebHelp Responsive output.

   ```
   <resources>
     <css file="css/custom-font.css"/>
     <!-- ... -->
   </resources>
   ```

6. Add `@font-face` definitions that reference the font files in your custom CSS file. The font files can be referenced using relative URLs since the CSS and the font files included in the Publishing Template package will be copied together in the output folder.

   ```
   @font-face {
     font-family: 'Roboto Mono';
     font-style: normal;
     src: url('../fonts/roboto-mono/RobotoMono-VariableFont_wght.ttf') format('truetype');
   }
   @font-face {
     font-family: 'Roboto Mono';
     font-style: italic;
     src: url('../fonts/roboto-mono/RobotoMono-Italic-VariableFont_wght.ttf') format('truetype');
   }
   ```

7. Add a CSS rule that applies the custom font on all elements.

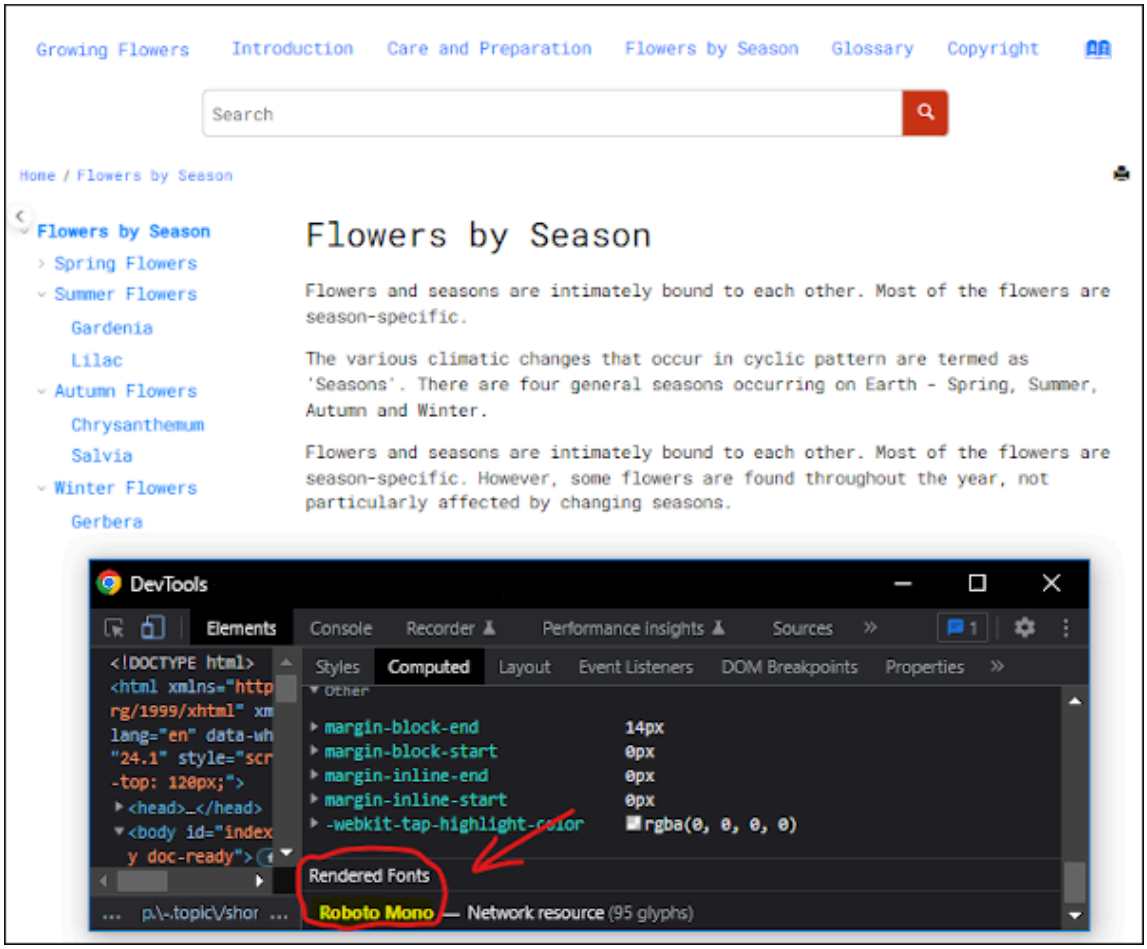
```
body {

  font-family: 'Roboto Mono', sans-serif;

}
```

8. Run the transformation with the publishing template selected.

Figure 408. Output Example



How to Use JQuery in WebHelp Responsive Output

The JQuery library that comes bundled with WebHelp is accessible in the browser's global context so that developers have access to use it.

To use the JQuery library in your WebHelp Responsive output, follow these steps:

1. If you have not already created a Publishing Template, see How to Create a Publishing Template.
2. Create the following items in the folder that contains your publishing template's descriptor file (the `.opt` file):
   - A folder named **js**
   - A folder named **fragments**
3. In the **js** folder, create a file named `custom.js`.
4. As a starting point, you can copy the following content to the `custom.js` file:

```
$(document).ready(function () {
    // Your JQuery code.
});
```

5. In the **fragments** folder, create a file named `jquery-scripts.html` with the following content:

```
<html>
    <script src="${oxygen-webhelp-template-dir}/js/custom.js" defer="defer"></script>
</html>
```

> **Important:**
> Make sure that the `@defer` attribute is present on the `<script>` element.

6. 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 for more details):

```
<fileset>
    ...
    <include name="js/**"/>
    ...
</fileset>
```

7. Include the `jquery-scripts.html` file in your WebHelp Responsive output by opening the `.opt` file and add the following content inside the `<webhelp>` element:

```
<html-fragments>
  <fragment file="jquery-scripts.html" placeholder="webhelp.fragment.head"/>
</html-fragments>
```

8. Run the transformation with your publishing template selected.

How to Display Certain Elements as Tabs

It is possible to display the following elements as tabs in the WebHelp output:

- Lists (both `<ol>` and `<ul>` elements)
- Definition lists (`<dl>` element)
- Choice tables (`<choicetable>` element)

For this, you just need to add the `@outputclass="wh-tabbed"` attribute on the element.

> **Note:**
> The tab name is extracted from:

- The first `<ph>` in lists.
- The first `<dt>` in definition lists.
- The first `<choption>` in choice tables.

Here are some examples:

- List

```
<ul outputclass="wh-tabbed">
    <li>
        <ph>Windows</ph>
        <ol>
            <li>First instruction for Windows</li>
            <li>Second instruction for Windows</li>
        </ol>
    </li>
    <li>
        <ph>Linux</ph>
        <ol>
            <li>First instruction for Linux</li>
            <li>Second instruction for Linux</li>
            <li>Third instruction for Linux</li>
        </ol>
    </li>
    <li>
        <ph>Mac</ph>
        <ol>
            <li>First instruction for Mac</li>
            <li>Second instruction for Mac</li>
        </ol>
    </li>
</ul>
```

- Definition List

```
<dl outputclass="wh-tabbed">
    <dlentry>
        <dt>Version 1</dt>
        <dd>Instructions for Version 1</dd>
    </dlentry>
    <dlentry>
        <dt>Version 2</dt>
        <dd>Instructions for Version 2</dd>
```

```
    </dlentry>
</dl>
```

- Choice Table

```
<choicetable outputclass="wh-tabbed">
    <chrow>
        <choption>First</choption>
        <chdesc>First option description</chdesc>
    </chrow>
    <chrow>
        <choption>Second</choption>
        <chdesc>Second option description</chdesc>
    </chrow>
</choicetable>
```

The outputs will be as follows:

- Windows
    1. First instruction for Windows
    2. Second instruction for Windows
- Linux
    1. First instruction for Linux
    2. Second instruction for Linux
    3. Third instruction for Linux
- Mac
    1. First instruction for Mac
    2. Second instruction for Mac

Version 1

Instructions for Version 1

Version 2

Instructions for Version 2

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.target.url.new.tab

Specifies if the URL of the link that is set on the logo image will be opened in a new tab.

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.

Context Sensitive Help Parameter

webhelp.csh.disable.topicID.fallback

Specifies whether or not topic ID *fallbacks* are enabled when computing the mapping of context sensitive help and `resourceid` information is not available. Possible values are **false** (default) and **true**.

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.enable.scroll.to.search.term

Specifies whether or not the page should scroll to the first search term when opening the search results page. Possible values are **no** (default) and **true**.

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 Developer (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.



Note:
All the XHTML fragments can contain WebHelp Responsive Macros *(on page 1336)*.

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.display.table.cell.tooltip

Specifies whether a tooltip indicating the row and column of each table cell should be shown. Possible values are **no** (default) or **yes**.

webhelp.enable.ai.friendly

Indicates if Markdown topics and an `llms.txt` file should be created alongside the standard HTML output. Possible values are **no** (default) or **yes**. The `llms.txt` file can be used together with companion Markdown (`.md`) files to provide LLMs with a clean, simplified view of site content and a curated index of the most relevant resources.

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.table.headers

Controls whether or not table headers will remain *sticky* in the output when the tables are taller than the viewport height. Possible values are **yes** or **no** (default).

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.full.page.width

Determines whether or not the content should use the entire page space. This is most beneficial for large displays such as 4K. Possible values are **no** (default) or **yes**.

webhelp.labels.generation.mode

Controls whether or not labels are generated in the output. These labels are useful because users can easily search for topics with the same label by simply clicking on the label presented in the output. Possible values are:

- **keywords-label** - Generates labels for each defined `<keyword>` element that has the `@outputclass` attribute value set to **label**.
- **keywords** - Generates labels for each defined `<keyword>` element. If the topic contains `<keyword>` elements with the `@outputclass` attribute value set to **label**, then only these elements will have labels generated for them in the output.
- **disable** - Disables the generation of labels in the Webhelp Responsive output.

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.image.map.areas

Indicates whether the image map areas are displayed permanently or only on hover. Possible values are **no** (default - areas visible on hover) or **yes** (areas always visible).

webhelp.show.indexterms.link

Specifies if an icon that links to the index terms page will be displayed in the output. The default value is `yes` (meaning the index terms icon is displayed). If set to `false`, the index terms icon is not displayed in the output and the index terms page is not generated.

webhelp.enable.dark.mode

Specifies if the dark mode is enabled in the output. The default value is `no` (no theme menu is displayed). If set to `yes`, a drop-down menu will appear, and the output will reflect either the browser's settings or the user's preference.

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.expand.collapse.sections

Specifies if links to collapse sections will be presented within each topic in the output. Collapsing sections will collapse all collapsible elements (nested topics with titles, sections with titles, or tables with titles). 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 `@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.skip.main.page.generation

If set to `true`, the default main page is not generated in the output. The default value is `false`.

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 (nested topics with titles, sections with titles, tables with titles or 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 `<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.default.operator

Makes it possible to change the default operator for the search engine. Possible values are `and`, `or` (default). If set to `and` while the search query is WORD1 WORD2, the search engine only returns results for topics that contain both WORD1 and WORD2. If set to `or` and the search query is WORD1 WORD2, the search engine returns results for topics that contain either WORD1 or WORD2.

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 1305)*
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 XSLT Extension Points from a Publishing Template *(on page 1385)*.

Example:

```
<publishing-template>
    ...
    <webhelp>
        ...
        <xslt>
            <extension
              id="com.oxygenxml.webhelp.xsl.createMainPage"
              file="xsl/customMainPage.xsl"/>
        </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 stylesheet referenced by the extension point becomes part of the normal build. You can use these extension points to override XSLT processing steps.

Example:

```
<plugin id="com.oxygenxml.webhelp.responsive.extension">
  <feature extension="com.oxygenxml.webhelp.xsl.dita2webhelp"
           file="xsl/fixup.xsl"/>
</plugin>
```

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.webhelp.responsive\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.webhelp.responsive\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.webhelp.responsive\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.webhelp.responsive\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 2304)* 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`.

com.oxygenxml.webhelp.xsl.contextHelpMap

Extension point to override the XSLT stylesheet (`contextHelpMapDita.xsl`) that generates the context sensitive help mapping. The path to this stylesheet is: *`DITA-OT-DIR`*`\plugins\com.oxygenxml.webhelp.responsive\xsl\contextHelp\contextHelpMapDita.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 1436)*.

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 1436)*.

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 1436)*.

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 1437)*.

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 1437)*.

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 1437)*.

com.oxygenxml.webhelp.xsl.contextHelpMap.param

Use this extension point to pass parameters to the stylesheet specified using the **com.oxygenxml.webhelp.xsl.contextHelpMap** extension point *(on page 1437)*.

Related Information:
[DITA-OT] *XSLT-Import* Extension Points
[DITA-OT] *XSLT-Parameter* Extension Points

DITA to PDF Output Customization

Oxygen XML Developer provides support for generating PDF output using transformation scenarios for certain types of documents (for example, DITA, DocBook, TEI, and JATS) and Oxygen XML Developer 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 Developer 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 1087)*
DITA PDF - based on HTML5 & CSS Transformation *(on page)*

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 DITA documents to PDF using CSS, Part 2 – Book Design, Pagination, Page Layout, and Bookmarks**.
- **Transforming DITA documents to PDF using CSS, Part 3 – Advanced Fonts Usage**.
- **Transforming DITA documents to PDF using CSS, Part 4 – Advanced CSS Rules**.
- **Transforming XML and HTML documents to PDF using CSS, Part 1 – Basic CSS Layout**.
- **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** - You can find a good tutorial here: https://developer.mozilla.org/en-US/docs/Learn/CSS/Introduction_to_CSS. Also, the specification is available on the W3C (https://www.w3.org/Style/CSS/Overview.en.html) or on the MDN (https://developer.mozilla.org/en-US/docs/Web/CSS) websites.
- **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. Also, there is a set of hands-on examples in the **Oxygen PDF Chemistry** user guide: 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 and: https://www.oxygenxml.com/dita/styleguide/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.
- **HTML5** - You will need a good knowledge of HTML5. You can find resources here: https://developer.mozilla.org/en-US/docs/Web/Guide/HTML/HTML5
- **Webinars** - Some helpful webinars are available on our website: https://www.oxygenxml.com/publishing_engine/videos.html?category=Webinars. They explain how to generate PDF from DITA documents using CSS, step-by-step.

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 1529)* only functions with this processor. If the plugin is started from an **Oxygen XML Editor/Author** distribution, a Chemistry installation is not needed.
- **Prince XML** - A commercial product, available at: https://www.princexml.com/.
- **Antenna House** - A commercial product, available at: https://www.antennahouse.com/formatter.

Technical Details

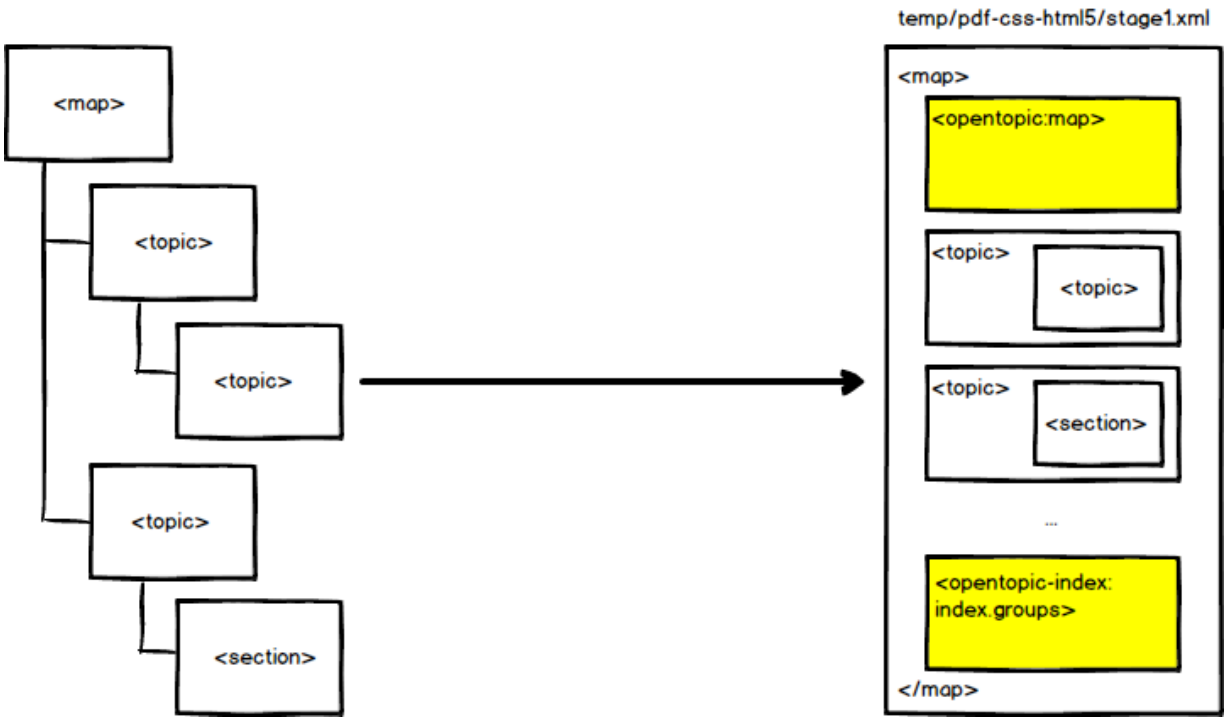
The **DITA-OT CSS-based PDF Publishing Plugin** comes bundled in the **Oxygen XML Editor/Author** distributions. The plugin ID is: **com.oxygenxml.pdf.css**. It is installed in the `[OXYGEN-INSTALL-DIR]frameworks/dita/DITA-OT/plugins/com.oxygenxml.pdf.css` folder.

It has the following transformation types:

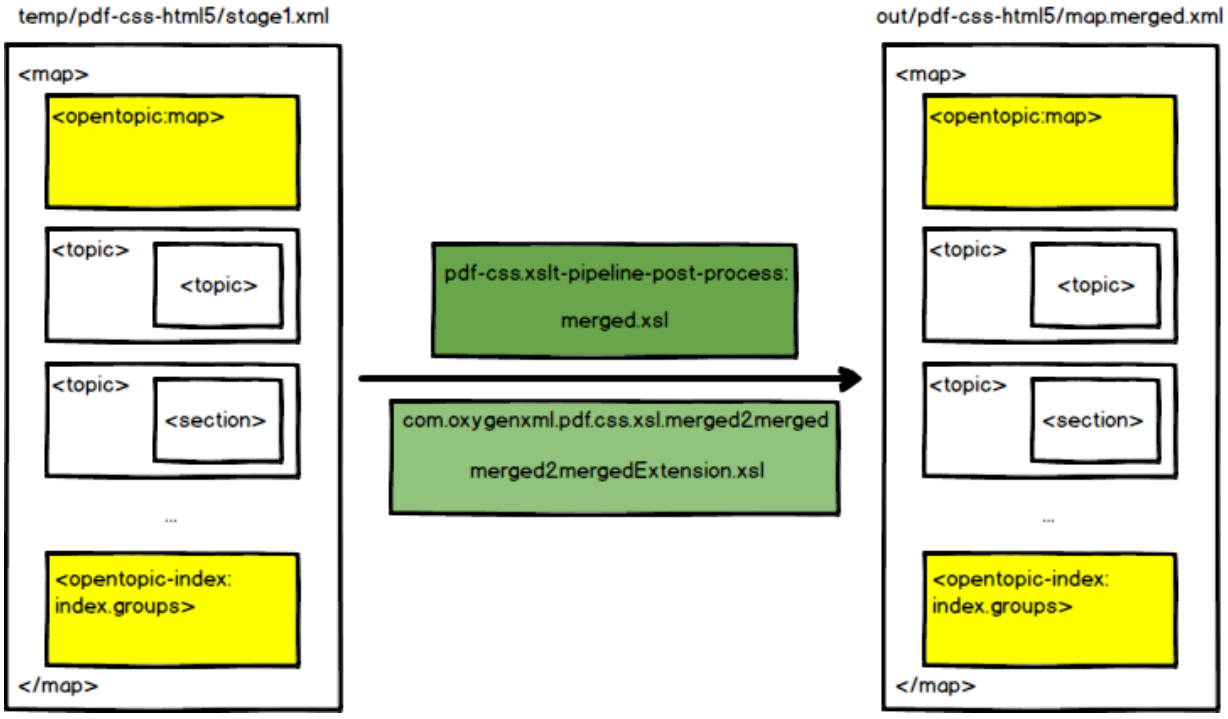
- **pdf-css-html5** (*DITA Map PDF - based on HTML5 & CSS* transformation) - CSS styling applied over a merged HTML5 document (the merged DITA map converted to HTML5).
- **pdf-css-html5-single-topic** (*DITA PDF - based on HTML5 & CSS* transformation) - CSS styling applied over a merged HTML5 document (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.

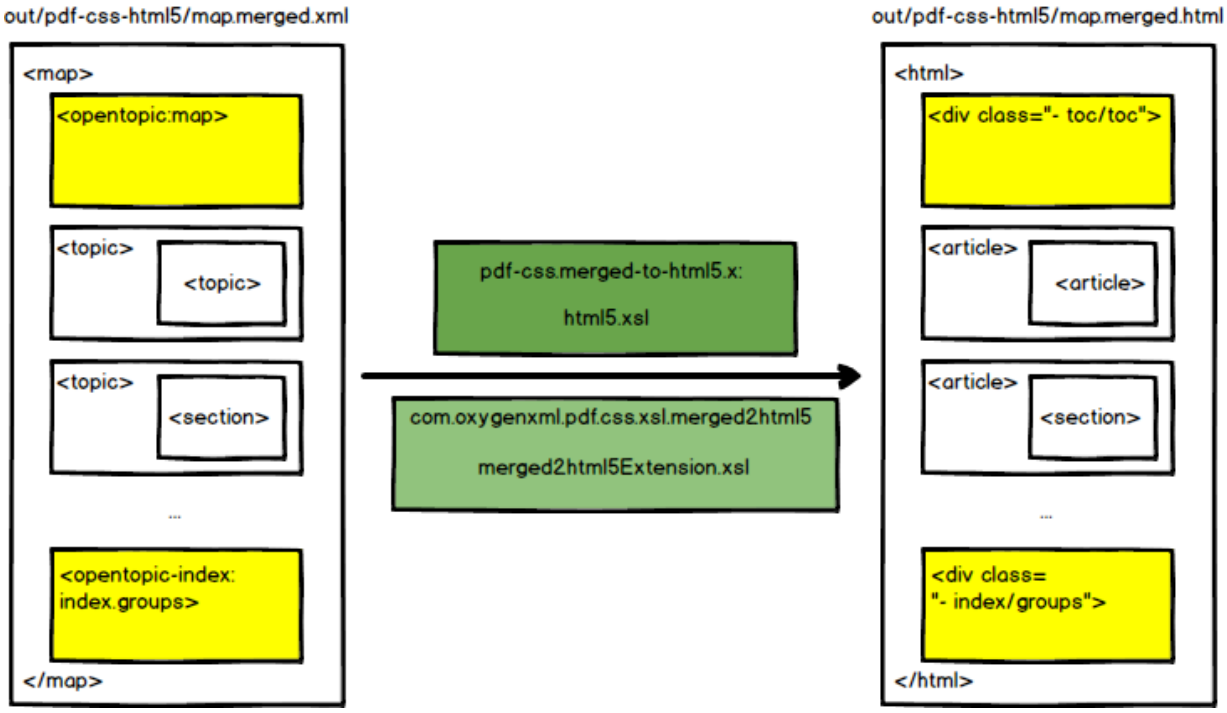


3. It post-processes the merged map. It fixes some of the structure in the TOC and index, moves the *frontmatter* and *backmatter* to the correct places, transforms any change tracking and review processing instructions to elements that can be styled later, etc. During this phase, the `com.oxygenxml.pdf.css.xsl.merged2merged` *(on page 1665)* extension points are also called. The result is another merged map.

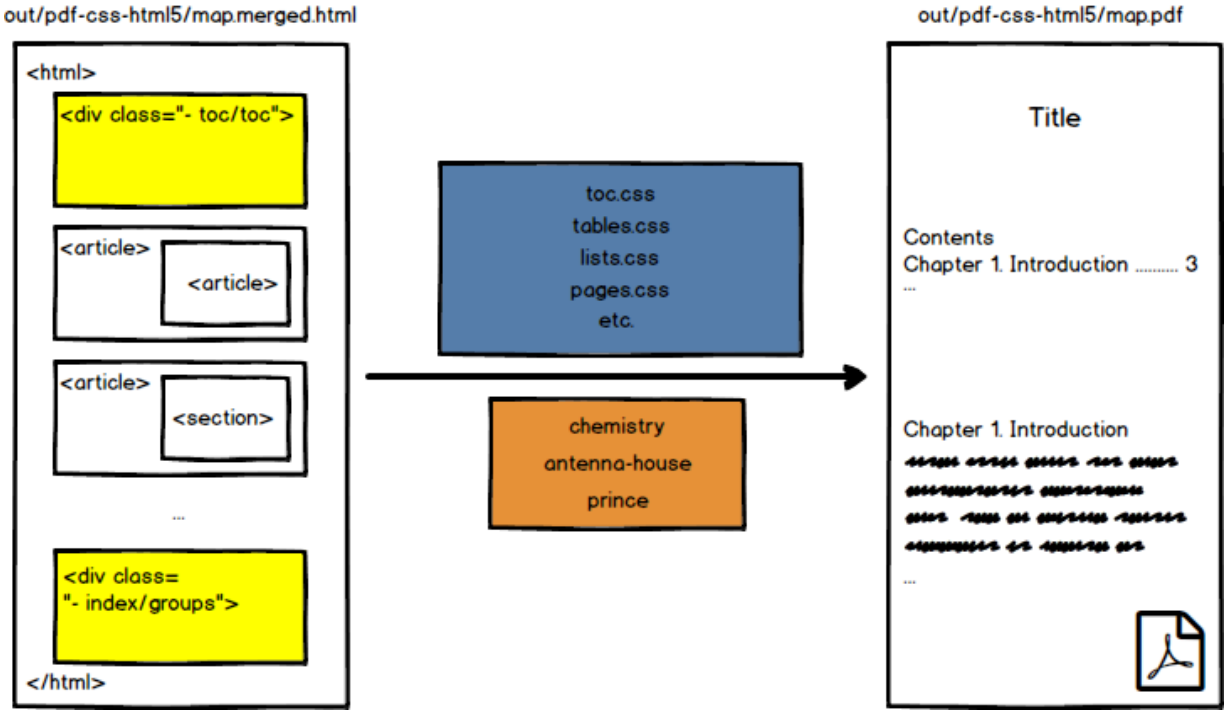


Note:
In the single topic transformation type (**DITA PDF - based on HTML5 & CSS**), these steps are simplified.

4. It converts the post-processed merged map or topic into a single HTML5 file. The generated HTML elements have the `@class` attribute from their original DITA elements. This means that you can either use selectors that were designed for DITA structure, or ones for the HTML structure. For more details, see Reusing the Styling for WebHelp and PDF Output *(on page 1616)*. During this phase, the `com.oxygenxml.pdf.css.xsl.merged2html5` *(on page 1665)* extensions points are also called.



5. It uses a collection of CSS stylesheets against the merged HTML5 file and uses a PDF processor to generate the final PDF. References to the CSS files are collected from the publishing template *(on page 1459)*.



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.

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 HTML document).
- From the Chemistry PDF CSS processor (the transformation of the merged HTML document to PDF).

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. Uncheck the **Prefer using the "dita" command** option
5. 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

- **If the DITA-OT process fails with Out Of Memory Error:** 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=-Xmx2048m
```

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.

Note:
These parameters must be prefixed by "-D" when used from a command line.

antenna-house.option.setting.file	Path to the Antenna House Option Setting File (eg: `C:\path\to\formatter\options\settings.xml` on Windows).
args.allow.external.coderefs	Enables the inclusion of code files that are located outside the DITA map folder hierarchy, referenced using the DITA `<coderef>` element. Allowed values are **yes** or **no** (default).
args.chapter.layout	Specifies whether chapter-level TOCs are generated for bookmaps. When set to **MINITOC**, a small section with links is added at the beginning of each chapter. The default is **BASIC**. For details, see: Table of Contents on a Page (Mini TOC) *(on page 1564)*. Allowed values: • **BASIC** - No chapter TOC is created. • **MINITOC** - A chapter-level TOC is generated. • **MINITOC-BOTTOM-LINKS** - A chapter-level TOC is generated, with the links under the chapter description.
args.css	You can use this to specify a list of CSS URLs to be used in addition to those specified in the publishing template. The files must have URL syntax and be separated using semicolons.
args.css.param.*	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 1660)*.
args.css.param.clone-referenced-footnotes	You can use this parameter to control the footnotes behavior:

	• When set to yes, footnotes that are referenced multiple times throughout a publication are cloned and placed at the bottom of the page for each occurrence. • When set to no (default value), only the first footnote reference is placed at the bottom of the page and subsequent references point back to the original footnote.
args.css.param.pdf-split-mode	You can use this parameter to generate additional PDFs alongside the main PDF publication. Allowed values: • **none** - Only the main PDF is generated. • **chapters** - All the chapters are generated beside the main PDF. The PDF chapter file name is equal to the original chapter `@id`. • **parts** - All the parts are generated beside the main PDF. The PDF part file name is equal to the original part `@id`.
args.css.param.numbering	You can use this parameter to change the numbering of the first-level topics (chapters) and nested topics. Allowed values: • **shallow** - Only the topics from the first level are numbered (chapters). This is the default. • **deep** - All the topics from the map are numbered (nested topics up to level 3). • **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 1550)*.
args.css.param.numbering-sections	Controls whether or not the sections are included in the table of contents. When set to **yes** (sections are included), they are numbered ac-

	cording the numbering scheme set by the `args.css.param.numbering` parameter.
args.css.param.show-onpage-lbl	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**.
args.css.param.show-profiling-attributes	Controls whether or not the profiling attributes are displayed in the output. Allowed values: • **yes** • **no** (default)
args.css.param.title.layout	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). • **normal** • **table** (avoid wrapping text under counter)
args.draft	Specifies whether or not the content of `<draft-comment>` and `<required-cleanup>` elements is included in the output. Allowed values: • **no** (default) - No draft information is shown in the output. • **yes** - The draft information is shown in the output.
args.figurelink.style	Specifies how cross references to figures are styled in output. Allowed values: • **NUMBER** - Only the number of the figures are shown in links. • **TITLE** - Only the title of the figures are shown in links. • **NUMTITLE** (default) - Both the title and number of the figures are shown in links.

args.gen.task.lbl	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 "About this task", "Before you begin", "Procedure", or "What to do next", are shown in the task contents.
args.hyph.dir	Specifies the directory that contains custom hyphenation dictionaries. Fore more details see: Hyphenation *(on page 1594)*.
args.input	Specifies the main DITA map file for your documentation project.
args.keep.output.debug.files	Specifies whether or not the debug files generated during the transformation should be kept in the output folder. Allowed values: **YES** (default) or **NO**.
args.output.base	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. A common use-cases is to use the ditamap title instead of the ditamap filename, the parameter value then become `${xpath_eval(normalize-space(string-join(/*[contains(@class, 'map/map')]/*[contains(@class, 'topic/title')]//text()))}`. **Note:** To replace spaces by a custom separator the query should call the `replace()` function: `${xpath_eval(replace(normalize-space(string-join(/*[contains(@class, 'map/map')]/*[contains(@class, 'topic/title')]//text())), '\s', '_'))}`.
args.rellinks.group.mode	Specifies the related links grouping mode. All links can be grouped into a single "Related Information" group or links grouped by their target type (topic, task, or concept). Allowed values: **single-group** (default) or **group-by-type**. Fore more details see: How to Group Related Links by Type *(on page 1628)*.
args.tablelink.style	Specifies how cross references to tables are styled in output. Allowed values: • **NUMBER** - Only the number of the tables are shown in links. • **TITLE** - Only the title of the tables are shown in links. • **NUMTITLE** (default) - Both the title and number of the tables are shown in links.
clean.temp	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**.

chemistry.log.file	Specifies the path to the `logback.xml` configuration file used by Chemistry. For more details, see: Console Logging *(on page 1454)*.
chemistry.merge.after	Specifies a list of PDF URLs to be added at the end of the generated PDF. The files must have URL syntax and be separated using semicolons.
chemistry.merge.before	Specifies a list of PDF URLs to be added in front of the generated PDF. The files must have URL syntax and be separated using semicolons.
chemistry.security.policy	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`.
chemistry.security.resources.dir	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 `chemistry.security-.policy` parameter is set.
chemistry.security.resources-.host	The host, specified as `name:port`, that Chemistry will use to get resources (e.g. CSS files, images, fonts). This optional parameter should only be used when the `chemistry.security.policy` parameter is set.
chemistry.security.workspace	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.
chemistry.sign.keystore	Specifies the path to the keystore file used by Chemistry to sign the PDF. This parameter must be set if you want to sign your PDF.
chemistry.sign.location	Specifies the location of the person/company who certified the PDF.
chemistry.sign.name	Specifies the name of the person/company who certified the PDF.
chemistry.sign.password	Specifies the export password of the keystore used to sign the PDF. Mandatory if the keystore has been exported with a password.
chemistry.sign.reason	Specifies the reason for certifying the PDF.
css.processor.path.antenna-house	Path to the Antenna House executable file that needs to be run to generate the PDF (for example, `C:\path\to\AHFCmd.exe` on Windows).
css.processor.path.chemistry	Path to the **Oxygen PDF Chemistry** executable file that needs to be run to generate the PDF (for example, `C:\path\to\chemistry.bat` 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**.
css.processor.path.prince	Path to the Prince executable file that needs to be run to generate the PDF (for example, `C:\path\to\prince.exe` on Windows).

css.processor.type	Specifies the processor to use for the transformation. Allowed values: **chemistry** (default), **antenna-house**, or **prince**.
default.language	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 1702)*.
drop.block.margins.at.page-.boundary	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**.
editlink.ditamap.edit.url	Use this parameter to add an *Edit* link next to the topic title in the Web-Help 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.
editlink.additional.query.parameters	You can use this optional parameter to add additional parameters to be appended to each generated edit link. Each parameter must start with *&* (for example: *&tags-mode=no-tags*).
editlink.remote.ditamap.url (deprecated)	Use this parameter in conjunction with `editlink.web.author.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 custom URL of the *main DITA map*. For example, a GitHub custom URL might look like this: `https://getFileContent/oxygenxml/userguide/master/UserGuide.ditamap`.
editlink.web.author.url (deprecated)	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: `https://www.oxygenxml.com/oxygen-xml-web-author/`.
enable.character.level.font.fallback	When set to **yes** (default), character-level font fallback is enabled for better Unicode support. Otherwise, a single font per word is used.
enable.chunk.processing	Enables the processing of the `@chunk` attribute. By default, this stage is skipped but it needs to be enabled, for example, if both the `@chunk` and `@copy-to` attributes are present on a `<topicref>`. Accepted values: **true** or **false**.

enable.latin.glyph.substitutions	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 (e.g. *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. **Warnings:** • Disabling substitutions also disables Latin ligatures. • 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.
expand.xpath.in.svg.templates	Expands XPath expressions (whose format is `${expression}`) contained in SVG templates. Allowed values: **yes** (default) or **no**.
figure.title.placement	Controls the title placement of the figures, relative to the image. Possible values include **top** (default) and **bottom**.
filter.unused.glossentries	When set to **no** (default), all glossary entries are displayed in the glossary. If set to **yes**, only referenced entries are displayed.
fix.external.refs.com.oxygenxml	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).
hide.frontpage.toc.index.glossary	When set to **yes**, the generated structures (table of contents, index list, front page, etc.) are removed from the output. The default is **no**.
image.resolution	You can use this parameter to set the default resolution used by images. It works mainly on *vector* images since *raster* images have their resolution defined in their metadata. The default is **96** (dpi). For more information, see how to change images resolution *(on page 1629)*.

pdf.accessibility	When set to **yes**, the PDF output is generated in compliance with the PDF/Universal Accessibility standard (also known as ISO 14289). The default is **no**.
pdf.archiving.mode	Specifies the archiving mode. The PDF output will be generated in compliance with the PDF/A standard. Allowed values (not set by default): • **PDF/A-1a** • **PDF/A-1b** • **PDF/A-2a** • **PDF/A-2b** • **PDF/A-2u** • **PDF/A-3a** • **PDF/A-3b** • **PDF/A-3u**
pdf.compress	When set to **yes**, the PDF output is compressed. This may increase processing time. The default is **no**.
pdf.version	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 **1.5**.
pdf.security.restrict.printhq	Restricts high quality printing. Used for protecting the PDF Document. The restriction is off by default. Accepted values: **yes** or **no**.
pdf.security.restrict.assembledoc	Restricts assembling document (e.g. adding pages). Used for protecting the PDF Document. The restriction is off by default. Accepted values: **yes** or **no**.
pdf.security.restrict.accesscontent	Restricts extracting text and graphics. Used for protecting the PDF Document. The restriction is off by default. Accepted values: **yes** or **no**.
pdf.security.restrict.fillinforms	Restricts filling in existing interactive forms. Used for protecting the PDF Document. The restriction is off by default. Accepted values: **yes** or **no**.
pdf.security.restrict.annotations	Restricts filling in existing interactive forms. Used for protecting the PDF Document. The restriction is off by default. Accepted values: **yes** or **no**.
pdf.security.restrict.print	Restricts printing. Used for protecting the PDF Document. The restriction is off by default. Accepted values: **yes** or **no**.
pdf.security.restrict.copy	Restricts copying content. Used for protecting the PDF Document. The restriction is off by default. Accepted values: **yes** or **no**.

pdf.security.restrict.edit	Restricts copying content. Used for protecting the PDF Document. The restriction is off by default. Accepted values: **yes** or **no**.
pdf.security.user.password	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.
pdf.security.owner.password	Owner password. There are no restrictions for people using this password.
pdf.security.encrypt.metadata	Encrypts the metadata. By default active when other security parameters are set. Accepted values: **yes** or **no**.
show.changes.and.comments	When set to **yes**, the user comments, colored highlights and tracked changes are shown in the output.
show.changes.and.comments-.as.changebars	When set to **yes** (default) and the `show.changes.and.comments` parameter is also set to **yes**, the user comments and tracked changes are shown as change bars in the PDF output. This parameter can be used in conjunction with the `show.changes.and.comments.as.pdf.sticky.notes` parameter to choose whether the change bars are displayed in footnotes or sticky notes. You can override this from your customization CSS *(on page 1469)*.
show.changes.and.comments-.as.pdf.sticky.notes	When set to **yes** (default) and the `show.changes.and.comments` parameter is also set to **yes**, the user comments and tracked changes are shown in the PDF output as sticky note annotations. When set to **no**, 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 *(on page 1469)*.
show.changed.text.in.pdf.sticky-.notes.content	When set to **yes** (default) and both the `show.changes.and.comments` and `show.changes.and.comments.as.pdf.sticky.notes` parameters are also set to **yes**, the inserted and deleted text is shown in the sticky note annotations. When set to **no**, only the *inserted* and *deleted* labels are shown in the annotations (this is useful for search scope).
show.image.map.area.numbers	When set to **yes**, a counter for each area from the image map is displayed over the image, near the defined shape. The default is **no**.
show.image.map.area.shapes	When set to **yes**, each of the image map area shapes is displayed with a translucent fill over the image. You can use this to debug your image maps. The default is **no**.

show.media.as.link	When set to **yes**, media objects will not appear and an external link is generated for each one instead.
sort.and.group.glossentries	When set to **no** (default), glossary entries are sorted based upon the document order. If set to **yes**, they are sorted alphabetically and grouped by their first letter. **Warning:** This only works for glossary entries inside the `<glossgroup>` element or children of the `<glossarylist>` element.
store-type	Setting this parameter to **memory** will increase the processing speed and thus, could help decrease the publishing time.
table.title.placement	Controls the placement of the title for tables. Possible values include **top** (default) and **bottom**.
table.title.repeat	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 **yes** (default) or **no**.
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:

pdf.publishing.template	Specifies the path to the folder containing the custom PDF template.
pdf.publishing.template.descriptor	Specifies the name of the descriptor file to be loaded from the PDF template folder or package. If not specified, the first encountered descriptor file is loaded.

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):

args.root.map	Specifies the path of the root map file used to expand the key references in the published topic.
args.enable.root.map.key.processing	Indicates whether or not the keys should be processed using the root map parameter. Allowed values: • **auto** (default) • **yes** • **no**

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 `-v` (or `--verbose`) DITA-OT parameter from the command line.



Note:
When the transformation is started from an **Oxygen** application, this parameter is automatically set.

Chemistry Console Logging

Besides the global verbosity, it is possible to control **Oxygen PDF Chemistry** console logging separately by creating a custom `logback.xml` configuration file. This is helpful if you want to hide the INFO messages or write the logs into a separate file.

Related Information:
Oxygen PDF Chemistry: Logging

License Key

When running the **Oxygen PDF Chemistry** engine or the **Oxygen Publishing Engine** from inside a started **Oxygen XML Editor/Author** installation, an extra license key is not required. The sections below pertain to running the publishing from the command line.

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 from the command line.

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 from the command line.

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 1444)*. Here is an example of how to write the commands:

- Windows

```
dita.bat -f pdf-css-html5 -i C:\path\to\map.ditamap -o C:\path\to\output\folder -v
```

- Linux/macOS

```
dita -f pdf-css-html5 -i /path/to/map.ditamap -o /path/to/output/folder -v
```



Note:
You can use the long form of the command-line options (e.g. `--format` or `--input`).

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 **Oxygen Publishing Engine**.
2. Go to the root of your Maven project and edit the `pom.xml` file to include the following fragment:

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
                      http://maven.apache.org/xsd/maven-4.0.0.xsd">
  <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</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>
      <plugin>
        <groupId>org.codehaus.mojo</groupId>
        <artifactId>exec-maven-plugin</artifactId>
        <version>1.6.0</version>
        <executions>
          <execution>
            <id>generate-pdf-css</id>
            <phase>generate-sources</phase>
            <goals>
              <goal>exec</goal>
            </goals>
```

```
            <configuration>
              <executable>${dita-ot-dir}/bin/dita</executable>
              <arguments>
                <argument>--format=pdf-css-html5</argument>
                <argument>--input=${input-file}</argument>
                <argument>--output=${output-dir}</argument>
                <argument>-Dpdf.publishing.template=${publishing-template}</argument>
                <argument>-v</argument>
              </arguments>
            </configuration>
          </execution>
        </executions>
      </plugin>
    </plugins>
  </build>
</project>
```

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.

Tip:
You can start creating publishing templates by using the **Oxygen Styles Basket**. https://styles.oxygenxml.com

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 409. Oxygen Publishing Template Package (WebHelp Responsive)

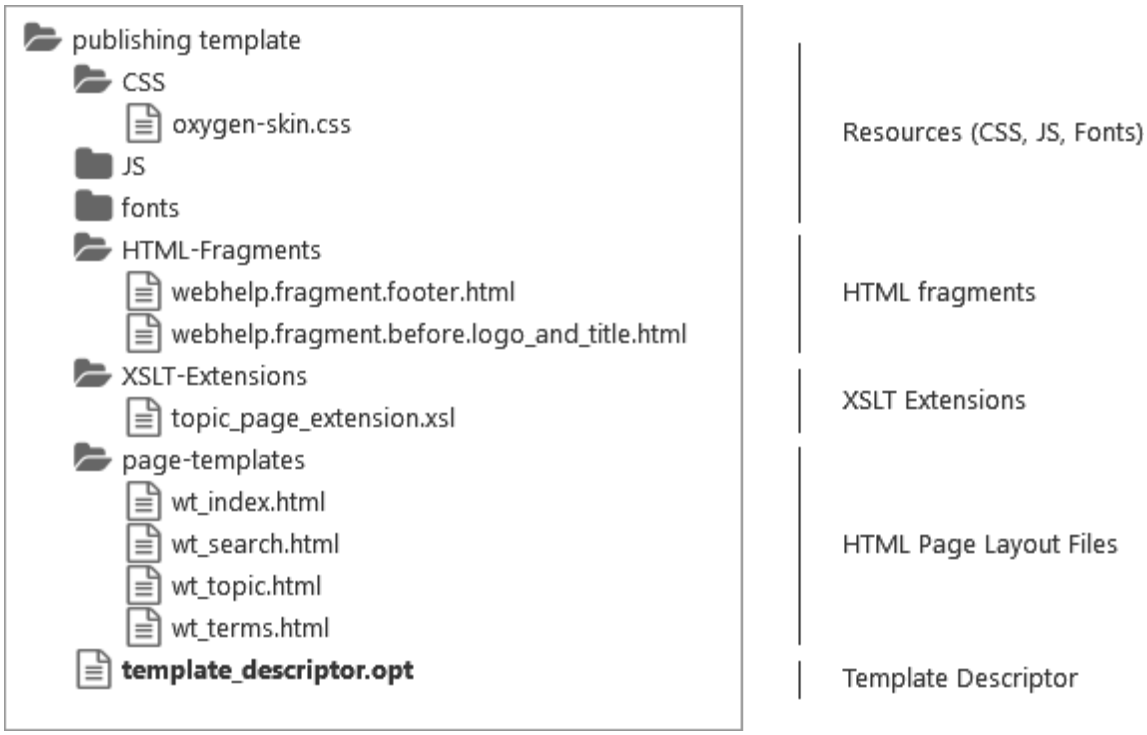


Figure 410. Oxygen Publishing Template Package (PDF)



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 1309)*
How to Edit a Packed Publishing Template *(on page 1324)*
How to Add a Publishing Template to the Publishing Templates Gallery *(on page 1324)*
How to Share a Publishing Template *(on page 1469)*

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.

```
<publishing-template>
    <name>Flowers</name>
    <description>Flowers themed light colored template</description>
    ...
```

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>`.

```
<publishing-template>
    ...
    <author>
      <name>John Doe</name>
      <email>jdoe@example.com</email>
      <organization>ACME</organization>
      <organizationUrl>http://www.example.com/jdoe</organizationUrl>
    </author>
    ...
```

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 1461)*, template preview image *(on page 1461)*, resources *(on page 1461)* (such as CSS files), transformation parameters *(on page 1462)*, or XSLT extensions *(on page 1463)*.

```
    <pdf>
        <tags>
          ...
        </tags>
        <preview-image file="MyPreview.png"/>
```

```
      <resources>
        ...
      </resources>

      <parameters>
        ...
      </parameters>
    </pdf>
```

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.

```
<publishing-template>
    ...
    <pdf>
      <tags>
        <tag>purple</tag>
        <tag>light</tag>
      </tags>
```

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.

```
<publishing-template>
    ...
    <pdf>
      ...
      <preview-image file="ashes/ashes-tree.png"/>
      <online-preview-url=https://www.example.com/samples/tiles/ashes</online-preview-url>
```

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).

```
<publishing-template>
    ...
    <pdf>
      ...
      <resources>
          <css file="css/custom_styles.css"/>
          <css file="css/custom_fonts.css"/>
      </resources>
```

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.

```
<publishing-template>
    ...
    <pdf>
      ...
      <parameters>
        <parameter name="show.changes.and.comments" value="yes"/>
      </parameters>
    </pdf>
```

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 1464)* and adding it to the templates gallery *(on page 1468)*, 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:

```
<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>
```

For more information about the available extension points, see: XSLT Extensions for PDF Transformations *(on page 1665)*.

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 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.

```
<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 1271)*

How to Create a Publishing Template

To create a customization, you can use the **Oxygen Styles Basket**, start from scratch or from an existing template, and then adapt it according to your needs.

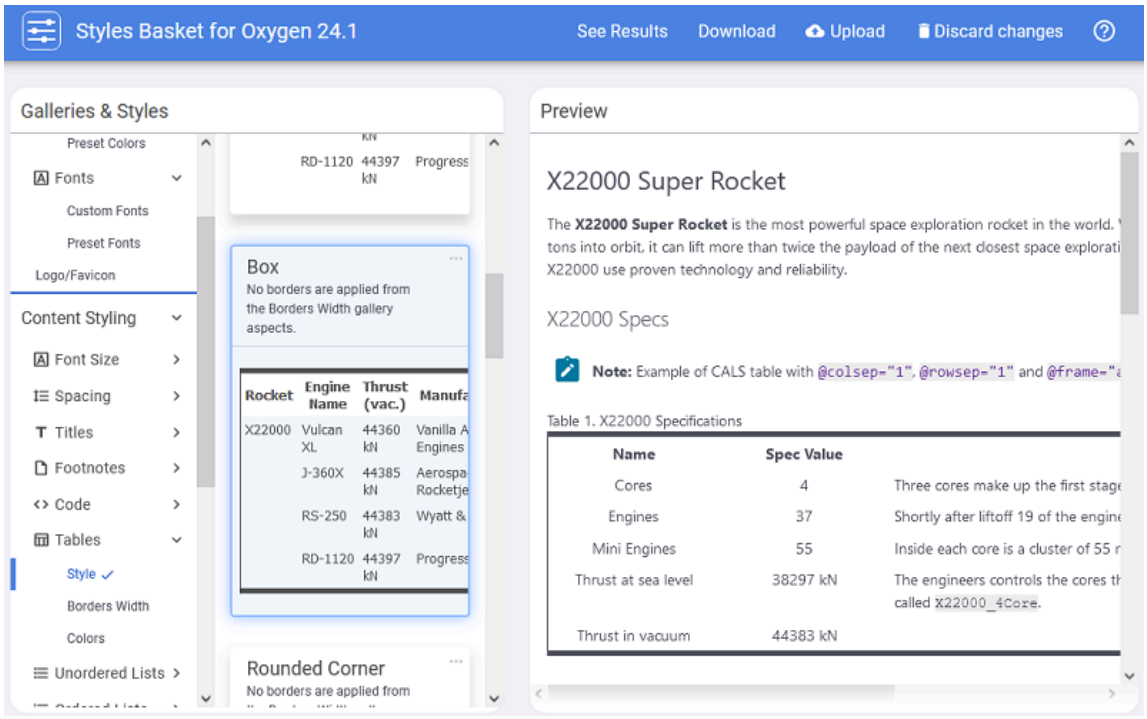
Creating a Publishing Template Using the Oxygen Styles Basket

Oxygen Styles Basket is a handy free-to-use web-based visual tool that helps you create your own *Publishing Template Package* to customize your **DITA Map WebHelp Responsive** and/or **DITA Map PDF - based on HTML5 & CSS** transformation scenarios.

To create an *Oxygen Publishing Template*, simply visit the galleries and pick styling aspects. Various different types of styles can be selected (such as fonts, tables, lists, spacing, code) and all changes can be seen in

the **Preview** pane. You can also click the **See Results** button to generate a preview of either PDF or WebHelp output.

Figure 411. Oxygen Styles Basket Interface



After creating your custom style, you can **Download** the current template, then back in **Oxygen**, click **Choose Custom Publishing Template** and select the downloaded archive in the transformation dialog to apply it.

At any time you can **Upload** a previously generated template for further customization.

Tip:
For more information about the **Oxygen Styles Basket**, see the following resources:

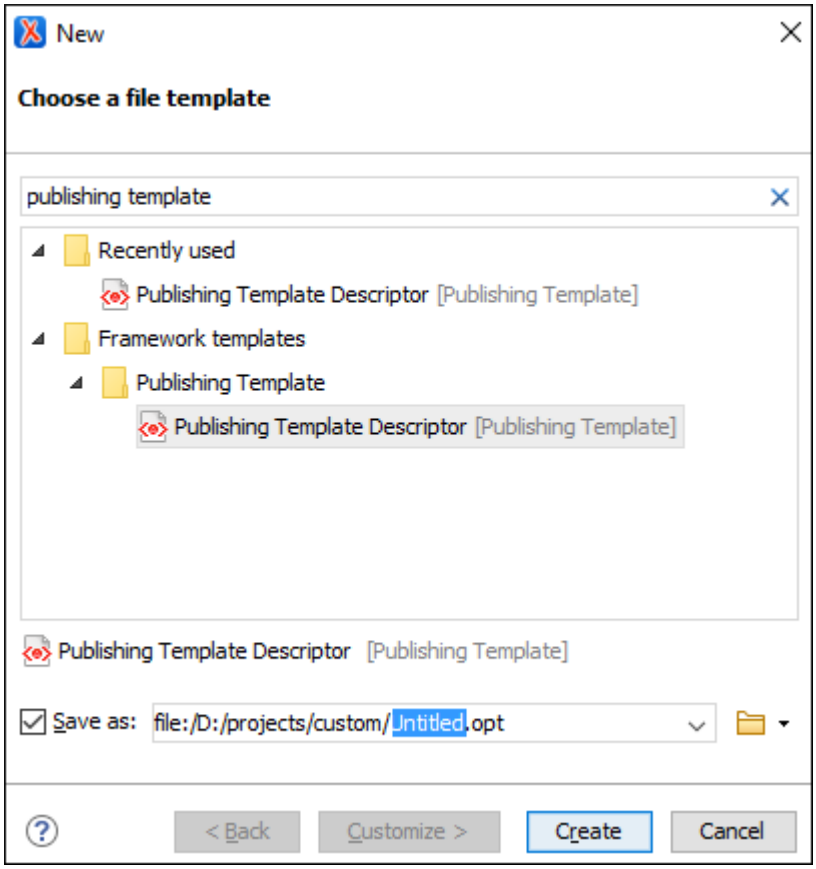
- **Video: Introducing the New Oxygen Styles Basket**
- **Webinar: Using Oxygen Styles Basket to Create CSS Customization from Scratch**

Creating a Publishing Template Starting from Scratch

To create a new *Oxygen Publishing Template (on page 2306)*, 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 412. Choosing the Publishing Template Descriptor Document 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 **DITA Map WebHelp Responsive** or **DITA Map PDF - based on HTML5 & CSS** transformation, the easiest way to create a new *Oxygen Publishing Template (on page 2306)* 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 1287)* 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/zNmXfKWXwO8

Related information
Publishing Template Package Contents for PDF Customizations *(on page 1459)*
Publishing Template Package Contents for WebHelp Responsive Customizations *(on page 1271)*

How to Edit a Packed Publishing Template

To edit an existing *Oxygen Publishing Template (on page 2306)* 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 1459)*
Publishing Template Package Contents for WebHelp Responsive Customizations *(on page 1271)*

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:

```
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 1444)*

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`).

Customizing PDF Output Using CSS

The publishing process is driven by a *customization CSS*.

Warning:
You should not edit the CSS stylesheet from *`DITA-OT-DIR`*`/plugins/com.oxygenxml.pdf.css/css/print`. Instead, create your own customization.

To change the styling of the output for the **DITA Map PDF - based on HTML5 & CSS** or the **DITA PDF - based on HTML5 & CSS** transformation scenarios you can either create your own custom CSS rules or create a publishing template using the **Oxygen Styles Basket**.

Create Custom CSS Rules from Scratch

1. Create a 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.

   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.

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.
3. Link the CSS file. For this, 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 1267)*.
- 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.

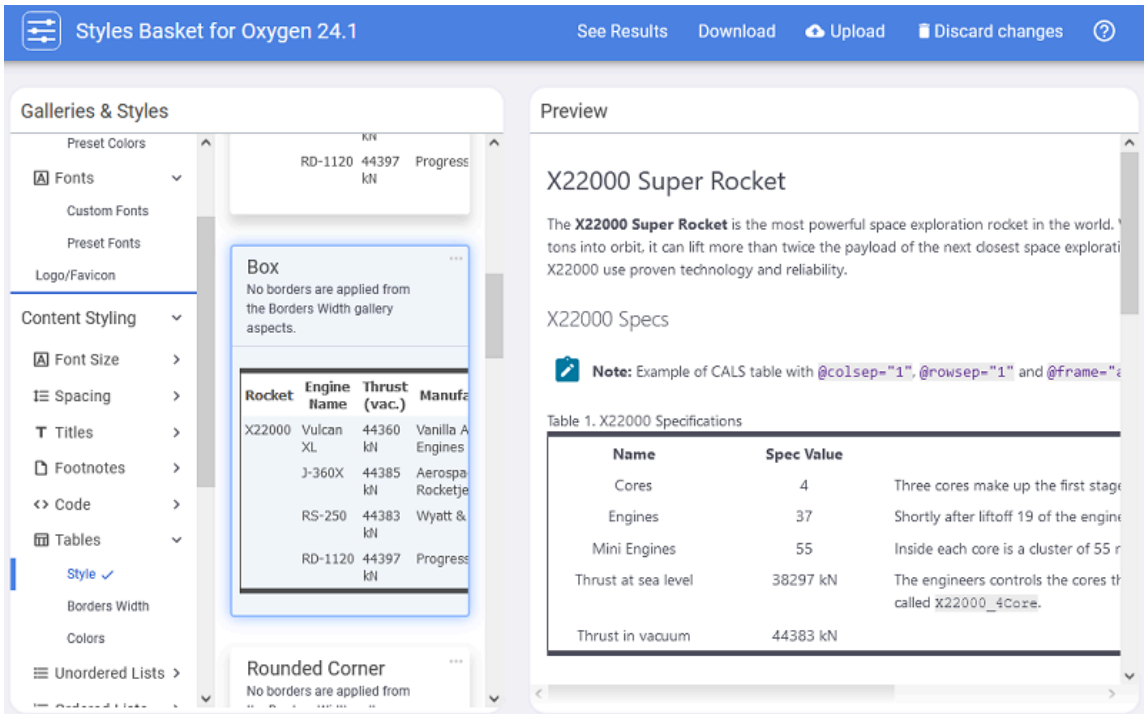
Creating a Publishing Template Using the Oxygen Styles Basket

Another way to create an *Oxygen Publishing Template (on page 2306)* is to use the **Oxygen Styles Basket**. This tool is a handy free-to-use web-based visual tool that helps you create your own *Publishing Template Package* to customize your **DITA Map PDF - based on HTML5 & CSS** transformation scenarios.

It is based on galleries that you can visit to pick styling aspects to create a custom look and feel. Various different types of styles can be selected (such as fonts, tables, lists, spacing, code) and all changes can be seen in the **Preview** pane. You can also click the **See Results** button to generate a preview of either PDF or WebHelp output.

It is possible to **Download** the current template or **Upload** a previously generated template for further customization.

Figure 413. Oxygen Styles Basket Interface



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**:
  https://www.youtube.com/embed/5NsVEOvxbas
- **Transforming DITA documents to PDF using CSS, Part 2 – Book Design, Pagination, Page Layout, and Bookmarks**:
  https://www.youtube.com/embed/UiYwPBOJQcg
- **Transforming DITA documents to PDF using CSS, Part 3 – Advanced Fonts Usage**:
  https://www.youtube.com/embed/1fzS8AzOGao
- **Transforming DITA documents to PDF using CSS, Part 4 – Advanced CSS Rules**:
  https://www.youtube.com/embed/rs04iX_RdIk

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.

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 Merged Map File

During the transformation stages, two merged map files are created. These files could be used to help debug unexpected results.

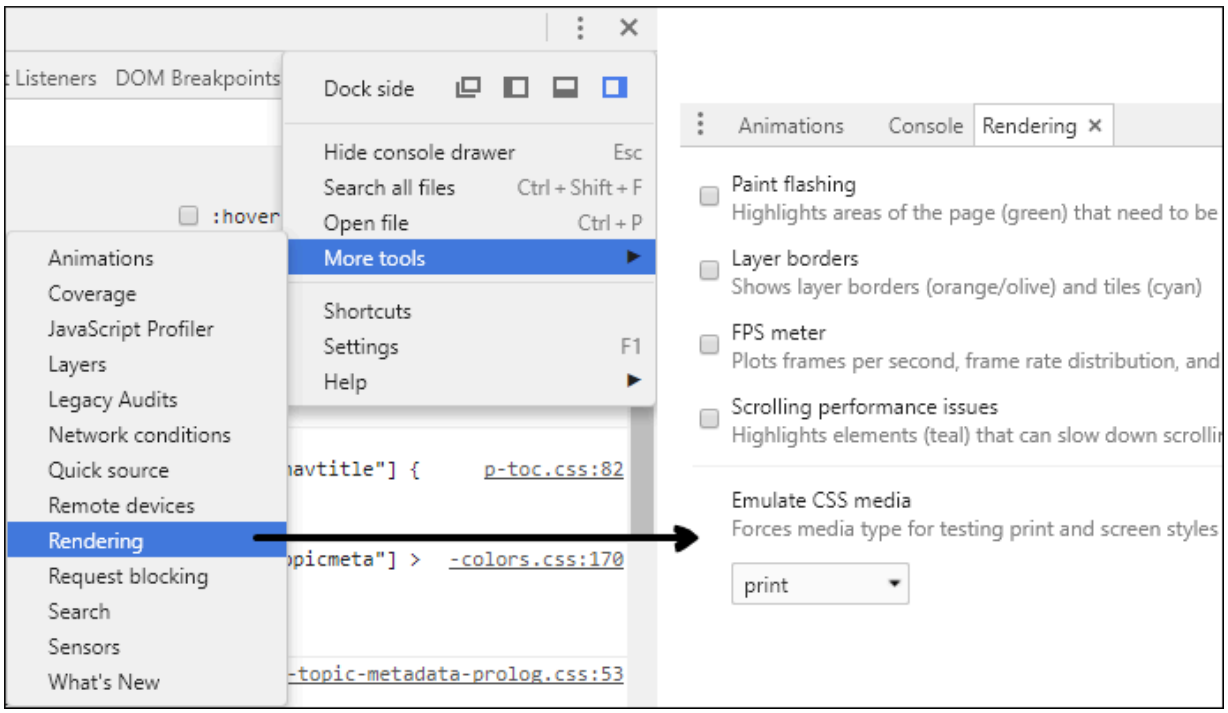
1. The first thing you should try is to check the file structure of the **HTML merged map file**. This file can be found in the `out/pdf-css` directory and it has the `.merged.html` file extension (you will also find a `.merged.xml` file that aggregates the entire DITA map structure). You can open the 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. If the structure is as expected, you can start checking that the CSS selectors are written correctly against the document structure.
3. If the CSS selectors are correctly written, you can start inspecting how the styles are applied (you can try any of the methods listed below).

Inspecting the Applied Styles Using a Browser

The following procedure explains how to inspect the applied CSS styles using Chrome, but any modern browser can be used and the procedure for each of them is similar:

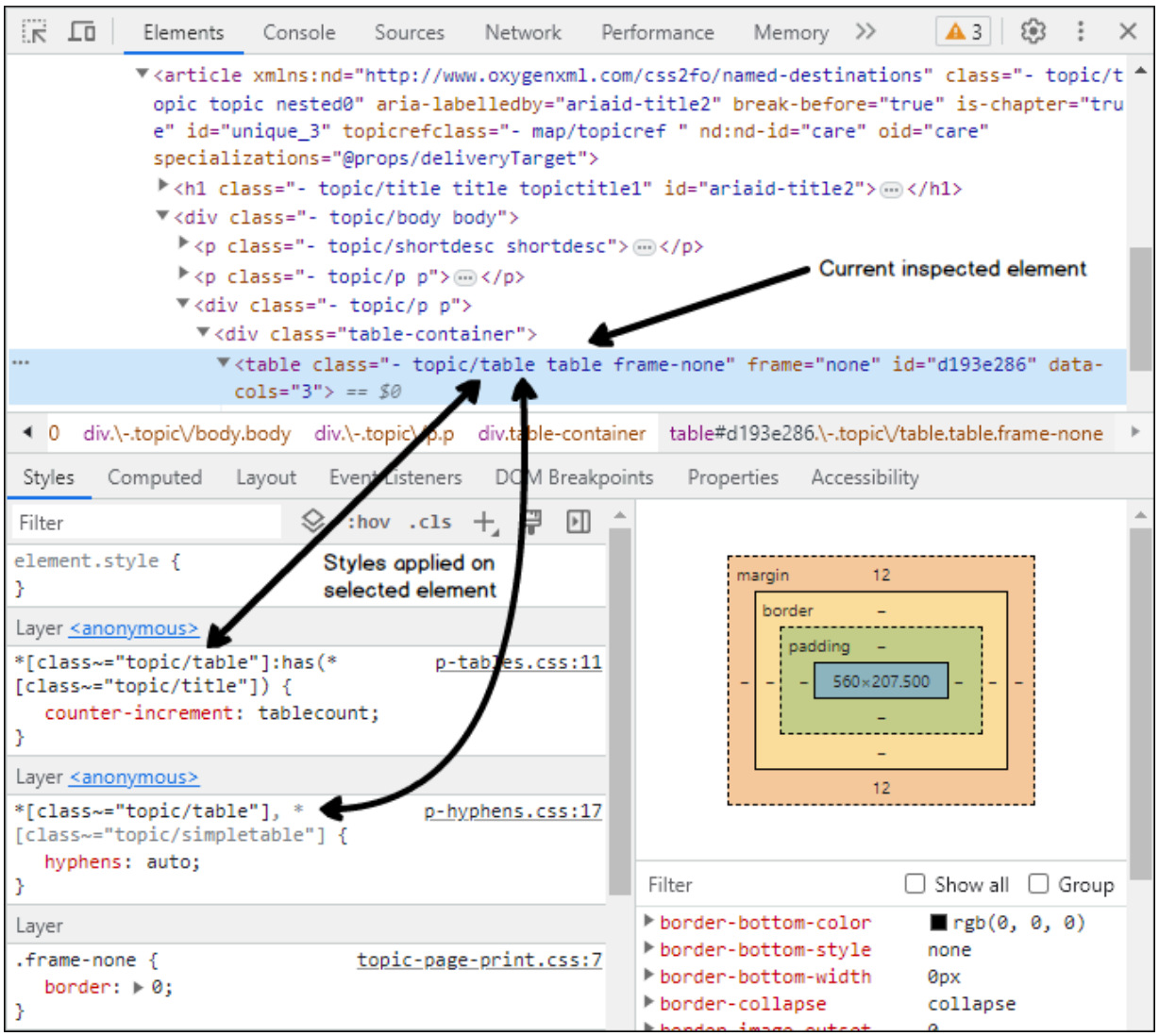
1. Open the file ending in `.merged.html`.
2. Open the **Chrome Developer Tools** by using ⋮ **> More Tools > Developer Tools** (or press **CTRL+SHIFT+I**).

3. Activate the **Rendering** pane by using ⋮ **> More Tools > Rendering** then select **print** from the **Emulate CSS media** section. This will activate the CSS selectors enclosed in `@media print {..}`:



> **Note:**
> This allows you to debug the styling of elements, the table of contents, and the index, but not the styles of the page margin boxes (headers, footers) or page breaks.

4. Right-click on the element you want to inspect and select the **Inspect** action, you will see the element (in the **Elements** pane) and the list of styles that are applied on it (in the **Styles** pane):



> **Tip:**
> Clicking any of the stylesheet links from the **Styles** pane opens the original CSS files in the **Sources** pane. Editing the rules in that pane results in a live preview of how the change will affect the output (these modifications will be lost on reload).

Inspecting the Applied Styles Using Oxygen XML Editor/Author

To inspect the applied CSS styles using **Oxygen**:

1. In **Oxygen XML Editor/Author**, open the file ending in `.merged.html`.
2. From the **Styles** toolbar, choose the **+ Print Ready** entry. This will activate certain CSS selectors enclosed in `@media print {..}`.

> **Note:**
> This allows you to debug the styling of elements, the table of contents, and the index, but not the styles of the page margin boxes (headers, footers) or page breaks.

3. Right-click on the element you want to inspect and select the **Inspect Styles** action. The dedicated **CSS Inspector** view will be opened and it will show the applied 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.

Other Debugging Techniques

Here are some other debugging techniques you may find useful:

- Add background and border properties to the specific CSS rule. If they do not appear in the output, then there is a problem with the rule selector.
- Add the `!important` keyword to a property that is not applied, or make the selector more specific (by adding more parent selectors).
- Add the following fragment in your customization CSS to show how the elements are mapped to PDF:

```
* {
   border: 1pt solid blue !important;
}

*:before(1000) {
   content:  oxy_name() !important;
   color: orange;
}

*:before(900) {
   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 are hidden.

How to Speed up CSS Development and Debugging

You may have already run the **DITA Map PDF - based on HTML5 & CSS** transformation scenario before using this procedure.

You can speed up your CSS development considerably by not invoking the entire pipeline of transforming your DITA maps to PDF. Instead, you can directly transform the merged map *(on page 1471)* (`.merged.html`) into PDF using **Oxygen PDF Chemistry**.

1. Open the `.merged.html` located in the output directory in the editor.
2. Configure a new **XML to PDF transformation with CSS** scenario. There is no need to set the CSS URL in the resulting dialog box. The stylesheets are already declared in the file `<head>`. This scenario uses the Chemistry CSS processor.
3. **Optional:** Enable the console output of the CSS processor from: **Options > Preferences > XML > PDF Output > CSS-based Processors**.

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 only involve 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 1471)*.

How to Use XPath Expressions in CSS

How to Write XPath Expressions

To use XPath expressions in CSS, you need to use the `oxy_xpath()` function. These XPath expressions are used to extract the content from the HTML merged DITA map document.

The following example shows how to display the product name meta-information before the front page title:

```
*[class~="front-page/front-page-title"]:before {
    text-align: left;
    content: oxy_xpath("(//*[contains(@class, 'topic/prodname')]/text())[1]");
    display:block;
}
```

Important:
Do not use the DITA element names directly. You must use the DITA `@class` attribute instead, as these attributes are propagated to the HTML elements while the element names can be lost. By using the class selectors, you also cover DITA specializations.

Tip:
Use the "[1]" XPath predicate to select the first value from the document. Do not forget the parenthesis between the node to be selected.

For example: `oxy_xpath("(//*[contains(@class, 'topic/prodname')]/text())[1]")`.

Note that the meta-information might be copied multiple times in the output, inherited by the `<topicref>` elements, so you might get more values than expected.

Other Notes:

- You can call the `oxy_xpath()` function in `string-set` property.
- You can use content extracted using the `oxy_xpath()` function in both pseudo-elements and @page at-rules.
- Do not use strings as values for pseudo-elements content because they are not supported in them.

How to Debug XPath Expressions

Suppose that you need to display the publication author in the bottom-left part of the cover page.

The ditamap content is the following:

```
<map>
  <title>The Art of Bike Repair</title>
  <topicmeta>
    <author>John Doe</author>
  </topicmeta>
  ...
</map>
```

To debug an XPath expression:

1. Read the XPath Expressions Guidelines *(on page 1475)*.
2. Launch the transformation of the DITA map using your customization CSS.
3. Open the `[MAP_NAME].merged.html` file (from the output folder) in **Oxygen XML Editor/Author**. You will find this inside the HTML:

   ```
   <div class="- front-page/front-page front-page">
     <div class="- map/topicmeta topicmeta">
       <div class="- topic/author author">John Doe</div>
     </div>
     <div class="- front-page/front-page-title front-page-title">
       <div class="- topic/title title">The Art of Bike Repair</div>
     </div>
   </div>
   ```

4. Activate the **XPath Builder** view (**Window > Show View > XPath/XQuery Builder**).
5. Paste your XPath expression (for example: `//*[contains(@class, "front-page/front-page")]/*[contains(@class, "map/topicmeta")]/*[contains(@class, "topic/author")]/text()`) and click the **Execute XPath** button. Check if it returns the expected results.
6. Copy the expression in your customization CSS and define the rules that will use it. For example:

```
:root {
  string-set: author oxy_xpath('//*[contains(@class, "front-page/front-page")]\
  /*[contains(@class, "map/topicmeta")]/*[contains(@class, "topic/author")]/text()');
}

@page front-page {
  @bottom-left {
    content: "Created by " string(author);
  }
}
```

Note:
The "\" character used in the expression allows the multi-line display without breaking the query.

7. Run the transformation again to obtain the desired output.

Note:
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:

```
@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:

```
@page :left, table-of-contents:left, chapter:left {
  @top-left {
    content: "...";
  }
}

@page :right, table-of-contents:right, chapter:right{
  @top-right {
    content: "...";
  }
}
```

Front Page

The cover page. It clears the headers.

```
@page front-page {
    @top-left-corner    {          content:none }
    @top-left           {          content:none }
    @top-center         {          content:none }
    @top-right          {          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 }
}
```

Table of Contents Page

The table of content page. It clears the headers and uses a lower roman page number in the header.

```
@page table-of-contents {
  @top-left      { content: none; }
  @top-center    { content: none; }
  @top-right     { content: none; }
  @bottom-left   { content: none; }
  @bottom-center { content: none; }
  @bottom-right  { content: none; }
}
@page table-of-contents:left {
  @top-left {
    content: string(toc-header) " | " counter(page, lower-roman);
  }
}
@page table-of-contents:right {
  @top-right {
    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:left {
  @top-left {
    content: none;
  }
}
@page table-of-contents:first:right {
   @top-right {
     content: none;
   }
}
```

Chapter Page

The chapter page 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:

```
@page chapter{
  /* Currently inherit from the default page. */
}

/* No headers on the chapter first page. */
@page chapter:first:left{
  @top-left {
    content: none;
  }
}
@page chapter:first:right{
  @top-right {
    content: none;
  }
}
```

Front Matter and Back Matter Page

The bookmap front matter and back matter page. It clears the headers.

```
@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 index page contains the index terms (and appears only if there are such items in your topics). It uses a lower alpha page number in the footer:

```
@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;
  }
}

@media oxygen-chemistry {
  @page index {
    column-count: 2;
    column-fill: auto;
  }
}
```

When transformed, the page layout is spread on two columns.

Landscape Page

The landscape page contains large tables (for example). The page is rotated and the header is moved on the right side of the page:

```
@page landscape-page:right {
  size: landscape;

  @top-left {
    content: none
  }
  @top-center {
    content: none
  }
  @top-right {
    content: none
  }

  @right-bottom {
```

```
    content: string(maptitle) string(parttitle) string(chaptertitle) string(sectiontitle) " | " counter(page);
    transform: rotate(90deg);
    vertical-align: middle;
    text-align: right;
  }
}

@page landscape-page:left {
  size: landscape;

  @top-left {
    content: none
  }
  @top-center {
    content: none
  }
  @top-right {
    content: none
  }

  @right-top {
    content: string(maptitle) string(parttitle) string(chaptertitle) string(sectiontitle) " | " counter(page);
    transform: rotate(90deg);
    vertical-align: middle;
    text-align: left;
  }
}
```

Blank Page

The following example clears the header for the blank pages that may be created by a `break-before`, `break-after`, or by using double side pagination *(on page 1571)*:

```
@page :blank{
  @top-left {
    content: none;
  }
  @top-right {
    content: none;
  }
}
```

Related information

Oxygen PDF Chemistry: The Page Counter

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:

```
@page {
    padding-top:0.2em;
    padding-bottom:0.2em;
    size: letter;
    margin: 1in;
}
```

Note:
This is listed solely for illustration purposes, as the plugin might use something different.

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 1469)*, use:

```
@page {
    size: A4;
    margin: 2cm;
}
```

If you need different margins depending on the page side:

```
@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 1469)*, use:

```
@page {
    size: 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 1471)*.

2. In your customization CSS *(on page 1469)*, 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.

   ```
   @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 string sets defined for publication, part, chapter, and section titles. These string-sets are defined in the numbering CSS *(on page 1545)*:

maptitle

Set to the current map title (on both DITA maps and bookmaps).

parttitle

Set to the current part number and title, prefixed with " | " (only for DITA bookmaps that use parts).

parttitle-no-prefix

Set to the current part number and title (only for DITA bookmaps that use parts).

chaptertitle

Set to the current chapter number and title, prefixed with " | " (on both Shallow and Deep numbering).

chaptertitle-no-prefix

Set to the current chapter number and title (on both Shallow and Deep numbering).

chaptertitle-only

Set to the current chapter title (on both Shallow and Deep numbering).

sectiontitle

Set to the current section number and title, prefixed with " | " (Deep numbering only).

To see where the default page rules are defined, see: Default Page Definitions *(on page 1477)*.

Important:
The `string-set` CSS property is not additive. Matching the same elements will end up breaking the current definitions. A very common use-case is to change one of the "`chaptertitle`" string-sets defined in the default CSS (see How to Change the Header Separators *(on page 1489)*). In this case, the best approach is to copy the default rules into your customization CSS *(on page 1469)*, then alter the property definition by adding your definition to the existing ones or changing the value of the existing ones (without removing them).

Related Information:
Numbering *(on page 1545)*

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 1477)*.

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:

```
@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 1477)* 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:

```
@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 1469)*, add a CSS rule similar to this:

```
@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:

```
@page table-of-contents {
    font-size: 12pt;
    font-family: "Arial";
}
```

Related Information:

How to Change TOC Header *(on page 1561)*

How to Display Chapter's Headers on First Page

By default, the header is not displayed on the first page of each chapter:

```
/* 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:

```
@page chapter:first:left{
  @top-left {
    content: string(maptitle) string(parttitle) string(chaptertitle) string(sectiontitle) " | " counter(page);
  }
}
@page chapter:first:right{
  @top-right {
    content: string(maptitle) string(parttitle) string(chaptertitle) string(sectiontitle) " | " counter(page);
  }
}
```

Tip:

It is also possible to import the `[PLUGIN_DIR]/css/print/p-optional-pages-and-headers.css` stylesheet into your custom CSS.

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: string(maptitle) string(parttitle) string(chaptertitle) string(sectiontitle) " | " counter(page);
  }
  @top-left-corner {
    content: string(maptitle) string(parttitle) string(chaptertitle) string(sectiontitle) " | " counter(page);
    white-space: nowrap;
    text-align:left;
  }
}

@page :right{
  @top-right {
    content: string(maptitle) string(parttitle) string(chaptertitle) string(sectiontitle) " | " counter(page);
  }
  @top-right-corner {
    content: string(maptitle) string(parttitle) string(chaptertitle) string(sectiontitle) " | " counter(page);
    white-space: nowrap;
    text-align: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 1494)*.

How to Change the Header Separators

There are some `strings` defined for parts, chapters, 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
```

- "My Publication" is the value of the `maptitle` string.
- "Introduction" is the value of the `chaptertitle` string.
- "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 the prefix separator. In your customization CSS *(on page 1469)*, add the following CSS rule:

```
*[class ~= "topic/topic"][is-part] > *[class ~= "topic/title"] {
    string-set: parttitle " - " counter(part, upper-roman) " - " content(),
    parttitle-no-prefix " " counter(part, upper-roman) " - " content(),
    chaptertitle "",
    chaptertitle-no-prefix "",
    chaptertitle-only ""; /* Avoid propagating a past chapter title on a new part */
}
*[class ~= "topic/topic"][is-chapter]:not([is-part]) > *[class ~= "topic/title"] {
  string-set: chaptertitle " - " counter(chapter) " - " content(),
  chaptertitle-no-prefix " " counter(chapter) " - " content(),
  chaptertitle-only " " content();
}
```

If you enabled the deep numbering for chapters and subsections *(on page 1550)*, then use:

```
*[class ~= "map/map"][numbering ^= 'deep'] *[class ~= "topic/topic"][is-part] > *[class ~= "topic/title"] {
  string-set: parttitle " - " counter(part, upper-roman) " - " content(),
  parttitle-no-prefix " " counter(part, upper-roman) " - " content(),
  chaptertitle "",
  chaptertitle-no-prefix "",
  chaptertitle-only ""; /* Avoid propagating a past chapter title on a new part */
}
```

```
*[class ~= "map/map"][numbering ^= 'deep'] *[class ~= "topic/topic"][is-chapter]:not([is-part]) > *[class ~= "topic/title"]
 {
  string-set: chaptertitle " - " counters(chapter-and-sections, ".") " - " content(),
  chaptertitle-no-prefix " " counters(chapter-and-sections, ".") " - " content(),
  chaptertitle-only " " content(),
  sectiontitle ""; /* Avoid propagating a past section title on a new chapter */
}
*[class ~= "map/map"][numbering ^= 'deep'] *[class ~= "topic/topic"][is-chapter]:not([is-part]) > *[class ~= "topic/topic"]
 > *[class ~= "topic/title"] {
  string-set: sectiontitle " - " counters(chapter-and-sections, ".") " - " content();
}
```

Note:
Always keep in mind that the `string-set` property is not additive (like the majority of CSS properties) so defining new string-sets using the same selector will override the default values.

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.

```
content: string(maptitle) string(parttitle) string(chaptertitle) 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 1469)*, add the following CSS rule:

```
@page :left {
    @top-left {
        content: string(chaptertitle) " | " counter(page);
    }
}
@page :right{
    @top-right {
        content: string(chaptertitle) " | " 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:

```
@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.

```
@top-right {
    content: string(chaptertitle) " | " counter(page) !important;
}
```

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 1492)*.
- Use two adjacent page margin boxes, and style them differently:

```
@page {
  @top-center {
    content: "First part";
    color: red;
    text-align:right;
  }
  @top-left {
    content: "- Second part";
    color: blue;
    text-align:left;
  }
}
```

- Use the `oxy_label` constructor. This is a function that creates a text label with a set of styles.

```
@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;");
  }
}
```

Notes:

- These functions only work with the Chemistry CSS processor.
- You can combine the `oxy_label` with `oxy_xpath`, to extract and style a piece of text from the document:

```
content: oxy_label(text, oxy_xpath("/some/xpath"), styles, "color:blue; "));
```

- You cannot use `string()` inside an `oxy_label()`. As a workaround, define general styles for the page margin box to affect the dynamic text from `string()` and use `oxy_label` to style the static text differently.

```
@page {
  @top-right {
    color: red;
    content: oxy_label(text, "My Company", styles, "color:black")
             ' '
             string(chaptertitle); /* This inherits the styling from @top-right*/
  }
}
```

How to Add a Background Image to the Header

Using Raster or Vector Images

A common use-case is to add a company or a product logo to one of the page corners.

Important:
Always specify a `content` property. If not, the page margin box will not be generated.

```
@page {
  @bottom-left-corner {
    content: " ";
    background-image: url('https://www.oxygenxml.com/img/resources/oxygen190x62.png');
    background-repeat: no-repeat;
    background-position: 50% 50%;
    background-size: 90%;
  }
}
```

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 1494)*):

```
@page :left {
  @top-left{
      content: '\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.

Using SVG Templates

Additional requirements may present the need to insert dynamic content (such as version or date). To achieve this, you can use SVG Templates (for example, if you need to display the last version in a `<bookmap>`):

```
<bookmap id="taskbook">
    ...
    <bookmeta>
        ...
        <prodinfo>
            <vrmlist>
                <vrm version="1.0"/>
                <vrm version="1.1"/>
            </vrmlist>
        </prodinfo>
        ...
    </bookmeta>
    ...
</bookmap>
```

1. Locate the node you want to extract in the `merged.html` file:

```
<div class="- front-page/front-page front-page">
  <div class="- map/topicmeta bookmap/bookmeta topicmeta bookmeta">
    ...
    <div class="- topic/prodinfo prodinfo">
      <div class="- topic/prodname prodname">Product tasks</div>
      <div class="- topic/vrmlist vrmlist">
        <div version="1.0" class="- topic/vrm vrm"></div>
```

```
        <div version="1.1" class="- topic/vrm vrm"></div>
      </div>
    </div>
    ...
  </div>
</div>
```

2. Create the template that will be set as footer (`footer.template.svg` in our example):

```
<svg xmlns="http://www.w3.org/2000/svg" width="1in" height="1in">
  <text x="0.10in" y="0.70in">Version:
    ${//*[contains(@class, 'bookmap/bookmeta')]
    //*[contains(@class, 'topic/vrmlist')]
    /*[contains(@class, 'topic/vrm')][last()]/@version}</text>
</svg>
```

3. Reference the template in your CSS customization:

```
@page {
  @bottom-left-corner {
    content: " ";
    background-image: url("footer.template.svg");
    background-repeat: no-repeat;
    background-position: 0;
  }
}
```

4. After the transformation processes, you will get **Version: 1.1** in the footer.

Related Information:

Images and Figures *(on page 1628)*
How to Add a Background Image for the Cover *(on page 1513)*
How to Add a Link in Headers and Footers *(on page 1500)*
How to Use XPath Expressions in CSS *(on page 1475)*

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:

```
@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 11in; /* Optional: Adapt to your page size. */
}
```

For a list of all the possible page names, see: Default Page Definitions *(on page 1477)*.

Related Information:

How to Add a Background Image for the Cover *(on page 1513)*

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 is by using the `<topicmeta>` element for the `<topicref>` topic reference:

```
...
 <topicref href="topics/installing.dita">
    <topicmeta>
      <data name="header-data" value="ID778-3211"/>
    </topicmeta>
    ...
```

In the above example, there is 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 1471)*, like this:

```
<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 ">
      ...
      <data class="- topic/data " name="header-data" value="ID778-3211"/>
      ...
```

This information can be extracted from the 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>
 <topicref href="..."> <!-- Uses the same value -->
 <topicref href="..."> <!-- Uses the same value -->
 <topicref href="..."> <!-- Uses the same value -->
 <topicref href="topics/change.dita">
    <topicmeta>
      <data name="header-data" value="ID990-3200"/>
    </topicmeta>
 </topicref>
 <topicref href="..."> <!-- The string set is changed now -->
 <topicref href="..."> <!-- The string set is changed now -->
 <topicref href="..."> <!-- The string set is changed now -->
```

To clear the text, use the `none` value:

```
...
     <topicref href="..."> <!-- The string 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:

```
...
 <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 1471)*, like this:

```
<topic 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:

```
/* 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 the `oxy_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:

```
© 2017 - My Company Ltd
All rights reserved
```

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 1469)*, add the following CSS rule:

```
@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:

```
@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:

   ```
   ...
   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:

```
*[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:

```
*[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 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*:

```
<svg width="180" height="20" viewBox="0 0 180 20" xmlns="http://www.w3.org/2000/svg"
  xmlns:xlink="http://www.w3.org/1999/xlink">
    <a xlink:href="https://www.oxygenxml.com/chemistry-html-to-pdf-converter.html">
        <rect x="0" y="0" width="180" height="20" opacity="0"/>
        <text x="5" y="15" fill="blue">Oxygen PDF Chemistry</text>
    </a>
</svg>
```

This creates an SVG link with **Oxygen 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:

```
@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.

```
@page {
  @top-left {
    content: "Link";
    -oxy-link: "https://www.oxygenxml.com/";
    color:blue;
  }
}
```

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 1469)*:

```
@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:

```
@page table-of-contents:left {
    @top-left {
        color:navy;
        font-style:italic;
    }
    @top-right {
        color:red;
    }
}
```

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(/)) - \
                              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>
   ...
    <data name="cover">
        <image href="product-cover.png" outputclass="cover-image"/>
    </data>
   ...
  </metadata>
</bookmeta>
```

```
@page {
   @top-center {
        content: url("oxy_xpath('//*[contains(@outputclass, "cover-image")]/@href')");
   }
}
```

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`:

```
@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:

```
@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.

```
:root{
       string-set: maptitle oxy_xpath('document(@xtrf)/*[contains(@class, " map/map ")]/*[contains(@class, " topic/title
 ")]/text()');
}
```

Related Information:

Oxygen PDF Chemistry User Guide: Headers and Footers

http://zvon.org/xxl/XPathTutorial/General/examples.html

Oxygen User Guide: oxy_xpath() Function

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:

```
@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{
  content:" ";
  border-bottom: 1px solid black;
  vertical-align: bottom;
}


@top-left-corner{
  content:" ";
  border-bottom: 1px solid black;
  vertical-align: bottom;
}
```



Note:

The `padding-top: 1em;` is used to avoid the border at the bottom of the header that joins with the page content.

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 1492)*.

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 1660)*.

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:

```
@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 1477)* 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:

```
@page front-page,
      table-of-contents,
      table-of-contents:first:left,
      table-of-contents:first:right,
      chapter:first:left,
      chapter:first:right{
    @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 1502)*. You can use this technique to alter the chapter titles as in the following example:

```
*[class ~= "map/map"][numbering^='deep']
   *[class ~= "topic/topic"][is-chapter]:not([is-part]) >
        *[class ~= "topic/title"] {
    string-set:
            chaptertitle " | " 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

Example 1: Adjust the headings according to the publication's language

It is possible to customize the text displayed in the headings depending on the language of the publication (set by the `@xml:lang` attribute on the main Dita map).

For instance, you can display the page counter at the bottom of each page, prefixed by the word 'Page' in the selected language:

```
@page chapter {
  @bottom-center {
    content: oxy_xpath("if (@lang='es') then  'Página' \
                        else if (@lang='it') then 'Pagina' \
                        else 'Page'") " " counter(page);
  }
}
```

Note:
Backslashes (\) are used to split the XPath into multiple lines to make it easier to read.

Example 2: Adjust the headings according to each topic language

It is possible to customize the heading text depending on the language of each topic as well (using the `@xml:lang` attribute).

For example, you can display the language code in the top-right corner of each page, according to each topic language:

```
*[class ~= "topic/topic"] {
  string-set: lang oxy_xpath("if (@lang) then upper-case(@lang) else 'EN'");
}


@page chapter {
  @top-right-corner {
    color: orange;
    content: string(lang);
  }
}
```

How to Display the Chapter and the Page Number in the Footer

It is possible to display the chapter number along with the page number in the footer of each page. For example, a CC-PP (using a 2-digits numbering) display can be done using the following CSS rules:

```
*[class ~= "map/map"] *[class ~= "topic/topic"][is-part] {
  string-set: chapternumber "";
```

```
}
*[class ~= "map/map"] *[class ~= "topic/topic"][is-chapter]:not([is-part]) {
  string-set: chapternumber counter(chapter, decimal-leading-zero);
}
*[class ~= "map/map"] *[class ~= "bookmap/frontmatter"]
*[class ~= "map/map"] *[class ~= "bookmap/backmatter"]
*[class ~= "map/map"] *[class ~= "topic/topic"][is-part] ~ *[class ~= "topic/topic"]:not([is-part]) {
  string-set: chapternumber "";
}
...
@page chapter {
  @bottom-center {
    content: string(chapternumber) "-" counter(page, decimal-leading-zero);
  }
}
```

Page Breaks

The page breaks can be controlled in multiple ways:

- By creating an `@page` and assigning it to an element. This will create a page break between this element and the sibling elements that have a different page.
- By using the CSS properties: `break-before`, `break-after`, or `break-avoid`.
- In your DITA topic, by setting the `@outputclass` attribute on the topic root (or any element) to contain one of the `break-before`, `break-after`, or `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 1571)*

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 `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 1469)*:

```
*[class ~= "topic/table"] {
    break-inside:avoid;
}
```

```
*[class ~= "topic/ol"] {
    break-inside:avoid;
}
*[class ~= "topic/ul"] {
    break-inside:avoid;
}
```

Note:
Since the task steps are inherited from `topic/ol`, they will also not be split over two separate pages. However, if you want to allow this, add the following CSS rule:

```
*[class ~= "task/steps"] {
    break-inside:auto;
}
```

Note:
Another way to do this is to mark the element with an `@outputclass` set to `break-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 1469)*:

```
*[class ~= "map/map"] > *[class ~= "topic/topic"] > *[class ~= "topic/topic"] {
  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:

break-before

Use this for a page break before the marked element.

break-after

Use this for a page break after the marked element.

break-avoid

Use this to avoid page breaks inside the marked element.

For example, to force a page break before a certain topic, use:

```
<topic outputclass="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:

```
<ol outputclass="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 1469)*.

Style the separating blank page:

```
@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:

```
*[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.

```
<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 1469)*:

```
: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 1471)* 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/" ...>
  <oxy:front-page xmlns:oxy="http://www.oxygenxml.com/extensions/author">
   <bookmeta xmlns:dita-ot="http://dita-ot.sourceforge.net/ns/201007/dita-ot"
            ...
   </bookmeta>
   <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>
```

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="- map/map bookmap/bookmap bookmap" ... >
   <div class=" front-page/front-page front-page">
      <div class="- map/topicmeta bookmap/bookmeta boometa">
        ...
      </div>
     <div class=" front-page/front-page-title front-page-title">
       <div class="- topic/title bookmap/booktitle booktitle">
         <div class="- topic/ph bookmap/booklibrary booklibrary">Retro Tools</div>
         <div class="- topic/ph bookmap/mainbooktitle mainbooktitle">Tasks</div>
         <div class="- topic/ph bookmap/booktitlealt booktitlealt">Product tasks</div>
      </div>
...
```

Cover Page - Built-in CSS rules

The element with the class `frontpage/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`.

```
@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";
      font-size:0;
    }

    *[class~="front-page/front-page-title"] {
        display:block;
        text-align:center;
        margin-top:3in;
        font-size:2em;
        font-family:arial, helvetica, sans-serif;
        font-weight:bold;
    }

    @page front-page {
        @top-left-corner    {          content:none }
        @top-left           {          content:none }
        @top-center         {          content:none }
        @top-right          {          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 }
    }
}
```



Note:
This is listed solely for illustration purposes, as the plugin might use something different.

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:

```
<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:

```
<svg xmlns="http://www.w3.org/2000/svg" width="8.5in" height="11in" viewBox="0 0 600 850">
```

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 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 1469)*, add the following:

```
@page front-page {
    background-image: url("us-letter.svg");
    background-position: center;
    background-repeat: no-repeat;
    background-size: 100% 100%;
}
```

For smaller artworks, you can use `background-position` with percentage values to position and center the artwork (for example, a company logo):

```
@page front-page {
    background-image:url("company-logo.svg");
    background-position:50% 5%; /* The first is the alignement 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 Display the Background Cover Image Before the Title

It is possible to split the font-page display into two pages so that the background image appears on one page and the title on another. The solution is to define a new page for the main title:

```
@page front-page {
  @top-left { content: none; }
  @top-right { content: none; }
  @bottom-center { content: none; }

  background-image: url("us-letter.svg");
  background-position: center;
  background-repeat: no-repeat;
  background-size: 100% 100%;
}

@page main-title-page {
  @top-left { content: none; }
  @top-right { content: none; }
  @bottom-center { content: none; }
}

*[class ~= "front-page/front-page-title"]:before {
  display: block;
```

```
  content: "\2002";
  margin-bottom: 3in;
}

*[class ~= "front-page/front-page-title"] {
  page: main-title-page;
}
```

How to Use Different Background Cover Images Based on Bookmap or Map Information

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.

```
@page front-page {
    background-image: url(oxy_xpath("\
        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'\
    "));
    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 1529)*.

Related information

Oxygen PDF Chemistry: Graphics

How to Use SVG Templates for Creating Dynamic Cover Pages *(on page 1523)*

How to Change Styling of the Cover Page Title

Match the front page title element in your customization CSS *(on page 1469)* based on its class attribute:

```
*[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 1469)*, 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:



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 1534)*.

Related Information:

How to Show Metadata in the Cover Page *(on page 1534)*

How to Place Cover on the Right or Left Side

In your customization CSS *(on page 1469)*, add the following CSS rules:

```
*[class ~= "front-page/front-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 1571)*

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:

```
@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;
  break-after: always;
  display: block;
  content: "\2002";
}
```

If you want to add a back cover page, you should use an `:after` pseudo element on the map itself:

```
*[class ~= "map/map"]:after
```

and bind it to another `@page` declaration:

```
@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:

```
<bookmap>
  <booktitle>
    ...
  </booktitle>
  <bookmeta>
    <metadata>
      <data name="second-cover-url" value="covers/second-cover.svg"/>
    </metadata>
  </bookmeta>
...
</bookmap>
```

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:

```
@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 1469)*, add the following CSS rules:

```
@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;
    break-after:always;
}

*[class ~= 'front-page/front-page']:after(2){
    page:my-blank-page;
    display:block;
    content: '\2002';
    break-after:always;
}

*[class ~= 'front-page/front-page']:after(3){
```

```
    page:my-blank-page;
    display:block;
    content: '\2002';
    break-after:always;
}
```

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 1573)*

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 1469)*, declare a new page layout:

```
@page copyright-notice-page {
  /* Clear the headers for the copyright page */
  @top-left {
    content: none;
  }
  @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. */
  content: "Copyright 2018-2019 MyCorp Inc. \A All rights reserved";
  padding-top: 8in; /* Use padding to position the text in the page. */
  text-align: center;
```

```
  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:

```
<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/copyryear ")]/@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 1476)*

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 1469)*:

```
*[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;
}
```

How to Add a Cover in Single-Topic Publishing

It is possible to add a cover page before the topic when publishing a single-topic PDF (without a DITA map) using the DITA PDF - based on HTML5 & CSS transformation scenario.

For example, to add a background image before the published topic, you need to create a new `@page` rule and add it in a block before the actual content of the document:

```
@page topic-cover {
  @top-left {content: none;}
  @top-right {content: none;}

  background-image: url("img/cover.svg");
  background-position: center;
  background-repeat: no-repeat;
  background-size: 100% 100%;
}

:root::before {
  page: topic-cover;
  display: block;
  content: "\2002";
  break-after: always;
}
```

How to Use SVG Templates for Creating Dynamic Cover Pages

It is possible to use XPath expressions inside SVG templates to insert dynamic text when creating PDF output using the DITA Map PDF - based on HTML5 & CSS scenario.

Using SVG Template as a Cover Page

A common use-case is when you want to create a custom cover page and this cover should display metadata information (i.e. the author, dates, and copyright information):

1. In the source `<bookmap>`, the various metadata elements are inserted inside the `<bookmeta>` element:

```
<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="2015-01-01"/>
            <revised modified="2016-04-03"/>
            <revised modified="2016-03-05"/>
        </critdates>
        ...
        <bookrights>
            <copyrfirst>
                <year>2004</year>
            </copyrfirst>
            <copyrlast>
                <year>2007</year>
            </copyrlast>
            <bookowner>
                <organization>Retro Tools, Inc.</organization>
            </bookowner>
        </bookrights>
    </bookmeta>
    ...
```

2. The corresponding `merged.html` file will have the following content:

```
...
<div class="- front-page/front-page front-page">
  <div class="- map/topicmeta bookmap/bookmeta topicmeta bookmeta">
    <div class="- topic/author author">Howe Tuduit</div>
    <div class="- topic/critdates critdates">
      <div date="2015-01-01" class="- topic/created created"></div>
      <div modified="2016-04-03" class="- topic/revised revised"></div>
      <div modified="2016-03-05" class="- topic/revised revised"></div>
    </div>
    ...
    <div class="- topic/data bookmap/bookrights data bookrights">
      <div class="- topic/data bookmap/copyrfirst data copyrfirst">
```

```
        <div class="- topic/ph bookmap/year ph year">2004</div>
      </div>
      <div class="- topic/data bookmap/copyrlast data copyrlast">
        <div class="- topic/ph bookmap/year ph year">2007</div>
      </div>
      <div class="- topic/data bookmap/bookowner data bookowner">
        <div class="- topic/data bookmap/organization data organization">Retro Tools,
          Inc.</div>
      </div>
    </div>
  </div>
  <div class="- front-page/front-page-title front-page-title">
    <div class="- topic/title bookmap/booktitle title booktitle">
      <span class="- topic/ph bookmap/booklibrary ph booklibrary">Retro Tools</span>
      <span class="- topic/ph bookmap/mainbooktitle ph mainbooktitle">Product
        tasks</span>
      <span class="- topic/ph bookmap/booktitlealt ph booktitlealt">Tasks and what they
        can do</span>
    </div>
  </div>
</div>
...
```

3. The cover image (for example, named `cover.template.svg`) should display `<bookmeta>` node information (author, creation date, and copyright information) and the `<mainbooktitle>` will be displayed rotated.

```
<svg version="1.1" id="Layer_1" xmlns="http://www.w3.org/2000/svg"
  xmlns:xlink="http://www.w3.org/1999/xlink" x="0px" y="0px"
  viewBox="0 0 610 790" style="enable-background:new 0 0 610 790;" xml:space="preserve">
<style type="text/css">
  .st0{fill:url(#SVGID_1_);}
  .st1{opacity:0.31;fill:#FFFFFF;enable-background:new    ;}
  .st2{fill:#FFFFFF;}
  .st3{fill:#F04C3E;}
  .st4{fill:none;stroke:#FFFFFF;stroke-width:0.3685;stroke-miterlimit:2.6131;}
  .st5{font-family:'Arial';}
  .st6{font-size:24.3422px;}
  .st7{font-size:10px;}
  .st8{font-size:63.3422px;font-weight:bold;}
  .st9{fill:#F04C3E;stroke:#000000;stroke-miterlimit:10;}
</style>
<linearGradient id="SVGID_1_" x1="305.6" y1="799.9393" x2="305.6" y2="8.9393"
  gradientUnits="userSpaceOnUse">
```

```
  <stop  offset="1.848748e-02" style="stop-color:#2F639F"/>
  <stop  offset="1" style="stop-color:#1C3E72"/>
</linearGradient>
<rect x="0.1" y="0.1" class="st0" width="611" height="791"/>
<path class="st1" d="M143.4,700.5l381.3-381.3c35.2-35.2,35.2-92.3,
  0-127.5L332.1-0.9H0.1v685.6l15.8,15.8C51.1,735.7,108.2,735.7,143.4,700.5z"/>
<path class="st2" d="M1.5,617.6c29.2,22.6,71.4,20.5,98.2-6.3l315.2-315.2c29.1-29.1,
  29.1-76.3,0-105.4L224.8,0.5H1.5V617.6z"/>
<text transform="matrix(1 0 0 1 419.998 615.9277)" class="st2 st5 st6">
  ${//*[contains(@class, 'bookmap/bookmeta')]/*[contains(@class, 'topic/author')]}
</text>
<text transform="matrix(1 0 0 1 419.998 660.9277)" class="st2 st5 st6">
  ${//*[contains(@class, 'bookmap/bookmeta')]//*[contains(@class, 'topic/created')]/@date}
</text>
<text transform="matrix(1 0 0 1 471.998 749.9277)" class="st2 st5 st7">©
  ${
    concat(//*[contains(@class, 'bookmap/bookmeta')]/*[contains(@class, 'bookmap/bookrights')]
    //*[contains(@class, 'bookmap/organization')], ' ',
    //*[contains(@class, 'bookmap/bookmeta')]/*[contains(@class, 'bookmap/bookrights')]
    /*[contains(@class, 'bookmap/copyrlast')]/*[contains(@class, 'bookmap/year')])
  }
</text>
<text transform="matrix(0.7071 -0.7071 0.7071 0.7071 88.1369 568.6693)" class="st9 st5 st8">
  ${
    //*[contains(@class, 'front-page/front-page-title')]
    //*[contains(@class, 'bookmap/mainbooktitle')]
  }
</text>
</svg>
```

Notes:

- XPath expressions are not expanded if the SVG template is open in **Author** mode.
- XPath expressions can be tested (without `${}`) using the XPath/XQuery Builder view.
- XPath *Conditional Expressions*, *For Expressions*, and *Let Expressions* are supported.

Important:

- If you received the SVG image from someone else (e.g. a graphics designer), make sure that the text from the image was not converted to glyph shapes and that it is rendered using the `<text>` element.
- The SVG `<text>` element does not wrap the text if it overflows the image. If you have longer text that needs to be rendered, you might consider using multiple `<text>` elements and more evolved XPath expressions (for example, using the `substring()` function) to place the text on multiple lines.

Tip:

You can ask a designer to fill the image with some placeholders that you can later find and replace with your XPath expressions. In the above SVG, the designer could place the text `Here comes the author`, that you replace with `${//*[contains(@class, 'bookmap/bookmeta')]/*[contains(@class, 'topic/author')]}`:

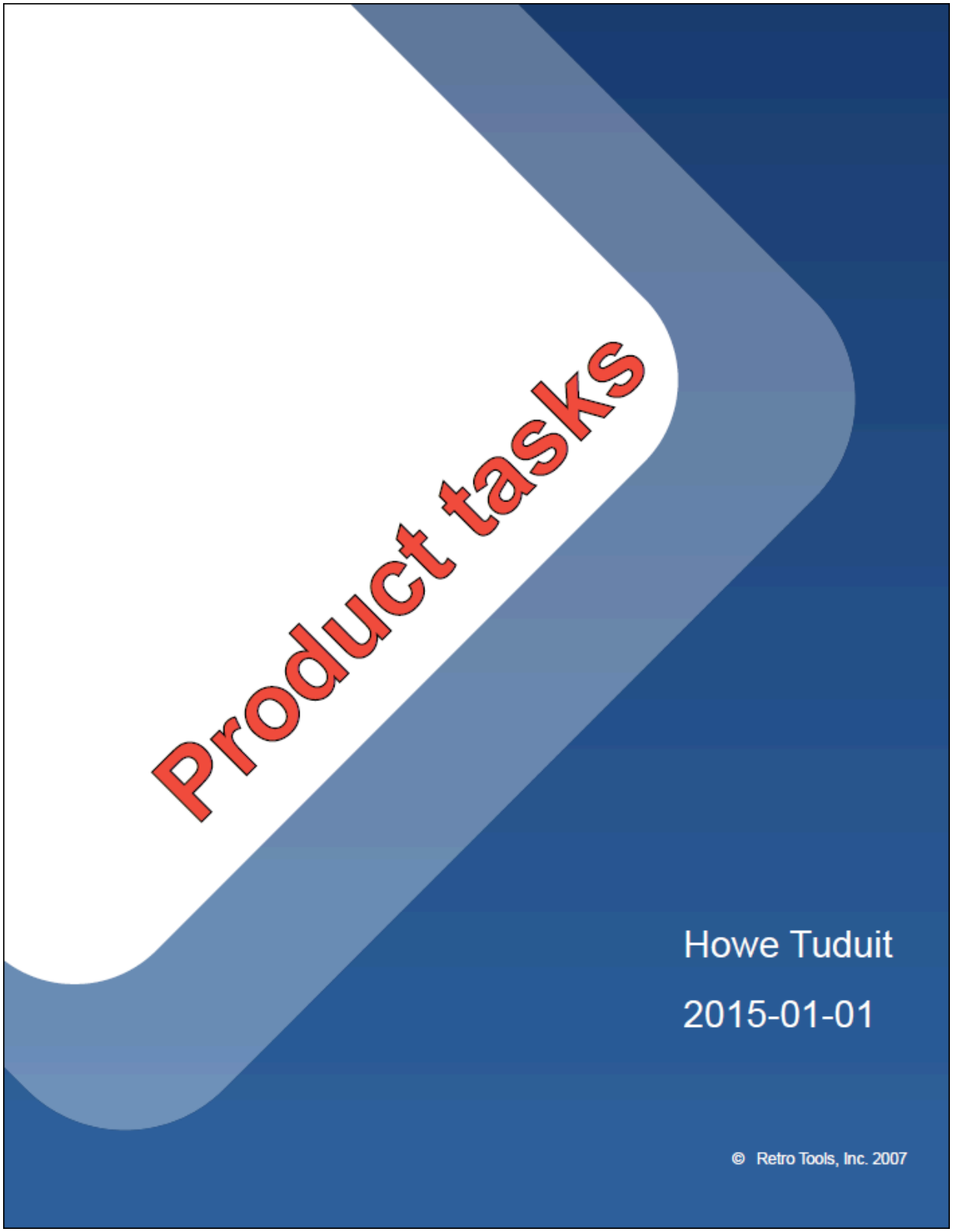
```
<text transform="matrix(1 0 0 1 419.998 615.9277)" class="st2 st5 st6">
  Here comes the author
</text>
```

4. The CSS stylesheet should declare the template file as a background-image for the cover *(on page 1513)*. Also the following example hides the `<mainbooktitle>` and its bookmark (it is displayed in the template):

```
@page front-page {
  background-image: url("cover.template.svg");
  background-repeat: no-repeat;
  background-size: 100% 100%;
}

*[class ~= "bookmap/booktitle"] {
  display: none;
}
*[class ~= "front-page/front-page-title"]
> *[class ~= "bookmap/booktitle"]
> *[class ~= "bookmap/mainbooktitle"] {
  bookmark-level: 0;
}
```

5. After the transformation, the final document cover will look like this:



Related information

How to Use XPath Expressions in CSS *(on page 1475)*

SVG Templates

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 1471)*, the metadata section is placed inside the `<oxy:front-page>` element. This is different from the original placement in the map or bookmap (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`:

```
<bookmap xmlns:ditaarch="http://dita.oasis-open.org/architecture/2005/"
  xmlns:opentopic-index="http://www.idiominc.com/opentopic/index" cascade="merge"
    class="- map/map bookmap/bookmap "
  ditaarch:DITAArchVersion="1.3" >


  <oxy:front-page xmlns:oxy="http://www.oxygenxml.com/extensions/author">


   <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>
        <maintainer class="- topic/data bookmap/maintainer ">
        <organization class="- topic/data bookmap/organization ">ACME Tools</organization>
        <person class="- topic/data bookmap/person "/>
       </maintainer>
     </bookid>
     <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>
            ...
...
```

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="- map/map bookmap/bookmap bookmap" ... >


   <div class=" front-page/front-page front-page">


   <div
     class="- map/topicmeta bookmap/bookmeta boometa">
     <div class="- topic/author author">Howe Tuduit</div>
     <div class="- topic/data bookmap/bookid bookid">
       <div class="- topic/data bookmap/isbn isbn">071271271X</div>
       <div class="- topic/data bookmap/booknumber booknumber">SG99-9999-00</div>
       <div class="- topic/data bookmap/maintainer maintainer">
          <div class="- topic/data bookmap/organization organization">ACME Tools</div>
          <div class="- topic/data bookmap/person person"/>
        </div>
      </div>
      <div class="- topic/data bookmap/bookrights bookrights">
        ...
         <div class="- topic/data bookmap/bookowner bookowner">
              <div class="- topic/data bookmap/organization organization">
                                      ACME Tools, Inc.
              </div>
         </div>
       </div>
     </div>


     <div class=" front-page/front-page-title front-page-title">
            ...
...
```

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:

```
<map xmlns:ditaarch="http://dita.oasis-open.org/architecture/2005/"
    xmlns:opentopic-index="http://www.idiominc.com/opentopic/index"
          cascade="merge" class="- map/map "
    ditaarch:DITAArchVersion="1.3">
    ...

<oxy:front-page xmlns:oxy="http://www.oxygenxml.com/extensions/author">

  <topicmeta class="- map/topicmeta ">
      <author class="- topic/author ">Dan C</author>
      <metadata class="- topic/metadata ">
         <prodinfo class="- topic/prodinfo ">
             <prodname class="- topic/prodname ">oXygen PDF CSS DITA Plugin</prodname>
         </prodinfo>
      </metadata>
      <audience class="- topic/audience "/>
  </topicmeta>
...
```

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 xmlns:ditaarch="http://dita.oasis-open.org/architecture/2005/"
    xmlns:opentopic-index="http://www.idiominc.com/opentopic/index"
          cascade="merge" class="- map/map "
    ditaarch:DITAArchVersion="1.3">
    ...

<div class=" front-page/front-page front-page">

  <div class="- map/topicmeta topicmeta">
      <div class="- topic/author author">Dan C</div>
      <div class="- topic/metadata metadata">
         <div class="- topic/prodinfo prodinfo">
             <div class="- topic/prodname prodname">oXygen PDF CSS DITA Plugin</div>
         </div>
      </div>
      <div class="- topic/audience audience"/>
  </topicmeta>
...
```

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:

```
<bookmap>
  ...
  <bookmeta>
    <keywords>
      <keyword>web server</keyword>
      <keyword>hard disk</keyword>
    </keywords>
  </bookmeta>
```

Maps

If you want your keywords to appear inside a map, you need to define them inside the `<topicmeta>` element:

```
<map>
  ...
  <topicmeta>
    <keywords>
      <keyword>flowers</keyword>
      <indexterm>care and preparation</indexterm>
      <keyword>seasons</keyword>
    </keywords>
  </topicmeta>
```

Topics

If you want your keywords to appear inside one or more topics, you need to define them inside the `<prolog>` element:

```
<topic>
  ...
  <prolog>
    <metadata>
      <keywords>
        <indexterm>iris</indexterm>
      </keywords>
    </metadata>
  </prolog>
```

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:

```
<map>
    ...
    <topicmeta>
        ...
        <audience type="programmer" job="programming" experiencelevel="expert"/>
```

To collect the `@type` attribute, add the following in your customization CSS *(on page 1469)*:

```
*[class ~= "map/map"] > *[class ~= "map/topicmeta"] > *[class ~= "topic/audience"] {
    -oxy-pdf-meta-custom: "Audience" attr(type);
}
```

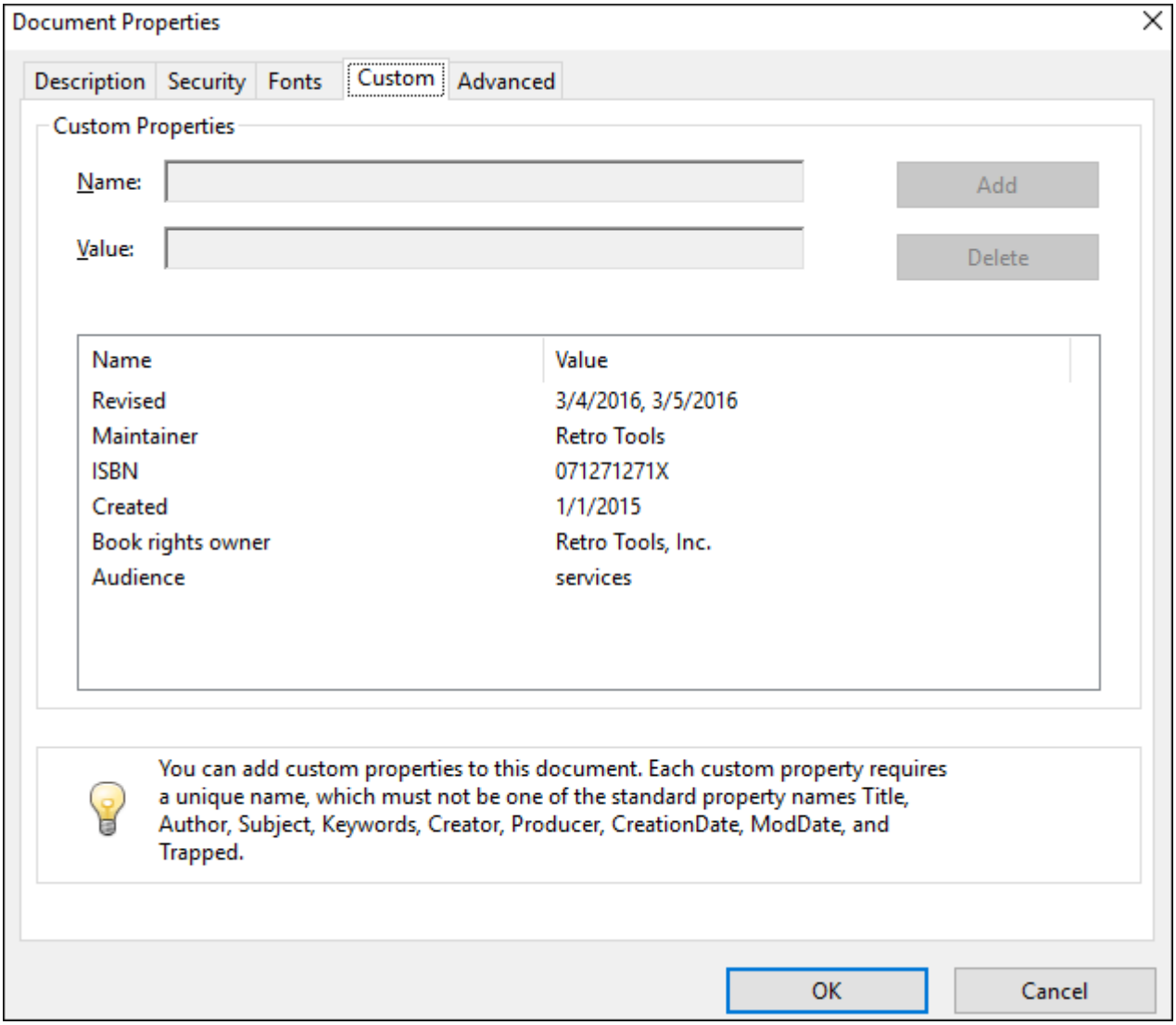
Notice:
It is best to use the class selector (such as `*[class ~= "map/topicmeta"]`) instead of `topicmeta` to cover cases where the elements are specialized (for instance, in a bookmap the `bookmeta` is a `topicmeta`, so your selector will also function for bookmaps, not only simple maps.

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:

```
-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 1535)*, 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:

```
<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>
...
```

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 1469)*, add the following CSS rules:

```
*[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:

```
<map>
  <title>Growing Flowers</title>
  <topicmeta>
    <critdates>
      <revised modified="2021-04-26"/>
    </critdates>
...
```

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 1469)*:

```
*[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 1469)*:

```
@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 {
```

```
    page: page-for-meta;
}
*[class ~= "bookmap/booktitle"]:after(2) {
    page: page-for-meta;
}
```

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 1469)*:

```
*[class ~= "front-page/front-page"] >
  *[class ~= "map/topicmeta"] >
    *[class ~= "topic/critdates"] >
      *[class ~= "topic/created"]{
    string-set: mapcreated attr(date);
}



@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 Show Metadata Information (Revision History) in the Topic Prologue

This topic explains how to present metadata information that is normally hidden in the published output. For the example that follows, this will be the revision history list:

```
<task id="task_3ml_qm3_rf">
    <title>Removing the battery</title>
    <shortdesc/>
    <prolog>
        <change-historylist>
            <change-item>
                <change-revisionid>abd3</change-revisionid>
                <change-completed>Build no 1.</change-completed>
```

```
            </change-item>
            <change-item>
                <change-revisionid>bc72</change-revisionid>
                <change-completed>Build no 2.</change-completed>
            </change-item>
        </change-historylist>
....
```

By default, the `<prolog>` element is hidden (`display:none`) and has several properties that make it collapse, even if the display property is changed.

- It has a transparent color.
- The font has size zero.
- The width and height values are zero.

Start by resetting the prolog properties, but only for prologs that contains a history list. The others will be kept hidden.

```
*[class~="topic/prolog"]:has(*[class~="relmgmt-d/change-historylist"] ){
    display:block;
    color:inherit;
    font-size:1rem;
    width:auto;
    height:auto;
}
```

Next, the following will keep the children of the prolog hidden (other than the change history):

```
*[class~="topic/prolog"]:has(*[class~="relmgmt-d/change-historylist"]) > *:not([class~="relmgmt-d/change-historylist"]) {
    display:none;
}
```

The `<change-item>`, `<change-revisionid>`, and `<change-completed>` (like the descendents of the `<change-historylist>`) are specializations of the `topic/data` element and are also hidden in the output, so you need to make them visible. In the following selector, you can add more classes, depending on what elements you want to be visible.

```
*[class~="relmgmt-d/change-item"],
*[class~="relmgmt-d/change-revisionid"],
*[class~="relmgmt-d/change-completed"] {
    display:block;
}
```

Now some styling for the entire list:

```
*[class~="relmgmt-d/change-historylist"] {
    font-size:1rem;
    border: 3pt solid silver;
```

```
    padding: 0.5em;
}
*[class~="relmgmt-d/change-historylist"]:before {
    content: "Revision History:";
    font-weight:bold;
}
```

And the child elements:

```
/* Example of styling some of the descendends of the history list. */
*[class~="relmgmt-d/change-item"] {
    margin:1em;
}
*[class~="relmgmt-d/change-revisionid"]:before {
    content: "Revision ID: " !important;
    font-weight:bold;
}
*[class~="relmgmt-d/change-completed"]:before {
    content: "Completed: "  !important;
    font-weight:bold;
}
```

In the output, the history list is now visible:



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 1469)*:

```
: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:

```
: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 1469)*:

```
/*
 * 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:

```
<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 1469)*:

```
:root {
    -oxy-pdf-meta-title: oxy_xpath('(//*[contains(@class, "bookmap/booktitlealt")])[1]/text()');
```

```
    -oxy-pdf-meta-description: "";
}
```

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:

```
*[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:

```
*[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:

  ```
  :root {
    -oxy-pdf-viewer-hide-menubar: true;
    -oxy-pdf-viewer-hide-toolbar: true;
  }
  ```

- To make the document be displayed with a different zoom level:

  ```
  :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):

  ```
  :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:

  ```
  :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:

```
: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 1471)*, 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"`:

```
<bookmap xmlns:ditaarch="http://dita.oasis-open.org/architecture/2005/" ...>
    <oxy:front-page class="- front-page/front-page ">
       ...
    </oxy: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">
          ...
        </frontmatter>
    </opentopic:map>
    <concept
           class="- topic/topic concept/concept "
           is-frontmatter="true"
           topicrefclass="- map/topicref bookmap/bookabstract " ...>
```

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 xmlns:ditaarch="http://dita.oasis-open.org/architecture/2005/" ...>
    <div class=" front-page/front-page front-page">
```

```
    ...
  </div>
  <div class="- toc/toc toc">
    <div class="- map/topicref bookmap/frontmatter topicref frontmatter">
      <div href="#unique_2" type="topic" class="- map/topicref topicref">
      ...
      </div>
  </div>
  <article
      class="- topic/topic concept/concept topic concept nested0"
      is-frontmatter="true"
      topicrefclass="- map/topicref bookmap/bookabstract "  ...>
```

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 1485)*

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 1469)*:

```
*[class ~= "map/map"] > *[class ~= "topic/topic"][is-frontmatter]{
  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>` bookmap elements are formatted using the `matter-page` as defined in Default Page Definitions *(on page 1477)*. 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:

```
*[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:

```
@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`
- `[PLUGIN_DIR]/css/print/p-numbering-appendix.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 1550)*.

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/" ...>
  <oxy:front-page xmlns:oxy="http://www.oxygenxml.com/extensions/author"
        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>
            ...
        </topicmeta>
      </topicref>
```

```
      ...
  </opentopic:map>
...
</map>
```

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.

```
<div class="- map/map map" ...>
  <div
      class=" front-page/front-page front-page">
      ...
  </div>
  <div class=" toc/toc toc">
    <div class="- topic/title title">Publication Title</title>

    <div is-chapter="true" class="- map/topicref topicref" ... >
      <div class="- map/topicmeta topicmeta" ... >
        <div href="#unique_1" class="- topic/navtitle navtitle">Overview</div>
          ...
      </div>
      <div class="- map/topicref " ...>
        <div class="- map/topicmeta " data-topic-id="dcpp_resources">
          <div href="#unique_2" class="- topic/navtitle ">Resources</div>
          ...
        </div>
      </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-numbering-shallow.css`/`[INSTALLATION_DIR]/css/print/p-numbering-deep.css` (depending on your numbering scheme - default is shallow) and used 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.

```
<oxy:front-page xmlns:oxy="http://www.oxygenxml.com/extensions/author">
        <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:

```
<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:

```
<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 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>
    </topic>
```

For the **DITA Map PDF - based on HTML5 & CSS** transformations:

```
<div class=" map/map map" ...>

    ...

   <div class=" toc/toc toc">

      ...

    </div>


    <div is-chapter="true" oid="dcpp_overview" class="- topic/topic topic">

        <div class="- topic/title title">Overview</title>

        <div class="- topic/body body">

            ...

        </div>

        <div class="- topic/topic topic" id="unique_2" oid="dcpp_resources">

            <div class="- topic/title title">Resources</div>

            ...

        </div>

        <div class="- topic/topic topic" id="unique_2" oid="dcpp_parameters">

            <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:

```
<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:

```
<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 39. Types of Numbering

Value	Chapters	Sections/ Nested Topics	Figures & Tables	Pages
shallow	num-bered	no	counted from the start of the publi-cation	from the start of the pub-lication
deep	num-bered	numbered	counted from the start of the publi-cation	from the start of the pub-lication
deep-chapter-scope	num-bered	numbered	numbering is restarted at the be-ginning 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 sec-tions	restarted at the beginning of each chapter
deep-chap-ter-scope-no-page-reset	num-bered	numbered	numbering is restarted at the be-ginning 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 sec-tions	from the start of the pub-lication

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
```

```
    Snowdrop.....................................................................6
  ...


List of Figures
  Figure 1: Iris


List of Tables
  Table 1: Flowers
```

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
```

```
List of Tables

  Table 2-1: Flowers
```

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
```

```
List of Tables

  Table 2-1: Flowers
```

Tip:
When using deep numbering, if you want to exclude sections from being numbered, see How to Include Topic Sections in TOC *(on page 1556)*.

How to Reset Page Numbering at First Chapter/Part

By default, pages are numbered from the start of the publication, but in some cases, you may need to restart the page numbering at the first chapter of your publication.

Warning:
The following sections do not apply for `args.css.param.numbering="deep-chapter-scope"` because it already define a specific numbering scheme that resets the page number at each chapter.

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 1469)*:

```
*[class ~= "map/map"] > *:not([class ~= "topic/topic"][is-chapter]) + *[class ~= "topic/topic"][is-chapter] {
  counter-reset: page 1;
}
*[class ~= "map/map"] > *:not([class ~= "topic/topic"][is-part]) + *[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 1469)*:

```
*[class ~= "map/map"][numbering ^= 'deep'] > *:not([class ~= "topic/topic"][is-chapter]) + *[class
 ~= "topic/topic"][is-chapter] {
  counter-reset: page 1 section1;
}
*[class ~= "map/map"][numbering ^= 'deep'] > *:not([class ~= "topic/topic"][is-part]) + *[class ~= "topic/topic"][is-part] {
  counter-reset: page 1 chapter chapter-and-sections;
}
```

Reset Page Numbering in Deep Chapter Scope No Page Reset Context

To reset the page counter at the first part/chapter when the `args.css.param.numbering="deep-chapter-scope-no-page-reset"` parameter value is set, use the following rules in your customization CSS *(on page 1469)*:

```
*[class ~= "map/map"][numbering ^= 'deep'] > *:not([class ~= "topic/topic"][is-chapter]) + *[class
 ~= "topic/topic"][is-chapter] {
  counter-reset: page 1 section1 tablecount figcount !important;
}
*[class ~= "map/map"][numbering ^= 'deep'] > *:not([class ~= "topic/topic"][is-part]) + *[class ~= "topic/topic"][is-part] {
  counter-reset: page 1 chapter chapter-and-sections section1 tablecount figcount !important;
}
```

How to Use Part, Chapter, and Subtopics Numbers in Links

This topic is applicable if you have enabled deep numbering *(on page 1550)*. 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:

```
*[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):

```
*[class ~= "topic/xref"]:after {
 content: "[" target-counter(attr(href), part, upper-roman) "/" target-counters(attr(href),
 chapter-and-sections, ".") "]" !important;
 color:blue;
}
```

Related Information:

Numbering Types *(on page 1550)*

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 1444)* to **yes**. In this case, they are numbered according the numbering scheme set by the `args.css.param.numbering` parameter *(on page 1550)*.

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>`):

```
<bookmap>
  ...
  <frontmatter>
    <booklists>
      <toc/>
      <figurelist/>
      <tablelist/>
    </booklists>
  </frontmatter>
  ...
  ...
```

Related Information:

Table of Contents on a Page (Mini TOC) *(on page 1564)*
List of Tables/Figures *(on page 1569)*
Index *(on page 1578)*

Table of Contents - XML Fragment

In the merged map file *(on page 1471)*, 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.

```
<bookmap ...>

<oxy:front-page> ... </oxy:front-page>
<oxy:front-matter> ... </oxy:front-matter>

<opentopic:map xmlns:opentopic="http://www.idiominc.com/opentopic" class="- toc/toc ">

  <oxy:toc-title xmlns:oxy="http://www.oxygenxml.com/extensions/author" empty="true"
      class="- toc/title "/>

  <booktitle 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>


  <chapter is-chapter="true"
```

```
      class="- map/topicref bookmap/chapter " href="#unique_5" type="topic">
      <topicmeta class="- map/topicmeta " data-topic-id="installing">
          <navtitle href="#unique_5" class="- topic/navtitle ">Installing</navtitle>
          ...
      </topicmeta>

      <topicref class="- map/topicref " href="#unique_6" type="task">
          <topicmeta class="- map/topicmeta " data-topic-id="installstorage">
              <navtitle href="#unique_6" class="- topic/navtitle ">Installing</navtitle>
                  ...
          </topicmeta>
               ...
      </topicref>
           ...
  </chapter>
```

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="- bookmap/bookmap map/map map bookmap" ...>

<div class="- front-page/front-page front-page"> ... </div>
<div class="- bookmap/frontmatter frontmatter"> ... </div>

<div class=" toc/toc toc">

 <div class="toc/toc-title toc-title" empty="true"/>

 <div class="- topic/title bookmap/booktitle booktitle">
      <div class="- topic/ph bookmap/booklibrary booklibrary">Retro Tools</div>
      <div class="- topic/ph bookmap/mainbooktitle mainbooktitle">Tasks</div>
      <div class="- topic/ph bookmap/booktitlealt booktitlealt">Product Tasks</div>
 </div>


 <div is-chapter="true"
  class="- map/topicref bookmap/chapter topicref chapter " href="#unique_5" type="topic">
  <div class="- map/topicmeta topicmeta" data-topic-id="installing">
      <div href="#unique_5" class="- topic/navtitle navtitle">Installing</div>
      ...
 </div>
```

```
<div class="- map/topicref  topicref chapter " href="#unique_6" type="task">
     <div class="- map/topicmeta topicmeta" data-topic-id="installstorage">
         <div href="#unique_6" class="- topic/navtitle navtitle">Installing</div>
             ...
 </div>
             ...
</div>
          ...
</div>
```

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 1485)*

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 1559)*) 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/topicref"] > *[class ~= "map/topicmeta"]:before {
  content: counters(toc-chapter-and-sections, ".") ". ";
}
```

How to Style TOC 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 1469)*, 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 TOC Header

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:

```
@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 1469)*:

```
@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:

```
@page table-of-contents:first:left {
  @top-left {
    content: string(toc-header) " | " counter(page, lower-roman);
  }
}
@page table-of-contents:first:right {
   @top-right {
     content: string(toc-header) " | " counter(page, lower-roman);
   }
}
```

Related information

Localization *(on page 1702)*

How to Make the TOC Start on an Odd Page

In your customization CSS *(on page 1469)*, add the following snippet for the *table-of-contents* page:

```
@page table-of-contents{
    -oxy-initial-page-number: auto-odd;
}
```

Related Information:

Double Side Pagination *(on page 1571)*

How to Display a Topic Before the TOC

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.

Related Information:

Controlling the Publication Content *(on page 1662)*

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 Display Subtopics in TOC

By default, subtopics (nested `<topic>` elements) are not displayed in the Table of Contents. To be displayed, they must be referenced in the DITA map:

```
<map>
  <title>Map</title>
  <topicref href="topic.dita">
    <topicref href="topic.dita#nested1">
      <topicref href="topic.dita#nested11"/>
    </topicref>
    <topicref href="topic.dita#nested2"/>
  </topicref>
</map>
```

where `nestedX` is the subtopic's `@id` value.

Note:

By adding the subtopics in the DITA map, they will also appear in the PDF bookmarks.

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 1450)*.
- 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 1444)*

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 1469)*, add the following CSS selector:

```
*[class ~= "map/topicref"][is-chapter = "true"] > *[class ~= "map/topicmeta"] > *[class ~= "map/shortdesc"] {
  display:block; /* The default is none - the shortdesc is hidden. */
  color:gray;
}
```

Note:
If you need all the TOC item short descriptions to be visible, remove the `[is-chapter]` condition.

Table of Contents on a Page (Mini TOC)

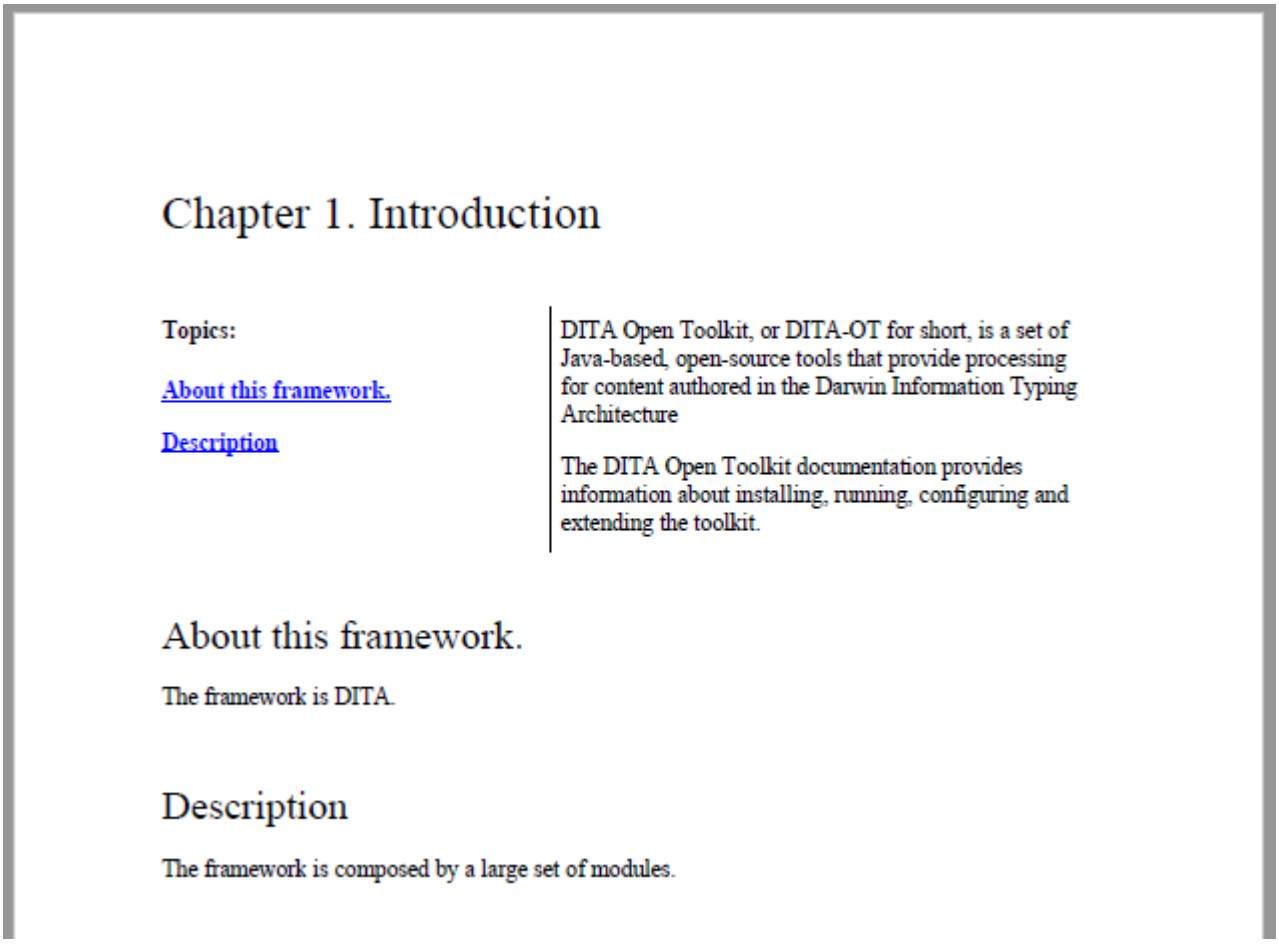
To add a mini table of contents for each chapter, you need to:

- 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.

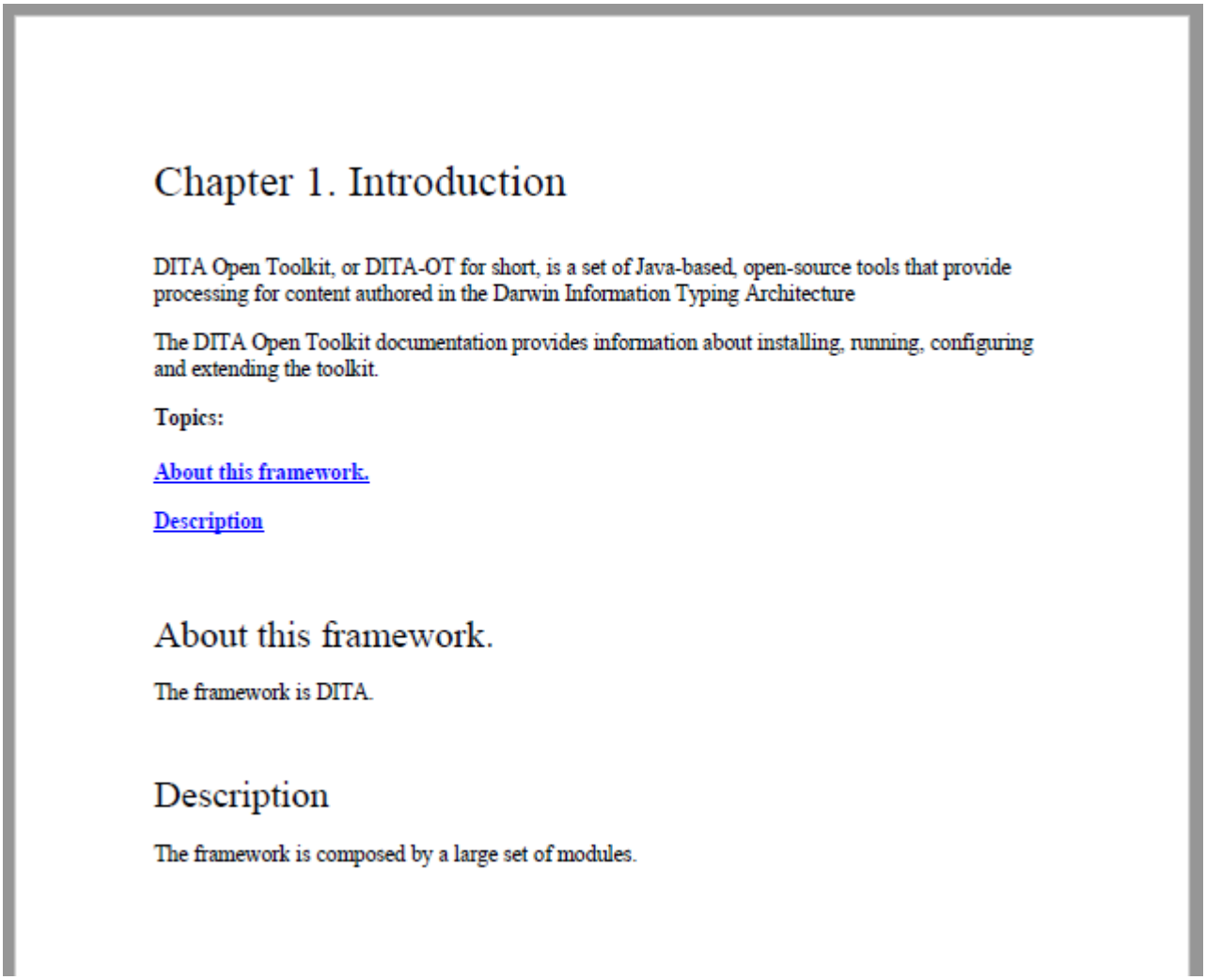
Layout for MINITOC

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.



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.



The above chapter example has the following DITA map fragment:

```
<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.

```
<div class="- topic/div chapter/minitoc ">
  <div class="- topic/div chapter/minitoc-links ">
    <related-links class="- topic/related-links ">
      <linklist class="- topic/linklist ">
        <desc class="- topic/desc ">
          <ph class="- topic/ph chapter/minitoc-label ">Topics: </ph>
        </desc>
        <link class="- topic/link " href="#unique_2" type="topic" role="child">
          <linktext class="- topic/linktext ">About this framework.</linktext>
        </link>
        <link class="- topic/link " href="#unique_3" type="topic" role="child">
          <linktext class="- topic/linktext ">Description</linktext>
        </link>
      </linklist>
    </related-links>
  </div>
  <div class="- topic/div chapter/minitoc-desc ">
    <shortdesc class="- topic/shortdesc ">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</shortdesc>
    <body class="- topic/body ">
      <p class="- topic/p ">The DITA Open Toolkit documentation provides information about
        installing, running, configuring and extending the toolkit.</p>
    </body>
  </div>
</div>
```

When using the `pdf-css-html5` transformation, this structure is converted to a set of HTML elements, preserving the class values:

```
<div class="- topic/div chapter/minitoc div minitoc">
 <div class="- topic/div chapter/minitoc-links div minitoc-links">
  <div class="wh_related_links">
   <nav role="navigation" class="- topic/related-links related-links">
    <div class="- topic/linklist linklist linklistwithchild">
     <div class="- topic/desc desc">
      <span class="- topic/ph chapter/minitoc-label ph minitoc-label">Topics: </span>
      </div>
```

```
        <ul class="linklist">
         <li class="- topic/link link ulchildlink" href="#unique_2"
           type="topic" role="child">
           <strong>
             <a href="#unique_2">About this framework.</a>
           </strong>
           <br/>
         </li>
         <li class="- topic/link link ulchildlink" href="#unique_3"
           type="topic" role="child">
           <strong>
             <a href="#unique_3">Description</a>
           </strong>
           <br/>
         </li>
        </ul>
       </div>
      </nav>
    </div>
  </div>
  <div class="- topic/div chapter/minitoc-desc div minitoc-desc">
    <div class="- topic/body body">
      <p class="- topic/shortdesc shortdesc">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</p>
      <p class="- topic/p p">The DITA Open Toolkit documentation provides information
        about installing, running, configuring and extending the toolkit.</p>
    </div>
  </div>
</div>
```

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 1469)*, 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 1474)*

List of Tables/Figures

To activate these:

1. The map must be a DITA bookmap.
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):

```
<frontmatter>
   <booklists>
     <toc/>
     <figurelist/>
     <tablelist/>
   </booklists>
 </frontmatter>
```

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 {
```

```
    @top-left { content: none; }
    @top-center { content: "Table List"; }
    @top-right { content: none; }
}

@page tablelist:first {
    @top-left { content: none; }
    @top-center { content: none; }
    @top-right { content: none; }
}
```

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:

```
<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:

```
*[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:

```
*[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 1469)*.

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 1469)*, add the following:

```
@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 1469)*:

```
@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.

Related Information:
How to Add a Background Image for the Cover *(on page 1513)*

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 1469)*, use the `-oxy-force-page-count` property with an even value:

```
@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 1469)*, add the following:

```
@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 1477)*.

Next, add the `column-count` and `column-gap` properties to that page. For example:

```
@page chapter{
  column-count:2;
  column-gap:1in;
}
```

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:

```
*[class ~= "topic/topic"][is-chapter] > *[class ~= "topic/title"] {
  column-span:all;
}
```

Limitation:
You cannot use multiple column configurations on the same page. Oxygen XML Developer 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 for just one topic, you can use the following technique:

1. Define an `outputclass` on the topic root element.

   ```
   <topic outputclass="two_columns" ...
   ```

2. Define a CSS rule that changes the `page` property for the matching element.

   ```
   *[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.

   ```
   @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.

Change Column Breaks for Headings

If you need to start each topic on a new column, you can use the following technique:

Suppose you have the following map:

```
<map>
    <title>Map</title>
    <topicref href="first.dita">
        <topicref href="second.dita"/>
    </topicref>
    <topichead navtitle="Topichead">
        <topicref href="second.dita"/>
    </topichead>
</map>
```

You can use the following rules to get the chapter on the new column display:

```
@page {
  column-count: 2;
}
*[class ~= "topic/topic"] *[class ~= "topic/topic"] > *[class ~= "topic/title"] {
  -oxy-column-break-before: always;
}
*[class ~= "topic/title"] + *[class ~= "topic/topic"] > *[class ~= "topic/title"] {
  -oxy-column-break-before: auto;
}
```

Each topic will be displayed on a new column except for topics that only have a title and no content.

Related Information:

Page Formatting in Oxygen PDF Chemistry

Multiple Page Formatting in Oxygen PDF Chemistry

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: `[PLUGIN_DIR]/css/print/p-bookmarks.css`.

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 `<navtitle>` element in the topic reference in the DITA map:

```
...
<topicref href="topics/my_topic.dita" locktitle="yes">
   <topicmeta>
      <navtitle>Introduction</navtitle>
   </topicmeta>
</topicref>
...
```

> **Note:**
> As a best practice, a `@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 `<navtitle>` element in the topic, as a title alternative.

```
<topic id="other_topic" xml:lang="en-us">
  <title>Normal Title</title>
  <titlealts>
    <navtitle>Navigation Title</navtitle>
  </titlealts>
  <body>
...
```

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 1469)*:

```
*[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/title"],
*[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"] {
  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 1469)*:

```
*[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/topic"] *[class ~= "topic/topic"] *[class ~= "topic/section"] > *[class
 ~= "topic/title"],
*[class ~= "topic/topic"] *[class ~= "topic/topic"] *[class ~= "topic/topic"] *[class ~= "topic/topic"] *[class
 ~= "topic/section"] > *[class ~= "topic/title"],
*[class ~= "topic/topic"] *[class ~= "topic/topic"] *[class ~= "topic/topic"] *[class ~= "topic/topic"] *[class
 ~= "topic/topic"] *[class ~= "topic/section"] > *[class ~= "topic/title"],
*[class ~= "topic/topic"] *[class ~= "topic/topic"] *[class ~= "topic/topic"] *[class ~= "topic/topic"] *[class
 ~= "topic/topic"] *[class ~= "topic/topic"] *[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 1469)*.

For example, to specify that all bookmarks for the first three levels are opened (expanded) in the initial state, use:

```
*[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 1469)*, add the following CSS rules:

```
*[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:

```
*[class~="topic/topic"]:has(*[class~="topic/titlealts"]) > *[class~="topic/title"] {...}
```

Related Information:

Numbering *(on page 1545)*

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):

```
<title>The topic title.</title>
<prolog>
  <metadata>
    <keywords>
      <indexterm>Installing <indexterm>Water Pump</indexterm></indexterm>
    </keywords>
  </metadata>
</prolog>
<body>
  .....
```

or in the content itself:

```
...
<p>Open the lid then turn the body pump to the right.
<indexterm>Installing <indexterm>Water Pump</indexterm></indexterm>
</p>
...
```

If you are using a bookmap, you need to specify where the index list should be presented (for instance in the *backmatter* of the book. Technically, it is possible to also add it to the *frontmatter*, but this is unusual). This is done using an `<indexlist>` element in the `<booklists>` element (inside the `<backmatter>`):

```
<bookmap>
   ...
```

```
    <chapter href="tasks/troubleshooting.dita">
      ...
    </chapter>
    <backmatter>
        <booklists>
            <indexlist/>
        </booklists>
    </backmatter>
</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 1471)*, the structure that holds the index tree is the `<opentopic-index:index.groups>` element.

```
<map class="- map/map " >
    <oxy:front-page>
        ...
    </oxy:front-page>
    <opentopic:map xmlns:opentopic="http://www.idiominc.com/opentopic">
        ...
    </opentopic:map>
    <topic class="- topic/topic ">
        <title class="- topic/title ">Request Support</title>
       ...
    </topic>
    <opentopic-index:index.groups id="d16e5548">
      ...
    </opentopic-index:index.groups>
</map>
```

Each of the groups contain:

- A label, the starting letter ("T" in the following example).
- A tree of `<opentopic-index:index.entry>` elements.

```
<opentopic-index:index.group>

 <opentopic-index:label>T</opentopic-index:label>

  <opentopic-index:index.entry value="table of contents">
    <opentopic-index:formatted-value>table of contents</opentopic-index:formatted-value>
```

```
    <opentopic-index:refID value="table of contents:">
        <oxy:index-link xmlns:oxy="http://www.oxygenxml.com/extensions/author"
               href="#d16e3988"> [d16e3988]
        </oxy:index-link>
    </opentopic-index:refID>
    <opentopic-index:index.entry value="change header">
        <opentopic-index:formatted-value>change header</opentopic-index:formatted-value>
        <opentopic-index:refID value="table of contents:change header:">
           <oxy:index-link xmlns:oxy="http://www.oxygenxml.com/extensions/author"
                href="#d16e4176">
                [d16e4176] </oxy:index-link>
        </opentopic-index:refID>
    </opentopic-index:index.entry>
    <opentopic-index:index.entry value="style">
        <opentopic-index:formatted-value>style</opentopic-index:formatted-value>
        <opentopic-index:refID value="table of contents:style:">
            <oxy:index-link xmlns:oxy="http://www.oxygenxml.com/extensions/author"
                href="#d16e4120">
                [d16e4120] </oxy:index-link>
        </opentopic-index:refID>
  </opentopic-index:index.entry>
 </opentopic-index:index.entry>
</opentopic-index:index.group>
```

Each of the entries contain:

- The formatted value (`<opentopic-index:formatted-value>`).
- A link to the publication content (`<opentopic-index:refID>`/`<oxy: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.

```
<div class="- map/map map" >
   <div class="front-page/front-page">
       ...
   </div>
   <div class="toc/toc toc">
       ...
   </div>
   <div class="- topic/topic topic">
       <div class="- topic/title title">Request Support</title>
      ...
```

```
    </div>
    <div class=" index/groups groups">
      ...
    </div>
</map>
```

The index group content becomes:

```
<div class=" index/group group">
  <div class=" index/label label">T</div>

  <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 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="#d16e4176"> [d16e4176] </div>
        </div>
    </div>
    <div class=" index/entry entry">
        <div class=" index/formatted-value formatted-value">style</div>
        <div class=" index/refid refid">
            <div class=" index/link link"
               href="#d16e4120"> [d16e4120] </div>
        </div>
    </div>
  </div>
</div>
```

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

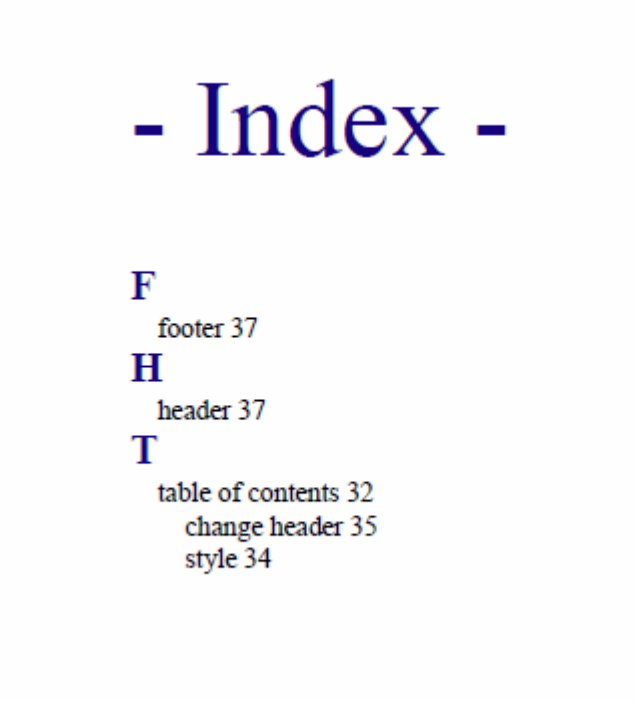
In your customization CSS *(on page 1469)*, add the following CSS rules:

```
*[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:

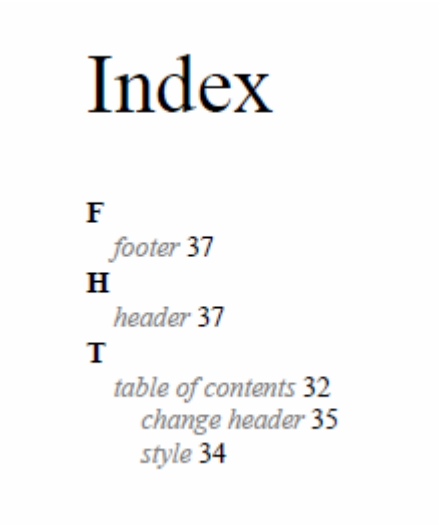


How to Style the Index Terms Labels

In your customization CSS *(on page 1469)*, add the following CSS rule:

```
*[class ~= 'index/groups'] *[class ~= 'index/formatted-value'] {
    font-style:oblique;
    color:gray;
}
```

The result is:



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 1469)*, add the following CSS rule:

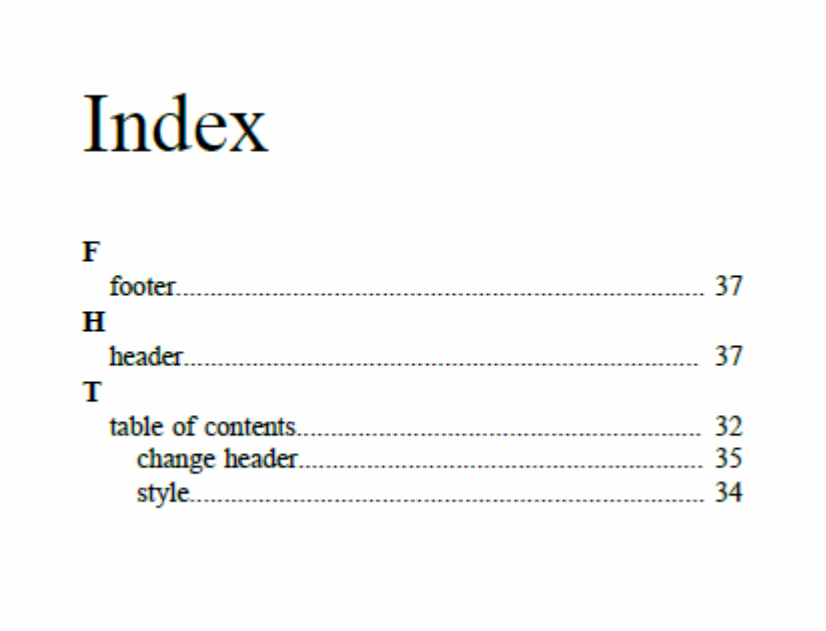
```
*[class~="index/formatted-value"],
*[class~="index/refid"] {
    display:inline;
}



/* Hide the sequences of links that actually do not contain links. */
*[class~="index/group"] *[class ~= "index/entry"] > *[class~="index/refid"]{
    display:none;
}
*[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 35
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:

```
*[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:

```
*[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 1469)*:

```
@page index {

        @bottom-center      {          content: counter(page, decimal) }

}
```

How to Display Index Terms and Page Numbers inline

By default, the page numbers that reference each index term are displayed beneath the term. This behavior can be overridden to render page numbers inline with their associated terms. The goal here is to render the following elements on the same line:

index/formatted-value

This element contains the index term text.

index/refid

This element contains the list of links for the given term (as page numbers).

To accomplish this, simply add the following CSS rule in your customization CSS *(on page 1469)*:

```
*[class ~= "index/formatted-value"],
*[class ~= "index/refid"] {
  display: inline;
}
```

How to Impose a Table-like Index Layout

For a more complex inline layout (requiring greater alignment constraint), using `inline-block` instead of `inline` provides a way to emulate a table-like appearance.

To implement this table layout, include the following CSS rule in your customization CSS *(on page 1469)*:

Notes:

- The 45% fixed width is used (slightly under 50% to keep content on the same line despite possible margins/paddings).
- Hierarchical layout is achieved by applying progressive padding to the formatted value text.

```
*[class ~= "index/formatted-value"],
*[class ~= "index/refid"] {
  display: inline-block;
  width: 45%;
}

/* Hide the sequences of links that actually do not contain links. */
*[class ~= "index/groups"] *[class ~= "index/entry"] > *[class ~= "index/refid"] {
  display: none;
}
*[class ~= "index/groups"] *[class ~= "index/entry"] > *[class ~= "index/refid"]:has(*[class ~= "index/link"]) {
  display:inline-block;
}

/* 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;
}
*[class ~= "index/groups"]
*[class ~= "index/entry"]
*[class ~= "index/entry"]
*[class ~= "index/formatted-value"] {
  padding-left: 0.4em;
}
*[class ~= "index/groups"]
*[class ~= "index/entry"]
*[class ~= "index/entry"]
*[class ~= "index/entry"]
*[class ~= "index/formatted-value"] {
  padding-left: 0.6em;
}

/* Some styling */
*[class ~= "index/formatted-value"],
*[class ~= "index/refid"] {
  padding: 0.2em;
  background-color: #EEEEEE;
}
```

You will obtain this result:

Index

C

Close programs	8
Computer cover	7,8
configure	10, 10,10

D

database	8, 8, 8, 9, 10, 11,12

H

hard drive	7, 7, 7, 8, 10, 11,12

I

install	7, 8,9

M

maintain	11,11
hdd	
drive	11

To avoid bleeding of the index term label, you may need to mark it as being hyphenated:

```
*[class~="index/formatted-value"] {
    hyphens:auto;
}
```

To activate hyphenation, see: How to Enable Hyphenation for Entire Map *(on page 1597)*.

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:

```
<part>
  <chapter/>
    <topicref/>
  <chapter/>
</part>
```

```
<appendices>  <!-- Appendices behaves like a Part -->
  <appendix/>  <!-- Appendix behaves like a Chapter -->
  <appendix/>
</appendices>
```

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 {
  break-before: always;
}
```

Footnotes

Footnotes are pieces of information that have several purposes, including citations, additional information (copyright, background), outside sources, and more. They are divided as follows:

- **The footnote call** - The number that remains in the content, usually superscripted.
- **The footnote marker** - The number displayed at the bottom of the page (the value matches the footnote call).
- **The footnote text** - The value of the `<fn>` element, also displayed at the bottom of the page.

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, use some colors, and change the footnote marker, add the following rules to your customization CSS *(on page 1469)*:

```
*[class ~= "topic/fn"]:footnote-call {
  font-weight: bold;
  color:red;
}

*[class ~= "topic/fn"]:footnote-marker {
  content: counter(footnote) " / ";
  font-weight: bold;
  color:red;
}
```

To indent the footnote content displayed at the end of the page, add the following rules to your customization CSS *(on page 1469)*:

```
*[class ~= "topic/entry"] > *[class ~= "topic/fn"] {
  padding-left: 1in;
}

*[class ~= "topic/entry"] > *[class ~= "topic/fn"]:footnote-marker {
  margin-left: 1in;
}
```

Related Information:

Oxygen PDF Chemistry: Footnotes

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.

```
@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

It is possible to reset the footnote counter. For example, if you want to reset the counter at the beginning of each chapter, add one of the following rules to your customization CSS *(on page 1469)*:

```
@page chapter {
  counter-reset: footnote 1;
}
```

```
*[class ~= "bookmap/chapter"],
*[class ~= "topic/topic"][is-chapter] {
  counter-reset: footnote 1;
}
```

In a deep numbering context, you need to use the following rule instead:

```
*[class ~= "map/map"][numbering ^= 'deep'] *[class ~= "topic/topic"][is-chapter]:not([is-part]) {
  counter-reset: section1 0 footnote 1;
}
```

or can mark any element with an `@outputclass` value, match that value, and reset the counter at any point in your counter:

```
<p outputclass="reset-footnotes"/>
```

```
*[outputclass ~= "reset-footnotes"] {
  counter-reset: footnote 1;
}
```

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 1464)*.
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 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"
xmlns:opentopic-func="http://www.idiominc.com/opentopic/exsl/function"
exclude-result-prefixes="xs opentopic-func"
version="3.0">

<!--
  Match only top level tables (i.e tables that are not nested in other tables),
  that contains some 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')]">
      <!--
        Try to preserve the footnote ID, if available, so that the xrefs will have a target.
      -->
      <li class="- topic/li " id="{if(@id) then @id else generate-id(.)}"
        outputclass="table-fn">
        <xsl:copy-of select="@callout"/>
        <xsl:apply-templates select="node()"/>
      </li>
    </xsl:for-each>
  </ol>
</xsl:template>

<!--
  The footnotes that have an ID must be ignored, they are accessible only
  through existing xrefs (already present in the merged.xml file).

  The above template already made a copy of these footnotes in the OL element
  so it is not a problem if markup is not generated for them in the cell.
-->
<xsl:template
  match="*[contains(@class, 'topic/entry')]//*[contains(@class, 'topic/fn')][@id]"/>

<!--
  The xrefs to footnotes with IDs inside table-cells. We need to recalculate
```

```
  their indexes if their referenced footnote is also in the table.
-->
<xsl:template match="*[contains(@class, 'topic/xref')][@type='fn']
  [ancestor::*[contains(@class, 'topic/entry')]]">
  <xsl:param name="top-level-table" tunnel="yes"/>
  <xsl:variable name="destination" select="opentopic-func:getDestinationId(@href)"/>
  <xsl:variable name="fn" select="
    $top-level-table//*[contains(@class, 'topic/fn')][@id = $destination]"/>
  <xsl:choose>
    <xsl:when test="$fn">
      <!-- There is a reference in the table, recalculate index. -->
      <xsl:variable name="fn-number" select="
        index-of($top-level-table//*[contains(@class, 'topic/fn')], $fn)"/>
      <xsl:copy>
        <xsl:apply-templates select="@*"/>
        <xsl:apply-templates select="$fn/@callout"/>
        <xsl:apply-templates select="node()
          except (text(), *[contains(@class, 'hi-d/sup')])"/>
        <sup class="+ topic/ph hi-d/sup ">
          <xsl:apply-templates select="child::*[contains(@class, 'hi-d/sup')]/@*"/>
          <xsl:value-of select="$fn-number"/>
        </sup>
      </xsl:copy>
    </xsl:when>
    <xsl:otherwise>
      <!-- There is no reference in the table, keep original index. -->
      <xsl:next-match/>
    </xsl:otherwise>
  </xsl:choose>
</xsl:template>

<!--
  The footnotes without ID inside table-cells. They are copied in the OL element, but have
  no xrefs pointing to them (because they have no ID), so xrefs are generated.
-->
<xsl:template
  match="*[contains(@class, 'topic/entry')]//*[contains(@class, 'topic/fn')][not(@id)]">
  <!-- Determine the footnote index in the document order. -->
  <xsl:param name="top-level-table" tunnel="yes"/>
  <xsl:variable name="fn-number" select="
    index-of($top-level-table//*[contains(@class, 'topic/fn')], .)"/>
  <xref type="fn" class="- topic/xref "
```

```
      href="#{generate-id(.)}" outputclass="table-fn-call">
      <xsl:copy-of select="@callout"/>
      <!-- Generate an extra <sup>, identical to what DITA-OT generates for other xrefs. -->
      <sup class="+ topic/ph hi-d/sup ">
        <xsl:value-of select="$fn-number"/>
      </sup>
    </xref>
  </xsl:template>

</xsl:stylesheet>
```

5. Open the *template descriptor file (on page 1459)* 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"/>
   </xslt>
```

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'] {
  line-height: none;
}
*[class ~= "topic/table"] *[class ~= "topic/xref"][type = 'fn'][callout] *[class ~= "hi-d/sup"] {
  content: oxy_xpath("ancestor::*[contains(@class, 'topic/xref')]/@callout");
}

/* Customize the list containing all the table footnotes. */
*[outputclass ~= 'table-fn-container'] {
  border-top: 1pt solid black;
  counter-reset: table-footnote;
}

/* Customize footnotes display, below the table. */
*[outputclass ~= 'table-fn'] {
  font-size: smaller;
```

```
  counter-increment: table-footnote;
}
*[outputclass ~= 'table-fn']::marker {
  font-size: smaller;
  content: "(" counter(table-footnote) ")";
}
*[outputclass ~= 'table-fn'][callout]::marker {
  content: "(" attr(callout) ")";
}

/* Customize xrefs pointing to footnotes, inside the table. */
*[class ~= "topic/table"] *[class ~= "topic/xref"][type = 'fn'] {
  color: unset;
  text-decoration: none;
}
*[class ~= "topic/table"] *[class ~= "topic/xref"][type = 'fn']:after {
  content: none;
}
*[class ~= "topic/table"] *[class ~= "topic/xref"][type = 'fn'] *[class ~= "hi-d/sup"]:before {
  content: "(";
}
*[class ~= "topic/table"] *[class ~= "topic/xref"][type = 'fn'] *[class ~= "hi-d/sup"]:after {
  content: ")";
}
```

7. Open the *template descriptor file (on page 1459)* 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>
```

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 **Oxygen PDF Chemistry** processor to perform hyphenation.

Hyphenation Dictionaries

The Oxygen XML Developer provides built-in hyphenation patterns for the following languages:

Code	Language
da	Danish
de	German
de_CH	German (Switzerland)
en	English
es	Spanish
fr	French
it	Italian
nb	Norwegian Bokmål
nl	Dutch
pt	Portuguese
ru	Russian
sv	Swedish
th	Thai

The built-in hyphenation pattern license terms are listed in the LICENSE.txt files in the `OXYGEN_INSTALL_DIR/lib` folder. Most of them comply with the *LaTex* distribution policy.

How to Install a Hyphenation Dictionary

Oxygen XML Developer uses the *TeX* hyphenation dictionaries converted to XML by the *OFFO* project: 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 1596)*).

The `.hyp` files are the compiled dictionaries that the Oxygen XML Developer actually uses.

One simple way to add more dictionaries:

1. Download and extract the `offo-hyphenation-compiled.zip` (or the `offo-hyphenation.zip`) file.
2. Copy and extract the `fop-hyph.jar` file (containing the dictionaries bundle) into a directory in your project.
3. If you just need a single dictionary, copy the `.hyp` or `.xml` file extracted from the above jar.
4. Reference the directory using the `args.hyph.dir` parameter.

How to Alter a Hyphenation Dictionary

You can copy the dictionaries you need to change in another directory, then use the `args.hyph.dir` parameter to reference it inside your transformation.

Each file is named with the language code and has the following structure:

```
<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 1469)*, add:

```
:root {
    hyphens: auto;
}
```

3. To except certain elements from being hyphenated, use `hyphens:none`. The following example excludes the `<keyword>` elements from being hyphenated:

```
*[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:

```
*[class~="pr-d/codeblock"] {
    hyphens: auto;
}
```

If you want to disable hyphenation inside tables:

```
*[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:

```
<codeblock outputclass="hyphens">
  ...
</codeblock>
```

If you want to disable hyphenation inside tables:

```
<table outputclass="no-hyphens" ...>
  ...
</table>
```

Note:

The default built-in CSS enables hyphenation for tables:

```
*[class ~= "topic/table"] {
    hyphens: auto;
}
```

Related information

How to Enable Line Wrap in Code Phrases *(on page 1653)*

How to Disable Hyphenation for a Word

How to Define Hyphenation for a Specific Word

1. Create a new hyphenation dictionary *(on page 1596)*.
2. Add the word under the `<exceptions>` section using hard hyphen symbols between its segments.
3. To make sure the words from your document match against the ones from the "exceptions", make sure that you add capitalized/lower case variants as well.

Related information

How to Disable Hyphenation for a Word

How to Alter a Hyphenation Dictionary *(on page 1596)*

How to Enable/Disable Hyphenation for an Element *(on page 1597)*

How to Force or Avoid Line Breaks at Hyphens

It is possible to force or avoid line breaks inside words with hyphens (`U+2010`). This can be useful, for example, inside tables that have product references if you want the display to remain on a single line (or to split it on multiple lines). To achieve this, you can use the `-oxy-break-line-at-hyphens` property:

The accepted values are:

auto

Words are hyphenated automatically according to an algorithm that is driven by a hyphenation dictionary. This can lead to line breaks at hyphens.

avoid

Words are still hyphenated automatically except no line break will occur on hyphens.

always

Words are still hyphenated automatically except line breaks will be forced on hyphens.

Example:

Suppose you have a products table like this:

```
<table>
  <row>
    <cell>Product-1233-55-88</cell>
    <cell>120</cell>
  <row>
  <row>
    <cell>Product-1244-66-99</cell>
    <cell>112</cell>
  <row>
</table>
```

and the following rule in a CSS stylesheet:

```
table {
  -oxy-break-line-at-hyphens: avoid;
}
```

In the output, the list of product references will be displayed in a single line. On the contrary, setting the property value to `always`, will force a break after each hyphen.

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 1469)*. For instance, you can use:

```
body { font-family: Arial }


code { font-family: "Courier New" }


*[class ~= "pr-d/codeph"],
*[class ~= "pr-d/codeblock"],
*[class ~= "sw-d/filepath"] {
  font-family: "Courier New";
}


@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 }


  @bottom-left {font-family: Arial }
  @bottom-right {font-family: Arial }
  @bottom-center {font-family: Arial }
  @bottom-left-corner {font-family: Arial }
  @bottom-right-corner {font-family: Arial }
}
```

2. Create a new transformation scenario (based on the **DITA Map PDF - based on HTML5 & CSS** or the **DITA 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 scenario.

Related Information:

Oxygen PDF Chemistry: Fully Accessible PDF (PDF/UA1)

Archiving

Your PDF files may need to be archived for security or legal reasons. In this case, the generated file must be compliant to the PDF/A ISO standard.

Related Information:

Oxygen PDF Chemistry: Archiving

How to Allow Document Archiving

To make your documents archive-able (PDF/A compliant), do the following:

1. The archiving 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 1469)*. For instance, you can use:

```
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 }


    @bottom-left {font-family: Arial }
    @bottom-right {font-family: Arial }
    @bottom-center {font-family: Arial }
    @bottom-left-corner {font-family: Arial }
    @bottom-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, select a value for the `pdf.archiving.mode` parameter from the list.
4. Run the transformation scenario.

Fonts

Fonts are an important part of the publication. Your font selection should take into consideration both design and the targeted ranges of characters.



Notes:

- Before using a font, make sure you have the permissions to use it and make sure you comply with all the license terms.
- When installing a font on Windows, make sure you select the **Install for all users** option.

To use them in the customization CSS *(on page 1469)*:

- 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 Set Fonts in Titles and Content

Suppose that in your customization CSS *(on page 1469)*, you have defined your font (for example, *Roboto*) using a Google web font:

```
https://fonts.googleapis.com/css2?family=Roboto:ital,wght@0,400;0,700;1,400;1,700&display=swap
```

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 1469)*, add an `!important` rule that associates a font to all the elements from the document:

```
* {
  font-family: "Roboto", sans-serif !important;
}
```

Note:
If you want to use the `:root` selector instead of the `*` sector, 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.

Next, if you want to use another font for the document headings, your customization CSS *(on page 1469)* should contain the following rule:

```
*[class ~= "front-page/front-page-title"],
*[class ~= "topic/title"] {
  font-family: "Roboto", sans-serif !important;
  font-weight: bold;
}
```

Then, 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 1471)*.

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:

```
*[class~="pr-d/codeph"],
*[class~="topic/pre"] {
```

```
  font-family: monospace !important;
}
```

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 1486)*.

Related information
How to Change the Font of the Headers and Footers *(on page 1486)*

How to Use Fonts for Symbols

For some specific symbols (e.g. arrows), you must use a font or a sequence of fonts that cover the needed character ranges. 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 you are using the *Roboto* font, the character is not found, the # symbol is used, and a warning is thrown (see also Glyph Not Available in Font *(on page 1707)*):

```
[CH] Glyph "?" (0x2192, arrowright) not available in font "Roboto-Bold".
```

The PDF output will then look like this:

```
  #
    This is the right arrow.
```

To solve this issue, you can use *Segoe UI* as a fallback font. In your customization CSS *(on page 1469)*, add:

```
*[class ~= "topic/dl"] {
  font-family: "Roboto", "Segoe UI";
}
```

To change the font for the entire publication, not just an element, you can use:

```
:root {font-family:  "Roboto", "Segoe UI" !important; }

@page {
    @top-left {font-family:  "Roboto", "Segoe UI" !important; }
    @top-right {font-family:  "Roboto", "Segoe UI" !important; }
    @top-center {font-family:  "Roboto", "Segoe UI" !important; }
    @top-left-corner {font-family:  "Roboto", "Segoe UI" !important; }
    @top-right-corner {font-family:  "Roboto", "Segoe UI" !important; }
```

```
    @bottom-left {font-family:  "Roboto", "Segoe UI" !important; }
    @bottom-right {font-family:  "Roboto", "Segoe UI" !important; }
    @bottom-center {font-family:  "Roboto", "Segoe UI" !important; }
    @bottom-left-corner {font-family:  "Roboto", "Segoe UI" !important; }
    @bottom-right-corner {font-family:  "Roboto", "Segoe UI" !important; }
}
```

Tip:
It is possible to use a generic family name as fallback (like `serif`, `sans-serif` or `monospace`) to call upon the processor's default fallback fonts system.

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:

```
font-family: Calibri, SimSun, "Malgun Gothic", "Microsoft JhengHei";
```

To apply this font sequence, see: How to Set Fonts in Titles and Content *(on page 1602)*.

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, **Oxygen 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 1469)*. The following example uses the Noto Sans Tamil font-family:

```
/* Font Declaration */
@font-face {
    font-family: "Noto Sans Tamil";
    font-style: normal;
    font-weight: 400;
    src: url(../fonts/ttf/notosanstamil/NotoSansTamil-Regular.ttf);
}

@font-face {
    font-family: "Noto Sans Tamil";
    font-style: normal;
    font-weight: 700;
    src: url(../fonts/ttf/notosanstamil/NotoSansTamil-Bold.ttf);
}

/* Font Usage */
* {
    font-family: sans-serif, "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:

```
@font-face {
  font-family: MusGlyphs;
  font-style: normal;
  font-weight: 400;
  src: url(../fonts/otf/musglyphs/MusGlyphs.otf);
}
```

```
@font-face {
  font-family: MusGlyphs-Text;
  font-style: normal;
  font-weight: 400;
  src: url(../fonts/otf/musglyphs/MusGlyphs-Text.otf);
}
@font-face {
  font-family: MusGlyphs-Text;
  font-style: normal;
  font-weight: bold;
  src: url(../fonts/otf/musglyphs/MusGlyphs-TextBold.otf);
}

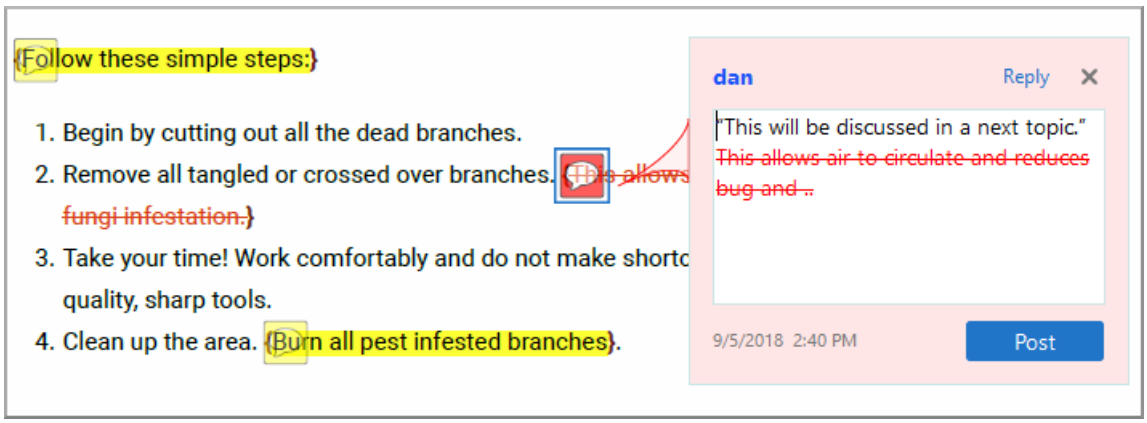
/*
 * All the elements are displayed with the MusGlyphs-Text.
 * If a text is prefixed with @, music will be displayed.
 */
body {
  font-family: MusGlyphs-Text, serif;
}

/* Elements with @outputclass="music" contain only music. */
*[outputclass ~= "music"] {
  font-family: MusGlyphs, serif;
}
```

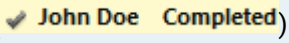
Comments, Highlights, and Tracked Changes

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 414. 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 (

).

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 1471)* 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 1452)*.

Related Information:
Transformation Parameters *(on page 1444)*
Debugging the CSS *(on page 1471)*

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`.

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 1471)*.

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.

```
<span class="oxy-range-start" id="sc_1" hr_id="1"/>

   <span class="oxy-insert" href="#sc_1" hr_id="1">
       <span class="oxy-author">dan</span>
       <span class="oxy-content">insert</span>
       <span class="oxy-date">2018/03/15</span>
       <span class="oxy-hour">09:38:29</span>
       <span class="oxy-tz">+02:00</span>
   </span>
   <span class="oxy-insert-hl">This is an insert!!</span>

<span class="oxy-range-end" hr_id="1"/>
```

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.

```
<span class="oxy-range-start" id="sc_1" hr_id="1"/>

    <span class="oxy-comment" href="#sc_1" hr_id="1">
          <span class="oxy-author">dan</span>
          <span class="oxy-comment-text">This is a comment.</span>
          <span class="oxy-date">2018/03/15</span>
          <span class="oxy-hour">09:56:59</span>
          <span class="oxy-tz">+02:00</span>
    </span>

    <span class="oxy-comment-hl">The commented text.</span>

<span class="oxy-range-end" hr_id="1"/>
```



Note:

Comments that are marked as done have a `flag="done"` attribute:

```
<span class="oxy-comment" href="#sc_6" hr_id="6" flag="done">
```

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`.

```
<element>


  <span class="oxy-range-start" id="sc_3" hr_id="3"/>
  <span class="oxy-range-end" hr_id="3"/>


  <span class="oxy-attributes" href="#sc_3" hr_id="3">


     <span class="oxy-attribute-change" type="inserted" name="platform">
          <span class="oxy-author">dan</span>
          <span class="oxy-current-value">windows</span>
          <span class="oxy-date">2018/03/15</span>
          <span class="oxy-hour">10:05:04</span>
          <span class="oxy-tz">+02:00</span>
     </span>
    ....
     <span class="oxy-attribute-change" type="removed" name="audience">
        ....
     </span>
  </span>
...
</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 an element with the class `oxy-delete-hl`.

```
   <span class="oxy-range-start" id="sc_2" hr_id="2"/>
   <span class="oxy-delete-hl"> This is a deleted text. </span>
   <span class="oxy-range-end" hr_id="2"/>
```

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:

```
   <span class="oxy-delete" href="#sc_2" hr_id="2">
      <span class="oxy-author">dan</span>
      <span class="oxy-content"><image href="../img/ex.gif"></span>
      <span class="oxy-date">2018/03/14</span>
      <span class="oxy-hour">11:38:06</span>
```

```
    <span class="oxy-tz">+02:00</span>
 </span>
```

Colored Highlights

To show some text as highlighted with a background color:

```
<span class="oxy-color-hl" color="rgba(140,255,140,50)">Some colored text.</span>
```

How to Style Tracked Changes or Comments

Here are some examples showing how to customize tracked changes and highlighted text:

- If you want to change the highlighted text color from the document content, use the `@class="oxy-comment-hl"` attribute (or `@class="oxy-delete-hl"`, `@class="oxy-insert-hl"`):

```
.oxy-comment-hl {
  color:magenta;
}
```

- If you want to change the range labels indicating the start or the end of a change (by default, formatted like this: "[n]...[/n]" where n is the change number), you can use the following selectors:

```
.oxy-range-start:before {
  content:'[START]';
  color:red;
}
.oxy-range-end:before {
  content:'[END]';
  color:red;
}
```

- If you want to only show the changes and comments highlights

```
.oxy-range-start,
.oxy-range-end {
  display: none;
}
.oxy-insert,
.oxy-delete {
  display: none;
}
```



Note:
No comments will be displayed in the PDF Viewer Comments view after this modification.

How to Style Tracked Changes Shown as Footnotes

Important:
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.

Here are some examples showing how to customize footnotes:

- If you want to change the background color and the border of the comment footnote, add the following snippet in your customization CSS *(on page 1469)*:

```
.oxy-comment {
  background-color:inherit;
  border: 2pt solid yellow;
}
```

  Similarly, you can style the other footnotes for `@class="oxy-attributes"`, `@class="oxy-delete"`, and `@class="oxy-insert"`.
- If you want to hide some footnotes (for example, the footnotes associated with the insertions, deletions, or attribute changes when your document contains a lot of tracked changes), add something like this in your customization CSS *(on page 1469)* (the following example results in the deletions and insertions being hidden, but the comments remain visible):

```
.oxy-attributes,
.oxy-delete,
.oxy-insert{
  float:none;
  display:none;
}
```

How to Show Only Change Bars on 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 1469)*:

```
.oxy-attributes,
.oxy-comment,
.oxy-delete,
.oxy-insert {
  float: initial;
  display: none;
}
```

3. Remove the change range markers (the **{** and **}** symbols):

```
.oxy-range-start:before,
.oxy-range-end:before {
  content:none;
}
```

4. Remove the styling for the insertions and deletions:

```
.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:

```
.oxy-range-start[is-changebar]:before(100) {
  -oxy-changebar-color: red;
  -oxy-changebar-width: 3pt;
}
```

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 1469)*:

```
@page {
    background-image: url("draft.svg");
    background-position:center;
    background-repeat:no-repeat;
    background-size: 100% 100%;
}
```

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 1469)*.

Related Information:
Images and Figures *(on page 1628)*

How to Add a Draft Watermark in the Foreground

If you want the watermark to be displayed above the text (in the foreground), instead of using the standard `background-image` property, you can use the `-oxy-foreground-image` property:

```
@page {
    -oxy-foreground-image: url("draft.svg");
}
```

You can set a more specific selector if you just need to display the foreground in a subset group of pages (for example, `chapter`). In this case, the above selector will not change it since it is more specific.

Note:
The usage of SVG images is preferred because other image types suffer from *pixelation* and because foreground images are stretched to the full page size.

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).

```
<bookmeta>
   <author>John</author>
```

```
   <critdates>
      <created date="1/1/2015"/>
      <revised modified="3/4/2016"/>
      <revised modified="3/5/2016"/>
   </critdates>
   <bookchangehistory>
       <approved/>
   </bookchangehistory>
...
```

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):

```
@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:

```
: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).

Related information
Metadata *(on page 1529)*
How to Debug XPath Expressions *(on page 1476)*

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>
```

Related Information:

Change Bars

DITAVAL Elements

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"
         backcolor="blue"
         color="yellow" >

      <startflag imageref="startflag.jpg">
        <alt-text>This is the start of my product info</alt-text>
      </startflag>

      <endflag imageref="endflag.jpg">
        <alt-text>This is the end of my product info</alt-text>
      </endflag>
   </prop>
</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 1469)* and then add CSS rules. To create the CSS rules, you can use the development tools described in Debugging the CSS *(on page 1471)*.

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:

```
<h1 class="title topictitle1" 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>
```

And the same example from the PDF transformation (note the additional emphasized class values):

```
<h1 class="- topic/title title topictitle1" id="ariaid-title2">Care andPreparation</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>
```

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:

```
.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 in PDF can be classified into two categories:

- Table of contents titles, identified by the `map/topicref` and `topic/navtitle` class attributes.
- Content titles, that can be styled by matching the `topic/title` class attribute.

How to Control Titles Layout

By default, titles are rendered on a single line (with both the chapter/section number and 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 each line of the title to start at the same location (and indented), you need to set the value of the `args.css.param.title.layout` transformation parameter to **table**. This means that the chapter/section number is placed in one cell and title text is placed in another cell (resulting and indented text):

```
4.5.5 This is a long title

      text that wraps.
```

How to Change Chapters Title Prefix

Changing Prefixes in Shallow Numbering

In shallow numbering (default), to replace the "Chapter N." prefix, use the following rules in your customization CSS *(on page 1469)*:

```
*[class ~= "map/topicref"][is-chapter]:not([is-part]) > *[class ~= "map/topicmeta"] > *[class ~= "topic/navtitle"]:before{

  content: "Module " counter(toc-chapter, decimal-leading-zero) " - ";

}

*[class ~= "topic/topic"][is-chapter]:not([is-part]) > *[class ~= "topic/title"]:before {

  content: "Module " counter(chapter, decimal-leading-zero) "\A";

}
```

Changing Prefixes in Deep Numbering

In deep numbering, to replace the "N." prefix, use the following rules in your customization CSS *(on page 1469)*:

```
*[class ~= "map/map"][numbering ^= 'deep'] *[class ~= "topic/topic"][is-chapter]:not([is-part]) > *[class

 ~= "topic/title"]:before,

*[class ~= "map/map"][numbering ^= 'deep'] *[class ~= "topic/topic"][is-chapter]:not([is-part]) *[class ~= "topic/topic"] >

 *[class ~= "topic/title"]:before {

  content: counters(chapter-and-sections, ".") "\A";

}
```

How to Remove Parts and Chapter Title Prefixes

Removing Prefixes in Shallow Numbering

In shallow numbering (default), to hide the "Part N" and "Chapter NN" prefixes, use the following rules in your customization CSS *(on page 1469)*:

```
*[class ~= "map/topicref"] > *[class ~= "map/topicmeta"] > *[class ~= "topic/navtitle"]:before {

  display: none !important;

}

*[class ~= "topic/topic"] > *[class ~= "topic/title"]:before {
```

```
  display: none !important;
}
```

You can also choose to remove only the "Part N" prefix:

```
*[class ~= "map/topicref"][is-part] > *[class ~= "map/topicmeta"] > *[class ~= "topic/navtitle"]:before {
  display: none !important;
}
*[class ~= "topic/topic"][is-part] > *[class ~= "topic/title"]:before {
  display: none !important;
}
```

Or to remove only the "Chapter NN" prefix:

```
*[class ~= "map/topicref"][is-chapter]:not([is-part]) > *[class ~= "map/topicmeta"] > *[class ~= "topic/navtitle"]:before {
  display: none !important;
}
*[class ~= "topic/topic"][is-chapter]:not([is-part]) > *[class ~= "topic/title"]:before {
  display: none !important;
}
```

Removing Prefixes in Deep Numbering

In deep numbering, to hide the "Part N" and "Chapter NN" prefixes, use the following rules in your customization CSS *(on page 1469)*:

```
*[class ~= "map/map"][numbering ^= 'deep'] *[class ~= "map/topicref"] > *[class ~= "map/topicmeta"]:before {
  display: none !important;
}
*[class ~= "topic/topic"] > *[class ~= "topic/title"]:before {
  display: none !important;
}
```

How to Display Chapters Title on a Separate Page

You may want to display the chapters title on a separate page (for example, with a background). To do this, a new page should be defined in your customization CSS *(on page 1469)*:

```
@page chapter-title-page {
  background-image: url("resources/title-bg.png");
  background-repeat: no-repeat;
  background-size: 100% 100%;
  background-position: center;
}
```

After that, you need to replace the default "chapter" page definition with the one created before:

```
*[class ~= "topic/topic"][is-chapter] {

  page: chapter-title-page;

}
```

Then, you need to set it back on the content following the title:

```
*[class ~= "topic/topic"][is-chapter] > *[class ~= "topic/title"] ~ *:not([class ~= "topic/title"]) {

  page: chapter;

}
```

Finally, you can customize the title color, size, and more:

```
*[class ~= "topic/topic"][is-chapter]:not([is-part]) > *[class ~= "topic/title"] {

  color: white;

  font-size: 32pt;

}
```

Related information

Default Chapter Page Definition *(on page 1479)*

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:

```
*[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.

How to Assign Different Numbering to Figures and Equations

By default, both `<fig>` and `<equation-figure>` elements are numbered as figures. To use a different type of numbering for equations, use the following rules in your customization CSS *(on page 1469)*:

```
*[class ~= "equation-d/equation-figure"]:has(*[class ~= "topic/title"]) {
  counter-increment: equationcount;
}

*[class ~= "topic/fig"] .fig--title-label {
  content: "Figure";
}
*[class ~= "equation-d/equation-figure"] .fig--title-label {
  content: "Equation";
}

/* Shallow and Deep Numbering. */
*[class ~= "topic/fig"] .fig--title-label:after {
  content: " " counter(figcount) ". ";
}
*[class ~= "equation-d/equation-figure"] .fig--title-label:after {
  content: " " counter(equationcount) ". ";
}

/* Deep Chapter Scope and Deep Chapter Scope No Page Reset Numbering. */
*[class ~= "map/map"][numbering ^= 'deep-chapter-scope'] *[class ~= "topic/fig"] > figcaption > .fig--title-label:after {
  content: " " counter(chapter) "-" counter(figcount) " ";
}
*[class ~= "map/map"][numbering ^= 'deep-chapter-scope'] *[class ~= "equation-d/equation-figure"] > figcaption
 > .fig--title-label:after {
  content: " " counter(chapter) "-" counter(equationcount) " ";
}
```

If there are links to these figures/equations, the following rules must also be added:

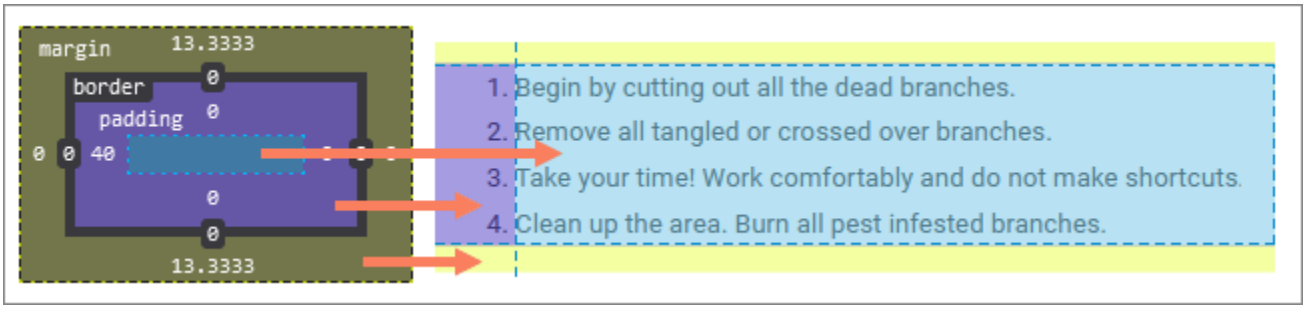
```
/* Shallow and Deep Numbering. */
*[class ~= "topic/xref"][href][type = "equation-figure"] {
  content: "Equation " target-counter(attr(href), equationcount);
}
*[class ~= "topic/xref"][href][type = "fig"] {
  content: "Figure " target-counter(attr(href), figcount);
}

/* Deep Chapter Scope and Deep Chapter Scope No Page Reset Numbering. */
```

```
*[class ~= "map/map"][numbering ^= 'deep-chapter-scope'] *[class ~= "topic/xref"][href][type = "equation-figure"] {
  content: "Equation " target-counter(attr(href), chapter) "-" target-counter(attr(href), equationcount);
}
*[class ~= "map/map"][numbering ^= 'deep-chapter-scope'] *[class ~= "topic/xref"][href][type = "fig"] {
  content: "Figure " target-counter(attr(href), chapter) "-" target-counter(attr(href), figcount);
}
```

Lists

This is the default layout for lists (both ordered and unordered 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 Style Lists

Some common use-cases for styling lists require you to change each list level separately. For example, for an ordered list:

```
*[class ~= "topic/ol"] > *[class ~= "topic/li"] /* First Level */ {
  font-size: 15pt;
}
*[class ~= "topic/ol"] *[class ~= "topic/ol"] > *[class ~= "topic/li"] /* Second Level */ {
  font-size: 13pt;
}
*[class ~= "topic/ol"] *[class ~= "topic/ol"] *[class ~= "topic/ol"] > *[class ~= "topic/li"] /* Third Level */ {
  font-size: 11pt;
```

```
}
/* Etc. */
```

Similarly, for an unordered list:

```
*[class ~= "topic/ul"] > *[class ~= "topic/li"]::marker /* First Level */ {
  color: red;
  content: "\2022";
}
*[class ~= "topic/ul"] *[class ~= "topic/ul"] > *[class ~= "topic/li"]::marker /* Second Level */ {
  color: orange;
  content: "\2022";
}
*[class ~= "topic/ul"] *[class ~= "topic/ul"] *[class ~= "topic/ul"] > *[class ~= "topic/li"]::marker /* Third Level */ {
  color: green;
  content: "\2022";
}
/* Etc. */
```

Note:
It is possible to mix lists type simply by mixing `*[class ~= "topic/ol"]` and `*[class ~= "topic/ul"]` in the CSS selector.

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:

```
ol {
    display:block;
    margin-top: 1.33em;
    margin-bottom: 1.33em;
    list-style-type:decimal;
    padding-left: 40px;
}
ul {
    display:block;
    margin-top: 1.33em;
    margin-bottom: 1.33em;
    list-style-type:disc;
    padding-left: 40px;
}
```

To align the lists, the following rules are sufficient in the customization CSS *(on page 1469)*:

```
*[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 Change List Item Markers Font

You can change the fonts for list items markers to make them easier to read or compliant with your company fonts. By default, **Oxygen PDF Chemistry** already provides a list of fallback fonts (to cover the majority of symbols).

To customize the fonts, add the following rule to your customization CSS *(on page 1469)*:

```
*[class ~= "topic/ul"] > *[class ~= "topic/li"]::marker {
  font-family: "Segoe UI", sans-serif;
}
```

Related information
Using Web Fonts
Using Local Font Files
Customizing :marker Elements

How to Continue List Numbering

It is possible to continue the numbering of an ordered list even when the content is split in multiple `<ol>` elements.

Using the 'start-x' outputclass

One solution consists of adding the `@outputclass="start-x"` attribute (where X is the number you want the list to start with) to the `<ol>` element. This is especially useful when lists do not have the same parent:

```
<table frame="all">
  <title>Table with nested order lists</title>
  <tgroup cols="1">
    <tbody>
      <row>
        <entry>
          <ol>
            <li>First Item</li>
            <li>Second Item</li>
```

```
          </ol>
        </entry>
      </row>
      <row>
        <entry>
          <ol outputclass="start-3">
            <li>Third Item</li>
            <li>Fourth Item</li>
          </ol>
        </entry>
      </row>
    </tbody>
  </tgroup>
</table>
```

Using the 'continue' outputclass and CSS

Another solution is to add the `@outputclass="continue"` attribute to the list where numbering should continue:

```
<ol>
  <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:

```
*[class ~= "topic/ol"] {
    counter-reset: item-count;
}

*[class ~= "topic/ol"][outputclass ~= "continue"] {
    counter-reset: none;
}

/* Add counter marker for each list level */
*[class ~= "topic/ol"] > *[class ~= "topic/li"]::marker {
    counter-increment: item-count;
    content: counter(item-count, decimal) ". ";
}
*[class ~= "topic/ol"][type=a] > *[class ~= "topic/li"]::marker{
    content: counter(item-count, lower-alpha) ". ";
```

```
}
*[class ~= "topic/ol"][type=A] > *[class ~= "topic/li"]::marker{
    content: counter(item-count, upper-alpha) ". ";
}
*[class ~= "topic/ol"][type=i] > *[class ~= "topic/li"]::marker{
    content: counter(item-count, lower-roman) ". ";
}
*[class ~= "topic/ol"][type=I] > *[class ~= "topic/li"]::marker{
    content: counter(item-count, upper-roman) ". ";
}
```

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`.

```
*[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`:

Important:
This example will work only for lists up to 28 items. You will have to extend it for longer lists!

```
*[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" }
*[class ~= "topic/ol"][outputclass ~= "cyrillic"] > *[class ~= "topic/li"]:nth-of-type(16):marker{ content:"р" }
*[class ~= "topic/ol"][outputclass ~= "cyrillic"] > *[class ~= "topic/li"]:nth-of-type(17):marker{ content:"с" }
*[class ~= "topic/ol"][outputclass ~= "cyrillic"] > *[class ~= "topic/li"]:nth-of-type(18):marker{ content:"т" }
*[class ~= "topic/ol"][outputclass ~= "cyrillic"] > *[class ~= "topic/li"]:nth-of-type(19):marker{ content:"у" }
*[class ~= "topic/ol"][outputclass ~= "cyrillic"] > *[class ~= "topic/li"]:nth-of-type(20):marker{ content:"ф" }
*[class ~= "topic/ol"][outputclass ~= "cyrillic"] > *[class ~= "topic/li"]:nth-of-type(21):marker{ content:"х" }
*[class ~= "topic/ol"][outputclass ~= "cyrillic"] > *[class ~= "topic/li"]:nth-of-type(22):marker{ content:"ц" }
*[class ~= "topic/ol"][outputclass ~= "cyrillic"] > *[class ~= "topic/li"]:nth-of-type(23):marker{ content:"ч" }
*[class ~= "topic/ol"][outputclass ~= "cyrillic"] > *[class ~= "topic/li"]:nth-of-type(24):marker{ content:"ш" }
*[class ~= "topic/ol"][outputclass ~= "cyrillic"] > *[class ~= "topic/li"]:nth-of-type(25):marker{ content:"щ" }
*[class ~= "topic/ol"][outputclass ~= "cyrillic"] > *[class ~= "topic/li"]:nth-of-type(26):marker{ content:"э" }
*[class ~= "topic/ol"][outputclass ~= "cyrillic"] > *[class ~= "topic/li"]:nth-of-type(27):marker{ content:"ю" }
*[class ~= "topic/ol"][outputclass ~= "cyrillic"] > *[class ~= "topic/li"]:nth-of-type(28):marker{ content:"я" }
```

Related Information:

Oxygen PDF Chemistry User Guide: Lists

Links

Links allow the users to navigate through the documentation.

How to Change '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, you can simply disable the `args.css.param.show-onpage-lbl` parameter.

This can also be done by adding the following in your customization CSS *(on page 1469)*:

```
*[class ~= "topic/xref"][href]:after,
*[class ~= "topic/link"][href]:after {
    content: none !important;
}
```

Note:

A variant is to remove the "on page" label only and keep the page number:

```
*[class ~= "topic/xref"][href]:after,
*[class ~= "topic/link"][href]:after {
    content: " (" target-counter(attr(href), page) ")" !important;
}
```

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:

```
*[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 1469)*, add this snippet:

```
*[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 1471)* like this:

```
<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>
         </link>
    </linkpool>
    ...
</related-links>
```

If you need to hide these descriptions, add the following code in your customization CSS *(on page 1469)*:

```
*[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:

```
Related information
  Target Topic
  Target Concept
  Target Task
```

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:

```
Related concepts
  Target Concept
Related tasks
  Target Task
Related information
  Target Topic
```

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.

Related information
The DITA Style Guide Best Practices for Authors: Image file formats

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 1469)*, 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 1635)* 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 1469)*:

```
*[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 1635)*.

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 1444)* to another value (from 72, 120, 300 and 600).

How to Place Big Images on Rotated Pages

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 1469)*, add:

```
*[class~="topic/image"][outputclass='land'] {
  page: landscape-page;
}
```

Setting the `@outputclass` = 'land' attribute on the `<image>` element forces the image to be displayed on a new landscape page.

If you want to rotate the entire topic that contains the image, use:

```
*[class~="topic/topic"]:has(*[class~="topic/body"] > *[class~="topic/image"][outputclass="land"]),
*[class~="topic/topic"]:has(*[class~="topic/body"] > * > *[class~="topic/image"][outputclass="land"]),
*[class~="topic/topic"]:has(*[class~="topic/body"] > * > * > *[class~="topic/image"][outputclass="land"]) {
  page: landscape-page;
}
```

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 inside a `<div>` element like this:

```
...
 <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 1469)*, add:

```
*[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:

```
*[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:

```
*[outputclass ~= "side-by-side"] > *[class ~= "topic/p"] {
  vertical-align:top;
}

*[outputclass ~= "side-by-side"] > *[class ~= "topic/fig"] {
  vertical-align:top;
}
```



Note:

If your figure does not have a title, you can also set `font-size:0pt` to remove the font ascent and descent around the image rectangle.

```
*[outputclass ~= "side-by-side"] > *[class ~= "topic/fig"] {
  vertical-align:top;
  font-size:0pt;
}
```

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:

```
*[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:

```
*[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:

```
*[class ~= "topic/note"]:before(2) {
    content: url('note.png') ;
    width:0.5in;
}

*[class ~= "topic/note"]:before(1) {
    content: "Some text";
}
```

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 1469)*, add:

```
*[class ~= "topic/fig"] {
  text-align:center;
}

/* 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 Surround Images with Borders

Suppose you want to surround all images contained inside figures that have the placement set to `break` by a border to emphasize them. To resize them, you need to use the width calculated by DITA-OT (by default, blocks use their parent max-width, which is the whole page in this case).

In your customization CSS *(on page 1469)*, add:

```
@namespace dita-ot url('http://dita-ot.sourceforge.net/ns/201007/dita-ot');


*[class ~= "topic/fig"] *[class ~= "topic/image"][placement='break'] {
  border: 1px solid black;
  padding: 1em;
}
*[class ~= "topic/image"][placement='break'][dita-ot|image-width] {
  width: attr(dita-ot|image-width px);
}
```

How to Remove the Figure 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: `fig--title-label`.

```
<figure class="- topic/fig fig ">
  <figcaption class="- topic/title title figcap">
    <span class="figtitleprefix fig--title-label">Figure
      <span class="fig--title-label-number">1</span>
      <span class="fig--title-label-punctuation">. </span>
    </span>
    <span class="fig--title">
      <span class="- topic/ph topic/title-wrapper ph title-wrapper">Lorem picsum</span>
    </span>
  </figcaption>
  ...
</figure>
```

To hide it, set its display to *none*:

```
.fig--title-label {
  display:none;
}
```

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>`.

```
*[class ~= "topic/topic"][is-chapter] {
  counter-reset: figcount;
}

.fig--title-label {
  display: none;
}

*[class ~= "topic/fig"] > .figcap:before {
  display: inline;
  content: "Figure " counter(chapter) "-" counter(figcount) " ";
}
```

Of course you will need to update the links to these figures to show the same number:

```
.fig--title-label-number,
.fig--title-label-punctuation {
  display: none;
}

.fig--title-label-number-placeholder {
  content: target-counter(attr(href), chapter) "-" target-counter(attr(href), figcount) "  ";
}
```

How to Fix Missing Images

If your images are not accessible, you may receive an error message in the transformation console like this:

```
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 1629)*, 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 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.

   ```
   <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):

```
<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>
      <xref
          href="parts/drive-shaft.dita#drive_shaft_topic/drive_shaft"
          format="dita">Drive Shaft</xref>
  </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.

Warning:
Areas coordinates are relative the image box and are not affected by the image resizing (change in image width/height or scaling), accordingly to the HTML specs:

> "For historical reasons, the coordinates must be interpreted relative to the displayed image after any stretching caused by the CSS 'width' and 'height' properties (or, for non-CSS browsers, the image element's width and height attributes - CSS browsers map those attributes to the aforementioned CSS properties)."

Tip:
Adding the `@scale` attribute on the `<imagemap>` element will scale both the image and areas.

3. Verify how the shapes look in the output. You can make the shapes visible by using **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:

```
.image-map-shape {
  fill: yellow;
  stroke: orange;
  fill-opacity: 0.3;
  stroke-opacity: 0.7;
}
.image-map-number-text {
  visibility: visible;
  stroke: red;

}
.image-map-number {
  fill: yellow;
  fill-opacity: 1;
}
```



Remember:
Make sure the `use.css.for.embedded.svg` parameter is set to 'yes' (it is by default).



Tip:
An `<svg>` with links (`<a>`) 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:

```
.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:

```
<syntaxdiagram id="syntaxdiagram_ok4_c1k_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:



Videos

Videos can be referenced in a DITA topic by using the `<object>` element:

```
<object data="path/to/video.mp4" outputclass="video"/>
```

Related information

Oxygen PDF Chemistry User Guide: Videos

How to Reference a Video Using a Key

Videos can also be referenced in DITA topics by using a key.

The key must be defined in the DITA map like this:

```
<keydef keys="video" href="path/to/video.mp4" format="mp4"/>
```

The key is referenced in the topic with the `@datakeyref` attribute within the `<object>` element:

```
<object datakeyref="video" outputclass="video" width="480" height="270"/>
```

How to Change Video Size

It is possible to set the size for your videos directly from a custom CSS stylesheet *(on page 1469)*:

```
.video {
  width: 480px;
  height: 270px;
}
```

Related information

Oxygen PDF Chemistry User Guide: Change the Video Size

How to Change the Videos Cover

By default, a placeholder is displayed in place of the video. When clicked, this placeholder launches the video. A popup is presented in Acrobat Reader to enable the Multimedia content (the document must be trusted for the video to launch).

It is possible to change this placeholder with a custom one by using the `-oxy-video-cover` property:

```
.video {
  -oxy-video-cover: url("files/cover.png");
}
```

How to Center Videos

It is possible to center the videos by centering their containers like this:

```
.video-container {
  text-align: center;
}
```

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 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.
- If all of the column width values are not declared in the table, the shares values (e.g. "2.5*") will not be used.

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):

```
*[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%;
  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:

```
*[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(*[class~="topic/body"] > *[class~="topic/table"][data-cols='5']),
*[class~="topic/topic"]:has(*[class~="topic/body"] > *[class~="topic/table"][data-cols='6']),
*[class~="topic/topic"]:has(*[class~="topic/body"] > *[class~="topic/table"][data-cols='7']),
*[class~="topic/topic"]:has(*[class~="topic/body"] > *[class~="topic/table"][data-cols='8']),
*[class~="topic/topic"]:has(*[class~="topic/body"] > *[class~="topic/table"][data-cols='9']),
*[class~="topic/topic"]:has(*[class~="topic/body"] > *[class~="topic/table"][data-cols='10']),
*[class~="topic/topic"]:has(*[class~="topic/body"] > * > *[class~="topic/table"][data-cols='5']),
*[class~="topic/topic"]:has(*[class~="topic/body"] > * > *[class~="topic/table"][data-cols='6']),
*[class~="topic/topic"]:has(*[class~="topic/body"] > * > *[class~="topic/table"][data-cols='7']),
*[class~="topic/topic"]:has(*[class~="topic/body"] > * > *[class~="topic/table"][data-cols='8']),
*[class~="topic/topic"]:has(*[class~="topic/body"] > * > *[class~="topic/table"][data-cols='9']),
*[class~="topic/topic"]:has(*[class~="topic/body"] > * > *[class~="topic/table"][data-cols='10']),
*[class~="topic/topic"]:has(*[class~="topic/body"] > * > * > *[class~="topic/table"][data-cols='5']),
*[class~="topic/topic"]:has(*[class~="topic/body"] > * > * > *[class~="topic/table"][data-cols='6']),
*[class~="topic/topic"]:has(*[class~="topic/body"] > * > * > *[class~="topic/table"][data-cols='7']),
*[class~="topic/topic"]:has(*[class~="topic/body"] > * > * > *[class~="topic/table"][data-cols='8']),
*[class~="topic/topic"]:has(*[class~="topic/body"] > * > * > *[class~="topic/table"][data-cols='9']),
*[class~="topic/topic"]:has(*[class~="topic/body"] > * > * > *[class~="topic/table"][data-cols='10']) {
  page: landscape-page;
}
```

Tip:
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 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:

```
*[class ~= "topic/table"] {
  overflow-wrap: break-word;
}
```

Related Information:

Hyphenation *(on page 1594)*
How to Enable/Disable Hyphenation for an Element *(on page 1597)*

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:

```
*[class ~= "topic/image"] {
  max-width: unset;
}
```

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:

```
*[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 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`.

```
<table class="- topic/table table ">
  <caption class="- topic/title title tablecap">
```

```
    <span class="table--title-label">Table
      <span class="table--title-label-number">1</span>
      <span class="table--title-label-punctuation">. </span>
    </span>
    <span class="table--title">
      <span class="- topic/ph topic/title-wrapper ph title-wrapper">Lorem Tablesum</span>
    </span>
  </caption>
  ...
</table>
```

To hide it, set its display to *none*:

```
.table--title-label {
  display:none;
}
```

How to Customize Rows, Columns and Cells

Common Use-Cases

Here are some common table use-cases and the CSS selectors for customizing table rows, columns, and cells. These example uses the `background-color` CSS property but any CSS property can be used (border, margin, padding, etc.).

- Select all non-header cells:

  ```
  *[class ~= "topic/tbody"] *[class ~= "topic/entry"] {
    background-color: lightgray;
  }
  ```

- Select some table rows (using `:nth-of-type()` pseudo-class):

  ```
  /* Select all even rows. */
  *[class ~= "topic/tbody"] *[class ~= "topic/row"]:nth-of-type(even) {
    background-color: lightgray;
  }
  /* Select the fourth row. */
  *[class ~= "topic/tbody"] *[class ~= "topic/row"]:nth-of-type(4) {
    background-color: yellow;
  }
  ```

- Select specific table columns (using `:nth-of-type()` pseudo-class):

  ```
  /* Select all odd columns. */
  *[class ~= "topic/tbody"] *[class ~= "topic/entry"]:nth-of-type(odd) {
    background-color: lightgray;
  }
  ```

```
/* Select the second column. */
*[class ~= "topic/tbody"] *[class ~= "topic/entry"]:nth-of-type(2) {
  background-color: yellow;
}
```

Applying Properties to Specific Elements

If you need to apply some properties to specific elements, you can use the DITA `@outputclass` attribute:

```
<table frame="none">
    <title>Flowers</title>
    <tgroup cols="3">
        <colspec colname="c1" colnum="1" colwidth="171pt"/>
        <colspec colname="c2" colnum="2" colwidth="99pt"/>
        <colspec colname="c3" colnum="3" colwidth="150pt"/>
        <thead>
            <row>
                <entry>Flower</entry>
                <entry>Type</entry>
                <entry>Soil</entry>
            </row>
        </thead>
        <tbody>
            <row>
                <entry>Chrysanthemum</entry>
                <entry outputclass="colored">perennial</entry>
                <entry>well drained</entry>
            </row>
            <row>
                <entry>Gardenia</entry>
                <entry>perennial</entry>
                <entry>acidic</entry>
            </row>
            <row outputclass="colored">
                <entry>Gerbera</entry>
                <entry>annual</entry>
                <entry>sandy, well-drained</entry>
            </row>
        </tbody>
    </tgroup>
</table>
```

In this case, the selector will be based on this `outputclass`:

```
*[class ~= "topic/table"] *[outputclass ~= "colored"] {
  background-color: yellow;
}
```

How to Add Stripes to a Table

To create a striped look for your tables, you can use the following CSS rules:

```
/* 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:

```
<table outputclass="stripes">...</table>
```

If you want to make all tables look the same, you can ignore this attribute and remove the `[outputclass ~= "stripes"]` 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 Disappearing Thin Lines or Cell Borders *(on page 1709)*.

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:

```
*[class ~= "topic/entry"] {

  -oxy-borders-conditionality: retain;

}
```

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_d1p_flb_crb">
  <title>Car Features</title>
  <tgroup 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"/>
    <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/>
        <entry>X</entry>
      </row>
...
```

Table 40. Car Features

Car Name	Features		
	Sport	All terrain	Family
Tesla Model S	X		X

The resulting output will be:

Car Name	Features		
	Sport	All terrain	Family
Tesla Model S	X		X
Nissan Leaf			X
Dacia Duster		X	X

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:

Car Name	Features		
	Sport	All terrain	Family
Tesla Model S	X		X
Nissan Leaf			X
Dacia Duster		X	X

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:

```
*[class~="topic/entry"][rotate='1'] {
  vertical-align:middle;
  white-space:normal;
}
```

The resulting output will be:

Car Name	Features		
	Sport	All terrain	Family
Tesla Model S	X		X
Nissan Leaf			X
Dacia Duster		X	X



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.

```
*[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.

```
*[class~="task/choptionhd"],
*[class~="task/choptionhd"],
*[class~="task/chdeschd"],
*[class~="task/choption"] {
 background-color: #EEEEEE;
 text-align: left;
}

*[class~="task/choicetable"] {
```

```
 border: 2pt solid #EEEEEE;
}

*[class~="task/choicetable"] *[class~="task/chrow"],
*[class~="task/choicetable"] *[class~="task/chhead"]{
 border-bottom: 2pt solid #EEEEEE;
}

*[class~="task/choicetable"] *[class~="topic/stentry"] {
 border-bottom: none;
 border-right: none;
}
```

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.

Programming Elements

Programming Elements are used to render lines of programming code. These elements have preserved line endings and use a monospace font in the output.

How to Change Font in Code Blocks

You can change fonts in code blocks to make them easier to read or compliant with your company fonts. To do so, add the following rule to your customization CSS *(on page 1469)*:

```
*[class ~= 'pr-d/codeblock'],
*[class ~= "pr-d/codeblock"] > code {
  font-family: 'Consolas', monospace;
}
```

Related information
Using Web Fonts
Using Local Font Files

How to Enable Syntax Highlight in Code Blocks

Note:
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-json
- language-yaml
- language-xml
- language-bourne
- language-c
- language-cmd
- language-cpp
- language-csharp
- language-css
- language-dtd
- language-ini
- language-java
- language-javascript
- language-lua
- language-perl
- language-powershell
- language-php
- language-python
- language-ruby
- language-sql
- language-xquery

For example, for a java snippet:

```
<codeblock outputclass="language-java">
   for (int i=0; i <100; i++) {
     // do something
   }
</codeblock>
```

The resulting HTML fragment in the merged HTML5 document is:

```
<pre class="+ topic/pre pr-d/codeblock pre codeblock language-java"
    xml:space="preserve">
  <strong class="hl-keyword" style="color:#7f0055">for</strong>
      (<strong class="hl-keyword" style="color:#7f0055">int</strong>
        i=<span class="hl-number">0</span>; i
           <<span class="hl-number">100</span>; i++) {
    <em class="hl-comment" style="color:#006400">// do something</em>
```

```
  }
</pre>
```

And in the output, it is rendered as:

```
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 1469)*:

```
.hl-keyword {
  color: red !important;
}
.hl-comment {
  color: blue !important;
}
```

How to Add Line Numbering in Code Blocks

Note:
This topic refers only to the **DITA Map PDF - based on HTML5 & CSS** transformation type.

You can add line numbering to make your code snippets easier to read. In the DITA topics, set the `@outputclass` attribute on the `<codeblock>` elements to the `show-line-numbers` value.

Note:
It is possible to use the `@outputclass="show-line-numbers"` together with any of the `language` `@outputclass` value (e.g. `@outputclass="language-java show-line-numbers"`).

For example, for a java snippet:

```
<codeblock outputclass="show-line-numbers">
public void convert(String systemId, InputStream is) {
 return new FileInputStream();
}
</codeblock>
```

The resulting HTML fragment in the merged HTML5 document is:

```
<pre class="+ topic/pre pr-d/codeblock pre codeblock show-line-numbers"
outputclass="show-line-numbers" xml:space="preserve">
<span class="+ topic/pre-new-line pre-new-line"></span>
<span class="+ topic/pre-new-line pre-new-line">
</span>public void convert(String systemId, InputStream is) {
<span class="+ topic/pre-new-line pre-new-line"></span> return new FileInputStream();
<span class="+ topic/pre-new-line pre-new-line"></span>}
<span class="+ topic/pre-new-line pre-new-line"></span></pre>
```

And in the output, it is rendered as:

```
1
2   public void convert(String systemId, InputStream is) {
3    return new FileInputStream();
4   }
5
```

How to Display Whitespaces in Code Blocks

Note:
This topic refers only to the **DITA Map PDF - based on HTML5 & CSS** transformation type.

You can display whitespace characters in code blocks to visualize indentation in the PDF. In the DITA topics, set the `@outputclass` attribute on the `<codeblock>` elements to the `show-whitespace` value.

Note:
It is possible to use the `@outputclass="show-whitespace"` together with any of the `language` or `show-line-numbers` `@outputclass` values (e.g. `@outputclass="language-java show-line-numbers show-whitespace"`).

For example, for a java snippet:

```
<codeblock outputclass="show-whitespace">
public void convert(String systemId, InputStream is) {
  return new FileInputStream();
}
</codeblock>
```

The resulting HTML fragment in the merged HTML5 document is:

```
<pre class="+ topic/pre pr-d/codeblock pre codeblock show-whitespaces"
outputclass="show-whitespaces" xml:space="preserve">
public·void·convert(String·systemId,·InputStream·is)·{
··return·new·FileInputStream();
```

```
}
</pre>
```

And in the output, it is rendered as:

```
public·void·convert(String·systemId,·InputStream·is)·{
··return·new·FileInputStream();
}
```

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 1469)*:

```
*[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 1665)*.

How to Enable Line Wrap in Code Phrases

By default, line wrapping does not apply on inline elements, which could cause some lines of code to bleed out of the page. To allow line wrapping, the property should be set on the parent block with the following rule in your customization CSS *(on page 1469)*:

```
*:has(*[class ~= 'pr-d/codeph']) {
  overflow-wrap: break-word;
}
```

Notes:

- It is possible to use `hyphens: auto` instead of `overflow-wrap: break-word`.
- It is possible to use the same rule for software domain elements (e.g. `<filepath>` or `<cmdname>`).

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 you can manually format the command using a continuation character:

- Windows

  ```
  $gist = ls -l * | count -n ^

  | some more
  ```

- Linux/macOS

  ```
  $gist = ls -l * | count -n \

  | some more
  ```

The command-line processor will automatically know that the command continues on the second line.

Notes

Notes contain an additional piece of information that calls attention to particular content. They may have various types (note, tip, fastpath, restriction, important, remember, attention, caution, danger, other).

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 1631)*.

How to Change Note Icons



Remember:

- The recommended icon format is SVG.
- The default size of the note icons is 24x24px.

To change the default icon for notes that do not have a `@type` attribute, add the following rule to your customization CSS *(on page 1469)*:

```
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:

```
div.warning {
  background-image: url("../img/warning.svg");
}
div.caution {
  background-image: url("../img/caution.svg");
}
div.trouble {
  background-image: url("../img/troubleshooting.svg");
}
```

For a note with `@type` attribute set to *other* and `@othertype` attribute set to *Safety*, add the following CSS rule:

```
div.note[type="other"][othertype=Safety] {
  background-image: url("../img/life-preserver.svg");
}
```

How to Change Note Colors

To change the background-color for notes that do not have a `@type` attribute, add the following rule to your customization CSS *(on page 1469)*:

```
*[class~="topic/note"]:not([class~="hazard-d/hazardstatement"]) {
  background-color: #50bbff;
}
```

For a note with a `@type` attribute set to *restriction*, add the following CSS rule:

```
*[class~="topic/note"].note_restriction {
  background-color: #ff5566;
}
```

For a note with `@type` attribute set to *other* and `@othertype` attribute set to *Safety*, add the following CSS rule:

```
*[class~="topic/note"][type = "other"][othertype = Safety] {
  background-color: #ffaa00;
}
```

Hazard

Hazards (embodied by the `<hazardstatement>` element) contain warning information. They are based on ANSI Z535 and ISO 3864 standards and may have various values set for the type (`note`, `tip`, `fastpath`, `restriction`, `important`, `remember`, `attention`, `caution`, `notice`, `danger`, `warning`, `other`).

How to Customize Hazard Statements

It is possible to customize hazard statements (`<hazardstatement>` elements) to add an icon, change the borders, change the colors, and more. For example, to customize the *attention* type hazards, add the following rules to your customization CSS *(on page 1469)*:

```
*[class ~= "hazard-d/hazardstatement"][type = "attention"] {
  border-width: 2px;
}
*[class ~= "hazard-d/hazardstatement"][type = "attention"],
*[class ~= "hazard-d/hazardstatement"][type = "attention"] td {
  border-color: #ff9900;
}

.hazardstatement .hazardstatement--attention {
  text-align: left;
  padding-left: 3em;
  background-position: .5em .3em;
  background-image: url("danger.svg");
  background-repeat: no-repeat;
  background-color: #ff9900;
  border-color: #ff9900;
  color: black;
}
```

Tip:
The default icon size is 24px and its URL is relative to the CSS stylesheet.

How to Customize Other Type Hazards

It is possible to create custom hazard types by using the `@type` and `@othertype` attributes. For example, to add a high voltage hazard in a microwave manual:

```
<hazardstatement id="hazardstatement_vzy_zdc_syb" type="other" othertype="HIGH_VOLTAGE">
    <messagepanel id="messagepanel_wzy_zdc_syb">
        <typeofhazard>Electrical Shock</typeofhazard>
        <howtoavoid>Do not disassemble or repair the microwave yourself.</howtoavoid>
    </messagepanel>
    <hazardsymbol id="hazardsymbol_z4t_gjc_syb" href="electricity_icon.svg"/>
</hazardstatement>
```

Tip:
SVG images are preferred for the `<hazardsymbol>` and you should set both `@height="1em"` and `@width="1em"` to obtain a rendering that is similar to default hazards.

To customize the hazard, add the following rules to your customization CSS *(on page 1469)*:

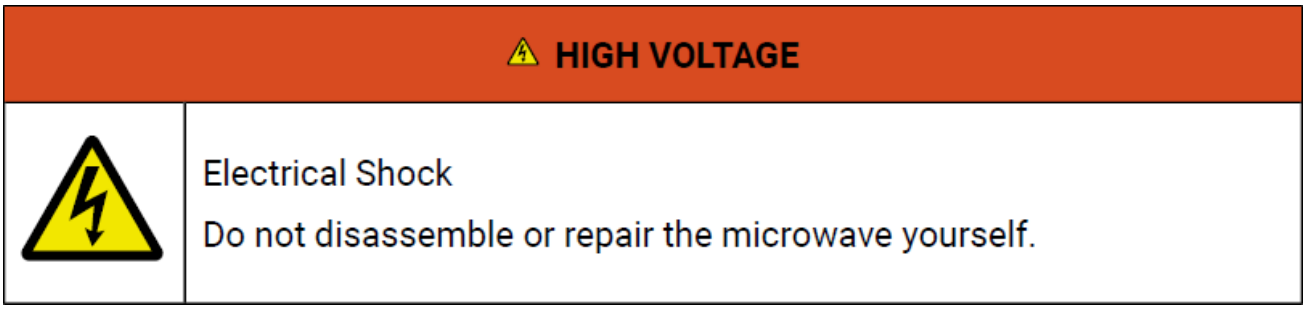
```
/* Change the header color. */
*[othertype ~= "HIGH_VOLTAGE"] .hazardstatement--other {
```

```
  content: "HIGH VOLTAGE"; /* Change the hazard text */
  background-color: #d84b20;
  color: unset;
}

/* Show logo in the header. */
*[othertype ~= "HIGH_VOLTAGE"] .hazardstatement--other::before {
  padding: .5rem;
  content: url("electricity_icon.svg");
}

/* Show logo in the left cell. */
*[othertype ~= "HIGH_VOLTAGE"] th {
  table-column-span: 2 !important;
}
*[othertype ~= "HIGH_VOLTAGE"] .hazardstatement--logo-col {
  display: table-column !important;
}
*[othertype ~= "HIGH_VOLTAGE"] td:first-of-type {
  display: table-cell !important;
}
*[othertype ~= "HIGH_VOLTAGE"] .hazardsymbol {
  height: 4em; /* Change the symbol dimension */
}
```

The result in the PDF output would look like this:



How to Remove the Hazard Symbol

Remove the Hazard Symbol from the Header

It is possible to remove the symbol that appears in the hazard header (for example, if you consider it redundant with the symbol in the column).

To remove it, simply hide its container with the following rules in your customization CSS *(on page 1469)*:

```
.hazardsymbol-container {
  display: none;
}
```

Remove the Hazard Symbol on the left of the Message Panel

It is possible to remove the column that contains the hazard symbol (for example, if you want to maximize the message panel).

To remove the caution symbols, for example, add the following rules to your customization CSS *(on page 1469)*:

```
.hazardstatement_caution .hazardstatement--logo-col {
  display: none;
}
.hazardstatement_caution th {
  table-column-span: 1
}
.hazardstatement_caution td:first-of-type {
  display: none;
}
```

Tasks

Tasks provide step-by-step instructions that enable a user to perform an operation.

How to Add Requirements Labels

It is possible to add tasks headings by setting the `args.gen.task.lbl` parameter in the transformation. However, **Machinery Tasks** have some extra required elements. It is possible to add labels for these requirements by adding the following rules to your customization CSS *(on page 1469)*:

```
*[class ~= "taskreq-d/reqconds"]:before,
*[class ~= "taskreq-d/reqpers"]:before,
*[class ~= "taskreq-d/supequip"]:before,
*[class ~= "taskreq-d/supplies"]:before,
*[class ~= "taskreq-d/spares"]:before,
*[class ~= "taskreq-d/safety"]:before {
  font-weight: bold;
  padding-left: 20px;
}

*[class ~= "taskreq-d/reqconds"]:before {
  content: "Conditions: ";
}
*[class ~= "taskreq-d/reqpers"]:before {
```

```
  content: "Personnel: ";
}
*[class ~= "taskreq-d/personnel"]:before {
  content: "Number of workers: " !important;
}
*[class ~= "taskreq-d/perscat"]:before {
  content: "Category: " !important;
}
*[class ~= "taskreq-d/perskill"]:before {
  content: "Skill level: " !important;
}
*[class ~= "taskreq-d/esttime"]:before {
  content: "Time estimate: " !important;
}
*[class ~= "taskreq-d/supequip"]:before {
  content: "Equipment: " !important;
}
*[class ~= "taskreq-d/supplies"]:before {
  content: "Supplies: " !important;
}
*[class ~= "taskreq-d/spares"]:before {
  content: "Spares:";
}
*[class ~= "taskreq-d/safety"]:before {
  content: " Safety:";
}
```

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 1469)*:

```
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
                        class="- topic/tmmark tmmark ">™</span></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 Change the Publication Title Using a Parameter

In the following example, a custom parameter is used as the publication's main title. The parameter is defined in the transformation scenario as:

```
args.css.param.main-title="${ask('Main title', generic, 'Flowers')}"
```

When running the scenario, a dialog box will appear prompting for a title, with 'Flowers' set as the default.

Then in the CSS, the attribute value is extracted and used as follows:

```
/* Update the title */
*[class ~= "front-page/front-page-title"] > *[class ~= "topic/title"] {
  content: oxy_xpath('/*/@*[local-name()="main-title"][1]');
}

/* Update the bookmark */
*[class ~= "front-page/front-page-title"] > *[class ~= "topic/title"]:not([class ~= "bookmap/booktitle"]),
*[class ~= "front-page/front-page-title"] > *[class ~= "bookmap/booktitle"] > *[class ~= "bookmap/mainbooktitle"] {
  bookmark-label: content(before) oxy_xpath('/*/@*[local-name()="main-title"][1]');
}

/* Update metadata and tab name */
*[class ~= "front-page/front-page-title"] *[class ~= "topic/title"]:not([class ~= 'bookmap/booktitle']),
*[class ~= "front-page/front-page"] *[class ~= "bookmap/booktitle"] > *[class ~= "bookmap/mainbooktitle"] {
  -oxy-pdf-meta-title: oxy_xpath('/*/@*[local-name()="main-title"][1]');
}
```

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:

```
args.css.param.only-chapters-in-toc="yes"
```

The CSS that hides the *topicrefs* at level 2 or more:

```
:root[only-chapters-in-toc='yes'] *[class ~= "toc/toc"]
        > *[class ~= "map/topicref"]> *[class ~= "map/topicref"] {
  display:none;
}
```

The `:root[a|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:

```
args.css.param.page-size="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:

```
args.css.param.cover-page="file:/path/to/cover-page.svg"
```

Then in the CSS, the attribute value is extracted and used as follows:

```
@page front-page {
    background-image: url(oxy_xpath('/*/@*[local-name()="cover-page"][1]'));
}
```

Controlling the Publication Content

It is possible to control the publication content by either using a DITA Bookmap or by specifying additional parameters in the transformation scenario.

Why Use a Bookmap

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 some 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:

```
<!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>
```

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:

```
  <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:

```
  ...
  <chapter href="topics/installation.dita" />
  ...
```

Note:
If you need more levels, you can use the `<part>` element and then group your chapters under it.

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 a list of tables. At the very end, there is a topic with some thank you notes:

```
  <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 more control over the final content of the publication than a regular map. It also offers more options for controlling the metadata that will go in the PDF (see the Metadata *(on page 1529)* 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 1444)*.

Related Information:

How to Remove Entries from the TOC *(on page 1563)*

How to Hide the TOC *(on page 1564)*

How to Extract Parts and Chapters as Individual PDF Files

Note:
This topic is only applicable for the **DITA Map PDF - based on HTML5 & CSS** transformation scenario.

There are two ways to tell the processor to extract some topics as individual PDF files:

1. Using the `args.css.param.pdf-split-mode` parameter: This option allows you to create individual PDF files for each part or chapter of a publication.
2. Using the `@outputclass="pdf-split"` attribute: This option allows you to create separate PDF files only for selected topics.

Remember:

- Only parts, chapters and topics with an `@id` attribute will be extracted.
- Individual PDFs will be generated alongside the main PDF.
- Extracted PDFs will preserve the styling of the original document.

Tip:
Generating additional PDFs may produce numerous files. To keep the output directory organized, it is recommended to disable the `args.keep.output.debug.files` parameter.

Since each part or chapter is independent and can be read separately, the page, figure, and table counters should restart at the beginning of each chapter. You can control this behavior using the `args.css.param.numbering` parameter, setting its value to `deep-chapter-scope`.

Beyond numbering, it is possible to enforce the creation of a chapter-specific table of contents *(on page 1564)*.

Related information
Split Mode Parameter *(on page 1445)*

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 1471)* 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 1471)* 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

This section contains some common examples of customizations using both XSLT and CSS stylesheets. These stylesheets must be used as CSS resources and XSLT extension points inside an *Oxygen Publishing Template*.

> **Tip:**
> The XSLT extension points are called on specific files during two different phases of the process: merged2merged *(on page 1441)* and merged2html5 *(on page 1442)*.

How to Style Codeblocks with a Zebra Effect

A possible requirement for your `<codeblock>` elements could be to alternate the background color on each line. Some advantages of this technique is that you can clearly see when text from the `<codeblock>` is wrapped.

> **Note:**
> Adding this styling will remove syntax highlights on codeblocks.

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 1464)*.
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 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="3.0">
```

```
  <xsl:template match="*[contains(@class, ' pr-d/codeblock ')]">
    <xsl:variable name="nm">
      <xsl:next-match/>
    </xsl:variable>
    <xsl:apply-templates select="$nm" mode="zebra"/>
  </xsl:template>

  <xsl:template match="node() | @*" mode="zebra">
    <xsl:copy>
      <xsl:apply-templates select="node() | @*" mode="#current"/>
    </xsl:copy>
  </xsl:template>

  <xsl:template match="*[contains(@class, ' pr-d/codeblock ')]" mode="zebra">
    <xsl:element name="{name()}">
      <xsl:copy-of select="@*"/>
      <xsl:attribute name="class" select="concat(@class, ' zebra')"/>
      <xsl:analyze-string regex="\n" select=".">
        <xsl:matching-substring/>
        <xsl:non-matching-substring>
          <div>
            <xsl:value-of select="."/>
          </div>
        </xsl:non-matching-substring>
      </xsl:analyze-string>
    </xsl:element>
  </xsl:template>

</xsl:stylesheet>
```

5. Open the *template descriptor file (on page 1459)* 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>
```

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:

```
.zebra {
  padding: 0;
}

.zebra > *:nth-of-type(odd) {
  background-color: lightgray;
}
```

7. Open the *template descriptor file (on page 1459)* 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>
```

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 1464)*.
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 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="3.0">

  <xsl:template match="*[contains(@class, ' topic/related-links ')]">
    <!-- Remove. -->
```

```
  </xsl:template>

</xsl:stylesheet>
```

5. Open the *template descriptor file (on page 1459)* 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"/>
    </xslt>
```

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 1464)*.
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 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="3.0">

  <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:value-of select="."/>
      </xsl:non-matching-substring>
    </xsl:analyze-string>

  </xsl:template>
</xsl:stylesheet>
```

5. Open the *template descriptor file (on page 1459)* 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"/>
    </xslt>
```

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 1464)*.
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 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="3.0">

  <xsl:template match="*[contains(@class, ' topic/dl ')]">
    <xsl:call-template name="setaname"/>
    <xsl:apply-templates select="
      *[contains(@class,
      ' ditaot-d/ditaval-startprop ')]" mode="out-of-line"/>
    <!-- Wrap in a table -->
    <table>
      <xsl:call-template name="commonattributes"/>
      <xsl:call-template name="setid"/>
      <xsl:apply-templates/>
    </table>
    <xsl:apply-templates select="
      *[contains(@class,
      ' ditaot-d/ditaval-endprop ')]" mode="out-of-line"/>
  </xsl:template>

  <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 1459)* 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>
```

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 1464)*.
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 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"
  xmlns:opentopic-func="http://www.idiominc.com/opentopic/exsl/function"
  exclude-result-prefixes="xs opentopic-func"
  version="3.0">

  <!--
    Match only top level tables (i.e tables that are not nested in other tables),
    that contains some 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')]">
        <!--
          Try to preserve the footnote ID, if available, so that the xrefs will have a target.
        -->
        <li class="- topic/li " id="{if(@id) then @id else generate-id(.)}"
          outputclass="table-fn">
          <xsl:copy-of select="@callout"/>
          <xsl:apply-templates select="node()"/>
        </li>
      </xsl:for-each>
    </ol>
  </xsl:template>

  <!--
    The footnotes that have an ID must be ignored, they are accessible only
    through existing xrefs (already present in the merged.xml file).
```

```
  The above template already made a copy of these footnotes in the OL element
  so it is not a problem if markup is not generated for them in the cell.
-->
<xsl:template
  match="*[contains(@class, 'topic/entry')]//*[contains(@class, 'topic/fn')][@id]"/>

<!--
  The xrefs to footnotes with IDs inside table-cells. We need to recalculate
  their indexes if their referenced footnote is also in the table.
-->
<xsl:template match="*[contains(@class, 'topic/xref')][@type='fn']
  [ancestor::*[contains(@class, 'topic/entry')]]">
  <xsl:param name="top-level-table" tunnel="yes"/>
  <xsl:variable name="destination" select="opentopic-func:getDestinationId(@href)"/>
  <xsl:variable name="fn" select="
    $top-level-table//*[contains(@class, 'topic/fn')][@id = $destination]"/>
  <xsl:choose>
    <xsl:when test="$fn">
      <!-- There is a reference in the table, recalculate index. -->
      <xsl:variable name="fn-number" select="
        index-of($top-level-table//*[contains(@class, 'topic/fn')], $fn)"/>
      <xsl:copy>
        <xsl:apply-templates select="@*"/>
        <xsl:apply-templates select="$fn/@callout"/>
        <xsl:apply-templates select="node()
          except (text(), *[contains(@class, 'hi-d/sup')])"/>
        <sup class="+ topic/ph hi-d/sup ">
          <xsl:apply-templates select="child::*[contains(@class, 'hi-d/sup')]/@*"/>
          <xsl:value-of select="$fn-number"/>
        </sup>
      </xsl:copy>
    </xsl:when>
    <xsl:otherwise>
      <!-- There is no reference in the table, keep original index. -->
      <xsl:next-match/>
    </xsl:otherwise>
  </xsl:choose>
</xsl:template>

<!--
  The footnotes without ID inside table-cells. They are copied in the OL element, but have
  no xrefs pointing to them (because they have no ID), so xrefs are generated.
```

```
  -->
  <xsl:template
    match="*[contains(@class, 'topic/entry')]//*[contains(@class, 'topic/fn')][not(@id)]">
    <!-- Determine the footnote index in the document order. -->
    <xsl:param name="top-level-table" tunnel="yes"/>
    <xsl:variable name="fn-number" select="
      index-of($top-level-table//*[contains(@class, 'topic/fn')], .)"/>
    <xref type="fn" class="- topic/xref "
      href="#{generate-id(.)}" outputclass="table-fn-call">
      <xsl:copy-of select="@callout"/>
      <!-- Generate an extra <sup>, identical to what DITA-OT generates for other xrefs. -->
      <sup class="+ topic/ph hi-d/sup ">
        <xsl:value-of select="$fn-number"/>
      </sup>
    </xref>
  </xsl:template>


</xsl:stylesheet>
```

5. Open the *template descriptor file (on page 1459)* 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"/>
   </xslt>
```

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'] {
  line-height: none;
}
*[class ~= "topic/table"] *[class ~= "topic/xref"][type = 'fn'][callout] *[class ~= "hi-d/sup"] {
  content: oxy_xpath("ancestor::*[contains(@class, 'topic/xref')]/@callout");
}


/* Customize the list containing all the table footnotes. */
```

```
*[outputclass ~= 'table-fn-container'] {
  border-top: 1pt solid black;
  counter-reset: table-footnote;
}

/* Customize footnotes display, below the table. */
*[outputclass ~= 'table-fn'] {
  font-size: smaller;
  counter-increment: table-footnote;
}
*[outputclass ~= 'table-fn']::marker {
  font-size: smaller;
  content: "(" counter(table-footnote) ")";
}
*[outputclass ~= 'table-fn'][callout]::marker {
  content: "(" attr(callout) ")";
}

/* Customize xrefs pointing to footnotes, inside the table. */
*[class ~= "topic/table"] *[class ~= "topic/xref"][type = 'fn'] {
  color: unset;
  text-decoration: none;
}
*[class ~= "topic/table"] *[class ~= "topic/xref"][type = 'fn']:after {
  content: none;
}
*[class ~= "topic/table"] *[class ~= "topic/xref"][type = 'fn'] *[class ~= "hi-d/sup"]:before {
  content: "(";
}
*[class ~= "topic/table"] *[class ~= "topic/xref"][type = 'fn'] *[class ~= "hi-d/sup"]:after {
  content: ")";
}
```

7. Open the *template descriptor file (on page 1459)* 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>
```

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 1464)*.
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 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="3.0">

  <!-- 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}}">
      <!-- The cell contains a scientific number like 1.23e-08. -->
      <xsl:matching-substring>
        <xsl:variable name="text" select="concat(substring-before(., 'e'),
          'e&#8203;', substring-after(., 'e'))"/>
        <xsl:value-of select="$text"/>
      </xsl:matching-substring>
      <xsl:non-matching-substring>
        <xsl:value-of select="."/>
      </xsl:non-matching-substring>
    </xsl:analyze-string>
  </xsl:template>

</xsl:stylesheet>
```

5. Open the *template descriptor file (on page 1459)* 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>
```

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:

```
...
<steps>
   <step>
        <cmd>My single step</cmd>
   </step>
</steps>
...
```

Normally, the output will be rendered as:

```
1. The step
```

instead of:

```
o 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 1464)*.
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 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="3.0">

  <xsl:template match="*[contains(@class, ' task/step ')][count(../*[contains
(@class, ' task/step ')]) = 1]">
    <xsl:copy>
      <xsl:copy-of select="@*"/>
      <xsl:attribute name="outputclass" select="concat(@outputclass, ' single ')"/>
      <xsl:apply-templates/>
    </xsl:copy>
  </xsl:template>

</xsl:stylesheet>
```

5. Open the *template descriptor file (on page 1459)* 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>
```

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:

```
*[outputclass ~= "single"] {
  list-style-type:circle !important;
  margin-left:2em;
}
```

7. Open the *template descriptor file (on page 1459)* 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>
```

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 1464)*.
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 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="3.0">

  <xsl:template match="
      *[contains(@class, 'topic/created')]/@date |
      *[contains(@class, 'topic/revised')]/@modified">
    <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 1459)* 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"/>
</xslt>
```

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 1464)*.
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:

```
<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: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="$result" mode="remove-extra-links">
      <xsl:with-param name="is-first-term-with-same-href"
        select="generate-id(.) = generate-id($first-term-with-same-href)" tunnel="yes"/>
    </xsl:apply-templates>
  </xsl:template>

  <xsl:template match="node() | @*" mode="remove-extra-links">
    <xsl:copy>
      <xsl:apply-templates select="node() | @*" mode="#current"/>
    </xsl:copy>
  </xsl:template>

  <xsl:template match="a" mode="remove-extra-links">
    <xsl:param name="is-first-term-with-same-href" tunnel="yes"/>
    <xsl:choose>
      <!-- Process the first term as a link -->
      <xsl:when test="$is-first-term-with-same-href">
        <xsl:next-match/>
      </xsl:when>
      <xsl:otherwise>
        <!-- Process the other terms as text -->
        <xsl:copy-of select="child::*"/>
      </xsl:otherwise>
    </xsl:choose>
  </xsl:template>

</xsl:stylesheet>
```

5. Open the *template descriptor file (on page 1459)* 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>
```

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 Glossary as a Table

Suppose you want to display the content of your Glossary as a table, to condense the information for one entry on a single line.



Remember:
Make sure all the glossary is contained within a single `<glossgroup>` element.

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 1464)*.
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:

```
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns:dita2html="http://dita-ot.sourceforge.net/ns/200801/dita2html"
  exclude-result-prefixes="xs dita2html" version="3.0">

  <!-- Create a table that will contain all the glossentries contained in the glossgroup. -->
  <xsl:template name="gen-topic">
    <xsl:param name="nestlevel" as="xs:integer">
      <xsl:choose>
        <!-- Limit depth for historical reasons, could allow any depth. -->
        <!-- Previously limit was 5. -->
        <xsl:when
          test="count(ancestor::*[contains(@class, ' topic/topic ')]) > 9"
          >9</xsl:when>
        <xsl:otherwise>
          <xsl:sequence
            select="count(ancestor::*[contains(@class, ' topic/topic ')])"/>
        </xsl:otherwise>
      </xsl:choose>
    </xsl:param>
```

```
<xsl:choose>
  <xsl:when test="parent::dita and not(preceding-sibling::*)">
    <!-- Do not reset xml:lang if it is already set on <html> -->
    <!-- Moved outputclass to the body tag -->
    <!-- Keep ditaval based styling at this point -->
    <!-- (replace DITA-OT 1.6 and earlier call to gen-style) -->
    <xsl:apply-templates
      select="*[contains(@class, ' ditaot-d/ditaval-startprop ')]/@style"
      mode="add-ditaval-style"/>
  </xsl:when>
  <xsl:otherwise>
    <xsl:call-template name="commonattributes">
      <xsl:with-param name="default-output-class"
        select="concat('nested', $nestlevel)"/>
    </xsl:call-template>
  </xsl:otherwise>
</xsl:choose>
<xsl:call-template name="gen-toc-id"/>
<xsl:call-template name="setidaname"/>
<xsl:choose>
  <xsl:when test="contains(@class, 'glossgroup/glossgroup')">
    <!-- Custom processing for glossgroup. -->
    <xsl:apply-templates select="*[contains(@class, 'topic/title')]"/>
    <table class="- glossgroup/table table">
      <thead class="- glossgroup/thead thead">
        <tr class="- glossgroup/row row">
          <th class="- glossgroup/entry entry">Acronym</th>
          <th class="- glossgroup/entry entry">Term</th>
          <th class="- glossgroup/entry entry">Full Term</th>
          <xsl:if
            test="exists(//*[contains(@class, 'glossentry/glossdef')])">
            <th class="- glossgroup/entry entry">Definition</th>
          </xsl:if>
        </tr>
      </thead>
      <xsl:apply-templates
        select="*[contains(@class, 'glossentry/glossentry')]"/>
    </table>
    <xsl:apply-templates select="
        * except (*[contains(@class, 'topic/title')]
        | *[contains(@class, 'glossentry/glossentry')])"/>
  </xsl:when>
```

```
    <xsl:otherwise>
      <!-- Default processing. -->
      <xsl:apply-templates/>
    </xsl:otherwise>
  </xsl:choose>
</xsl:template>

<!-- Create a row for each glossentry. -->
<xsl:template
  match="*[contains(@class, 'glossentry/glossentry')]
  [parent::*[contains(@class, 'glossgroup/glossgroup')]]">
  <xsl:variable name="glossentry" as="node()">
    <xsl:next-match/>
  </xsl:variable>
  <tr>
    <xsl:copy-of select="$glossentry/@*"/>
    <xsl:copy-of
      select="$glossentry/*[contains(@class, 'glossentry/glossAlt')]"/>
    <xsl:copy-of
      select="$glossentry/*[contains(@class, 'glossentry/glossterm')]"/>
    <xsl:copy-of
      select="$glossentry/*[contains(@class, 'glossentry/glossSurfaceForm')]"/>
    <xsl:copy-of
      select="$glossentry/*[contains(@class, 'glossentry/glossdef')]"/>
    <xsl:copy-of select="
        $glossentry/* except $glossentry/*[contains(@class, 'glossentry/glossAlt')
        or contains(@class, 'glossentry/glossterm')
        or contains(@class, 'glossentry/glossSurfaceForm')
        or contains(@class, 'glossentry/glossdef')]"/>
  </tr>
</xsl:template>

<!-- Process only glossBody's children nodes. -->
<xsl:template
  match="*[contains(@class, 'glossentry/glossBody')]
  [ancestor::*[contains(@class, 'glossgroup/glossgroup')]]">
  <xsl:apply-templates/>
</xsl:template>

<!-- Create a cell for each glossterm, glossSurfaceForm and glossAlt. -->
<xsl:template match="
    *[contains(@class, 'glossentry/glossterm')]
```

```
      [ancestor::*[contains(@class, 'glossgroup/glossgroup')]] |
      *[contains(@class, 'glossentry/glossSurfaceForm')]
      [ancestor::*[contains(@class, 'glossgroup/glossgroup')]] |
      *[contains(@class, 'glossentry/glossAlt')]
      [ancestor::*[contains(@class, 'glossgroup/glossgroup')]]">
    <xsl:variable name="glossContent" as="node()">
      <xsl:next-match/>
    </xsl:variable>
    <td>
      <xsl:copy-of select="$glossContent/@*"/>
      <xsl:copy-of select="normalize-space(string-join($glossContent//text()))"
      />
    </td>
  </xsl:template>

  <!-- Create a cell for each glossdef. -->
  <xsl:template
    match="*[contains(@class, 'glossentry/glossdef')]
    [ancestor::*[contains(@class, 'glossgroup/glossgroup')]]">
    <td>
      <xsl:call-template name="commonattributes"/>
      <xsl:apply-templates/>
    </td>
  </xsl:template>

</xsl:stylesheet>
```

5. Open the *template descriptor file (on page 1459)* 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>
```

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:

```
*[class ~= "glossgroup/table"] {
  width: 100%;
  border: 1px solid black;
  border-collapse: collapse;
}

*[class ~= "glossgroup/table"] th {
  background-color: lightgray;
}

*[class ~= "glossgroup/table"] th,
*[class ~= "glossgroup/table"] td {
  border: 1px solid black;
  padding: 0.3em !important;
  vertical-align: inherit !important;
}

/* Remove glossSurfaceForm */
th:nth-of-type(3),
*[class ~= "glossentry/glossSurfaceForm"] {
  display: none;
}

/* Discard the default glossterm layout */
*[class ~= "glossentry/glossterm"] {
  font-size: unset;
  font-weight: unset;
}
```

Note:
The `<glossSurfaceForm>` removal part is optional. It is present as an example of how to fully remove a column.

7. Open the *template descriptor file (on page 1459)* 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>
```

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 Include Sections in the Mini TOC

By default, the *Mini TOC* only displays the child topics of a given chapter topic. To add the possibility of also displaying the child sections, 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 1464)*.
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:

```
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="3.0"
  xmlns:ditaarch="http://dita.oasis-open.org/architecture/2005/"
  xmlns:opentopic-index="http://www.idiominc.com/opentopic/index"
  xmlns:opentopic="http://www.idiominc.com/opentopic"
  xmlns:oxy="http://www.oxygenxml.com/extensions/author" xmlns:saxon="http://saxon.sf.net/"
  xmlns:xs="http://www.w3.org/2001/XMLSchema" exclude-result-prefixes="#all">
  <xsl:template match="*[contains(@class, ' topic/topic ')]">
    <xsl:choose>
      <xsl:when test="
          ($args.chapter.layout = 'MINITOC' or
          $args.chapter.layout = 'MINITOC-BOTTOM-LINKS') and
          oxy:is-chapter(/, oxy:get-topicref-for-topic(/, @id)) and
          *[contains(@class, ' topic/topic ')]">
        <!-- Minitoc. -->
        <xsl:copy>
          <xsl:apply-templates select="@*"/>
          <xsl:apply-templates select="*[contains(@class, ' topic/title ')]"/>
          <xsl:apply-templates select="*[contains(@class, ' topic/prolog ')]"/>
          <xsl:apply-templates select="*[contains(@class, ' topic/titlealts ')]"/>
          <div>
            <xsl:choose>
              <xsl:when test="$args.chapter.layout = 'MINITOC'">
                <xsl:attribute name="class">- topic/div chapter/minitoc </xsl:attribute>
                <xsl:call-template name="generate-minitoc-links"/>
                <xsl:call-template name="generate-minitoc-desc"/>
              </xsl:when>
```

```
            <xsl:when test="$args.chapter.layout = 'MINITOC-BOTTOM-LINKS'">
<xsl:attribute name="class">- topic/div chapter/minitoc chapter/minitoc-bottom </xsl:attribute>
              <xsl:call-template name="generate-minitoc-desc"/>
              <xsl:call-template name="generate-minitoc-links"/>
            </xsl:when>
          </xsl:choose>
        </div>
        <xsl:apply-templates select="*[contains(@class, ' topic/topic ')]"/>
      </xsl:copy>
    </xsl:when>
    <xsl:otherwise>
      <!-- No minitoc. -->
      <xsl:next-match/>
    </xsl:otherwise>
  </xsl:choose>
</xsl:template>
<!--
  The chapter topic content. This has the role of describing the chapter.
-->
<xsl:template name="generate-minitoc-desc">
  <div class="- topic/div chapter/minitoc-desc ">
    <xsl:apply-templates select="
        *[not(contains(@class, ' topic/title ')) and
        not(contains(@class, ' topic/prolog ')) and
        not(contains(@class, ' topic/titlealts ')) and
        not(contains(@class, ' topic/topic ')) and
        not(contains(@class, ' topic/section '))
        ]"/>
  </div>
</xsl:template>
<!--
  Child links.
-->
<xsl:template name="generate-minitoc-links">
  <div class="- topic/div chapter/minitoc-links ">
    <related-links class="- topic/related-links ">
      <linklist class="- topic/linklist ">
        <desc class="- topic/desc ">
          <ph class="- topic/ph chapter/minitoc-label ">
            <xsl:call-template name="getVariable">
              <xsl:with-param name="id" select="'Mini Toc'"/>
            </xsl:call-template>
```

```
          </ph>
        </desc>
        <xsl:apply-templates select="
            *[contains(@class, ' topic/topic ')] |
            descendant-or-self::*[contains(@class, ' topic/section ')]"
          mode="in-this-chapter-list"/>
      </linklist>
    </related-links>
  </div>
 </xsl:template>
 <xsl:template match="
    *[contains(@class, ' topic/topic ')
    or contains(@class, ' topic/section ')]" mode="in-this-chapter-list">
  <xsl:variable name="link-type" select="
    if (contains(@class, ' topic/section ')) then
      'section'
    else
      'topic'"/>
  <link class="- topic/link " href="#{@id}" type="{$link-type}" role="child">
    <linktext class="- topic/linktext ">
      <xsl:value-of select="*[contains(@class, ' topic/title ')]"/>
    </linktext>
  </link>
 </xsl:template>
</xsl:stylesheet>
```

5. Open the *template descriptor file (on page 1459)* 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.mergedmerged"
      file="xslt/merged2mergedExtension.xsl"/>
  </xslt>
  <parameters>
    <parameter name="args.chapter.layout" value="MINITOC"/>
  </parameters>
```



Note:
This solution works also with `args.chapter.layout` set to `MINITOC-BOTTOM-LINKS`.

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 Add a Link to the TOC

For making the navigation easier in the PDF, you may want to add a link that sends the reader back to the table of contents. To add this link, 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 1464)*.
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 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="3.0">

  <!-- Add an anchor after the TOC title. -->
  <xsl:template match="*[contains(@class, 'toc/title')]" mode="div-it">
    <div>
      <xsl:attribute name="class" select="'- toc/anchor anchor'"/>
      <xsl:attribute name="id" select="'toc-anchor'"/>
    </div>
    <xsl:next-match/>
  </xsl:template>

</xsl:stylesheet>
```

5. Open the *template descriptor file (on page 1459)* 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>
```

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:

```
@page chapter:first:left:right {
  @top-right {
    content: "Back to Table of Contents";
    -oxy-link: "#toc-anchor";
    color: #337ab7;
  }
}
@page chapter:left:right {
  @top-right {
    content: "Back to Table of Contents";
    -oxy-link: "#toc-anchor";
    color: #337ab7;
  }
}
```

7. Open the *template descriptor file (on page 1459)* 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>
```

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 Repeat Note Titles After a Page Break

Suppose that you have large notes that split between pages or columns and you want the note icon and title to be displayed on the next page/column. 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 1464)*.
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 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="3.0">

  <!-- Display notes titles and content in table cells. -->
  <xsl:template match="*" mode="process.note.common-processing">
    <xsl:param name="type" select="@type"/>
    <xsl:param name="title">
      <xsl:call-template name="getVariable">
        <xsl:with-param name="id" select="concat(upper-case(substring($type, 1, 1)),
substring($type, 2))"/>
      </xsl:call-template>
    </xsl:param>
    <table>
      <xsl:call-template name="commonattributes">
        <xsl:with-param name="default-output-class"
select="string-join(($type, concat('note_', $type)), ' ')"/>
      </xsl:call-template>
      <xsl:call-template name="setidaname"/>
 <!-- Normal flags go before the generated title; revision flags only go on the content. -->
      <xsl:apply-templates select="*[contains(@class, ' ditaot-d/ditaval-startprop ')]
/prop" mode="ditaval-outputflag"/>
      <thead>
        <tr>
          <th class="note__title">
            <xsl:copy-of select="$title"/>
            <xsl:call-template name="getVariable">
              <xsl:with-param name="id" select="'ColonSymbol'"/>
            </xsl:call-template>
          </th>
        </tr>
      </thead>
      <tbody>
        <tr>
```

```
          <td>
            <xsl:text> </xsl:text>
            <xsl:apply-templates select="*[contains(@class, ' ditaot-d/ditaval-startprop ')]
/revprop" mode="ditaval-outputflag"/>
            <xsl:apply-templates/>
            <!-- Normal end flags and revision end flags both go out after the content. -->
            <xsl:apply-templates select="*[contains(@class, ' ditaot-d/ditaval-endprop ')]"
mode="out-of-line"/>
          </td>
        </tr>
      </tbody>
    </table>
  </xsl:template>

</xsl:stylesheet>
```

5. Open the *template descriptor file (on page 1459)* 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>
```

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:

```
table.note th,
table.note td {
  text-align: left;
  padding: .75em .5em .75em 3em;
}

table.note {
  background-repeat: no-repeat;
  background-image: url("../img/note.svg");
  background-position: .5em .5em;
  border: 1px solid;
}
```

```
table.note.note_other { background-image: none; }
table.warning { background-image: url("../img/warning.svg"); }
table.caution { background-image: url("../img/caution.svg"); }
table.trouble { background-image: url("../img/troubleshooting.svg"); }
table.important { background-image: url("../img/important.svg"); }
table.attention { background-image: url("../img/attention.svg"); }
table.notice { background-image: url("../img/notice.svg"); }
table.remember { background-image: url("../img/remember.svg"); }
table.fastpath { background-image: url("../img/fastpath.svg"); }
table.restriction { background-image: url("../img/restriction.svg"); }
table.danger { background-image: url("../img/danger.svg"); }
table.tip { background-image: url("../img/tip.svg"); }

table.note {
  background-color: rgba(0, 120, 160, 0.09);
  border-color: #0078A0;
}
table.note_danger,
table.note_caution {
  background-color: rgba(255, 202, 45, 0.1);
  border-color: #606060;
}
table.note_warning,
table.note_attention,
table.note_important {
  background-color: rgba(255, 202, 45, 0.1);
  border-color: #FFCA2D;
}
table.note_restriction {
  background-color: rgba(255, 226, 225, 0.32);
  border-color: #FF342D;
}
```

7. Open the *template descriptor file (on page 1459)* 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>
```

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 Create a Custom Code Block Highlighter

You may want to add additional highlighters in your `<codeblock>` elements (for example, to highlight method names or arguments). 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 1464)*.
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 a custom template matching the codeblock for a given language (based on the `@outputclass`):

```
<?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="3.0">

  <xsl:template match="*[contains(@class, 'pr-d/codeblock')]
    [@outputclass='language-python']/text()">
    <xsl:analyze-string select="." regex="\(([\w,\s]+)\):">
      <xsl:matching-substring>
        <xsl:text>(</xsl:text>
        <span>
          <xsl:attribute name="class" select="'hl-arguments'"/>
          <xsl:value-of select="regex-group(1)"/>
        </span>
        <xsl:text>):</xsl:text>
      </xsl:matching-substring>
      <xsl:non-matching-substring>
        <xsl:next-match/>
      </xsl:non-matching-substring>
    </xsl:analyze-string>
  </xsl:template>

</xsl:stylesheet>
```

5. Open the *template descriptor file (on page 1459)* 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>
```

6. Create a `css` folder in the publishing template directory. In this directory, save a custom CSS file with rules that style the highlight span. For example:

```
.hl-arguments {
  color: orange;
}
```

7. Open the *template descriptor file (on page 1459)* 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>
```

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 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`.

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:

```
<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 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="3.0">

    <xsl:template match="*[contains(@class, ' pr-d/codeblock ')]">
       <div class='zebra'>
          <xsl:analyze-string regex="\n" select=".">
            <xsl:matching-substring/>
            <xsl:non-matching-substring>
               <div><xsl:value-of select="."/></div>
            </xsl:non-matching-substring>
          </xsl:analyze-string>
       </div>
```

```
    </xsl:template>
</xsl:stylesheet>
```

4. Use the **Integrate/Install DITA-OT Plugins** transformation scenario *(on page 1096)* 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:

```
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:

```
<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:

```
<?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="3.0">

    <xsl:template match="*[contains(@class, ' topic/related-links ')]">
        <!-- Remove. -->
```

```
    </xsl:template>
</xsl:stylesheet>
```

4. Use the **Integrate/Install DITA-OT Plugins** transformation scenario *(on page 1096)* 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:

```
<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:

```
<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 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="3.0">
  <xsl:param name="custom-param"/>

  <xsl:template match="*[contains(@class, ' topic/div ')]">
    <div>
      <xsl:call-template name="commonattributes"/>
      <xsl:call-template name="setid"/>
      <xsl:if test="$custom-param">
```

```
        <xsl:attribute name="custom" select="$custom-param"/>
      </xsl:if>
      <xsl:apply-templates/>
    </div>
  </xsl:template>
</xsl:stylesheet>
```

5. Use the **Integrate/Install DITA-OT Plugins** transformation scenario *(on page 1096)* 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 1665)*

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:

```
<feature extension="com.oxygenxml.pdf.css.init" file="init.xml"/>
```

2. Create a file named **init.xml** with the following ANT content:

```
<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.

Related Information:
How to Use XSLT Extension Points for PDF Output from a DITA-OT Plugin *(on page 1697)*

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/dev/topics/globalization-languages.

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 1469)* 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.

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 three locations:

- *DITA-OT-DIR*`/plugins/org.dita.base/xsl/common`
- *DITA-OT-DIR*`/plugins/org.dita.pdf2/cfg/common/vars`
- *DITA-OT-DIR*`/plugins/com.oxygenxml.pdf.css/resources/localization`

Related information
Webinar: Transforming DITA documents to PDF using CSS, Part 4 – Advanced CSS Rules

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:

```
*[class ~= "toc/title"][empty]:before {
  content: "Agenda";
}
```

```
/* 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 1471)*

How to Modify Existing Strings

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:

```
<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:

```
<langlist>
  <lang xml:lang="en"     filename="strings-en-us.xml"/>
  <lang xml:lang="en-us"  filename="strings-en-us.xml"/>
</langlist>
```

4. Create an empty `strings-en-us.xml` file and copy the strings you want to change from the default files into it. Make sure you leave the `@id` attribute unchanged because this is the key used to look up the string. Then replace their values:

```
<variables>
  <variable id="Figure">Fig</variable>
  <variable id="Table">Array</variable>
</variables>
```

5. Use the **Integrate/Install DITA-OT Plugins** transformation scenario *(on page 1096)* 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:

```
<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:

```
<langlist>
  <lang xml:lang="pl"     filename="strings-pl-pl.xml"/>
  <lang xml:lang="pl-pl"  filename="strings-pl-pl.xml"/>
</langlist>
```

4. Copy the PDF strings file (*`DITA-OT-DIR`*`\plugins\org.dita.pdf2\cfg\common\vars\en.xml`) to your plugin directory and rename it as `strings-pl-pl.xml`.

5. Copy the common DITA-OT strings defined in the *DITA-OT-DIR*`\plugins\org.dita.base\xsl\common\strings-en-us.xml` file into the file created at step 4.
6. Copy the PDF-CSS-HTML5 strings defined in the *DITA-OT-DIR*`\plugins\com.oxygenxml.pdf.css\resources\localization\strings-en-us.xml` file into the file created in step 4. Translate the content of each `<variable>` element (make sure to leave the `@id` attribute unchanged).

```
<variables>
  <variable id="Figure">Polish word for 'Figure'</variable>
  <variable id="Table">Polish word for 'Table'</variable>
  ...
  <variable id="Table of Contents">Polish word for 'Table of Contents'</variable>
  ...
  <variable id="Continued">(ciąg dalszy)</variable>
</variables>
```

7. Use the **Integrate/Install DITA-OT Plugins** transformation scenario *(on page 1096)* found in the **DITA Map** section in the **Configure Transformation Scenario(s)** dialog box.
8. 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 1444)* section can be used for this purpose.

You can also sign your PDF files to ensure their authenticity and integrity.

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:

Parameter	Value	Description
pdf.security.owner.password	<OWNER PASSWORD>	People using this password will be able to open the document, with full permissions.
pdf.security.user.password	<USER PASSWORD>	People using this password will be able to open the document, but they will not be able to print.
pdf.security.restrict.print	yes	Restricts users from printing.

Parameter	Value	Description
pdf.security.restrict.copy	yes	Restricts users from copying content.

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.

How to Sign a PDF

The following Bouncy Castle libraries must be downloaded (in their last available version):

- bcpkix-jdk18on
- bcprov-jdk18on
- bcutil-jdk18on

1. Install an external **Oxygen PDF Chemistry** processor.

   Restriction:
   A separate installation is required to include the additional libraries.

2. Copy the Bouncy Castle libraries to the `[CHEMISTRY_INSTALL_DIR]/lib` directory.
3. Create a keystore file (if you do not already have one). See Oxygen PDF Chemistry User Guide: Signing for information about how to create one.
4. Set the keystore file as the value for the `chemistry.sign.keystore` parameter.
5. **Optional:** If the keystore uses an export password, set it as the `chemistry.sign.password` value.
6. **Optional:** If you need to add more information (such as your company name), you can set the other `chemistry.sign.*` *(on page 1448)* parameters.
7. Set the external **Oxygen PDF Chemistry** path as the value for `css.processor.path.chemistry`.
8. Run the transformation scenario.

Related information
Oxygen PDF Chemistry: Signing

Troubleshooting

There are cases when the PDF CSS-based processing fails when trying to publish DITA content to a PDF file. This topic lists some of the common problems and possible solutions.

Access Denied

Problem

The transformation fails due to `Access to file:/tmp/chemistry-temp/ denied` (commonly responsible for such failures).

Cause

This usually happens on Linux systems when the user executing the transformation does not have the necessary permissions for the `/tmp` directory.

Solution

If the user's access cannot be modified, you can run the scenario by setting the `chemistry.security.workspace` parameter to a different directory.

Damaged PDF File

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`.

Glyph Not Available in Font

Problem

A warning appears during the PDF transformation that is similar to this:

```
[CH] Glyph "?" (0x2192, arrowright) not available in font "Roboto-Bold".
```



Warning:
Even if the message is a warning, sometimes it can lead to a failed transformation. This usually occurs for unique special characters (different from letters or common characters).

Cause

Some fonts lack specific characters, such as CJK characters or Greek symbols commonly used in scientific publications. Those characters are replaced with the # symbol in the PDF output.

Solution

Specify one or more fallback font(s) in your customization CSS:

```
body{
  font-family: Roboto, "Segoe UI", "Arial Unicode MS";
}
```

Tip:

- It is possible to use a generic family name as fallback (like `serif`, `sans-serif` or `monospace`) to call upon the processor's default fallback fonts system.
- To determine the needed font, you can copy the text fragment from the DITA source document and paste it into any text editor (e.g. MS Word). Make sure there are no licensing restrictions on that particular font.

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.

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 not 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**.

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 **Menu > 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.

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.

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:

```
title:before{
   content: oxy_xpath('format-date( current-date(), "[M] [Y]", "en", (), ())');
}
```

or you can use a more complex XPath expression like this:

```
title:before{

            content: oxy_xpath("let $cm:= format-date(current-date(), '[MNn]') \

return  concat( \

if ($cm= 'January') then  'JAN' else \

if ($cm= 'February') then  'FEB' else \

if ($cm= 'March') then  'MAR' else \

if ($cm= 'April') then  'APR' else \

if ($cm= 'May') then  'MAY' else \

if ($cm= 'June') then  'JUNE' else \

if ($cm= 'July') then  'JUL' else \

if ($cm= 'August') then  'AUG' else \

if ($cm= 'September') then  'SEPT' else \

if ($cm= 'October') then  'OCT' else \

if ($cm= 'November') then  'NOV' else  '' \

, \

' ', \

format-date(current-date(), '[Y0001]') \

) ");

}
```

Make sure the entire expression is rendered blue in the CSS editor. Replace the capitalized month names with the translation in the desired language.

Highlights Span 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 (it is easiest to see them in **Text** mode). The intervals are defined as follows:

For highlights:

```
<?oxy_custom_start type="oxy_content_highlight" color="140,255,140"?>
<?oxy_custom_end?>
```

For comments:

```
<?oxy_comment_start author="dan" timestamp="20201102T092905+0200" comment="Test"?>
<?oxy_comment_end?>
```

For inserted text:

```
<?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.

Unexpected Page Break Before or After an Element

Problem

A page break occurs before or after an element that has `break-before` or `break-after` (`break-before` or `break-after`) property set to *avoid*. For example, after a topic/section title (set by default):

```
*[class ~= "topic/title"] {
  break-after: avoid;
}
```

Cause

An empty element (for example, `<p>` or `<shortdesc>`) is present before or after the element with the break set to avoid. The page-break actually occurs at this element level.

Solution

Either remove the empty element from the DITA source topic (preferable) or set the display to *none* using the following CSS rule:

```
*[class ~= "topic/shortdesc"]:empty {
  display: none;
}
```

Error When Processing Topics With Chunk and Copy-To Attribute

Problem

A topic marked with both the `@chunk` and `@copy-to` attributes is missing from the PDF output and the following error appears in the **Results** view:

```
[DOTX008E] File 'file:/D:/path/to/file.dita' does not exist or cannot be loaded.
```

Cause

The chunk processing is skipped by default and must be enabled.

Solution

Set the `enable.chunk.processing` parameter to the value of **true** and re-run the transformation scenario.

XSL FO-based DITA to PDF Customization

Oxygen XML Developer comes bundled with the DITA Open Toolkit that provides a mechanism for converting *DITA maps (on page 2304)* to PDF output. Oxygen XML Developer includes a built-in **DITA Map PDF - based on XSL-FO** transformation scenario *(on page 1090)* 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:
>
> - Sample Plugin: DITA-OT PDF Customization Plugin for Oxygen User Manual
> - Sample Plugin: DITA-OT PDF2 - Generate Numbers Before Topic's Title

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 1090)* 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 1714)* 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 1090)*, 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 for the **DITA Map PDF - based on XSL-FO** transformation scenario *(on page 1090)*.

1. Create a customization directory *(on page 1714)* (if you have not already done so).
2. Create a `common\artwork` directory structure in your customization directory and copy your logo in it (for example, `[C:\Customization\common\artwork\logo.png`).
3. Rename `Customization\catalog.xml.orig` to: `Customization\catalog.xml`.
4. Open the `catalog.xml` in Oxygen XML Developer and *uncomment* this line:

   ```
   <!--uri name="cfg:fo/xsl/custom.xsl" uri="fo/xsl/custom.xsl"/-->
   ```

   It now looks like this:

```
<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 Developer and create the template called `createFrontCoverContents` for DITA-OT 4.4.0.

   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 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>
        <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>

<!-- Load the image logo -->
 <fo:block text-align="center" width="100%">
  <fo:external-graphic
     src="url({concat($artworkPrefix,
                         'Customization/OpenTopic/common/artwork/logo.png')})"
   />
 </fo:block>
</xsl:template>

</xsl:stylesheet>
```

7. Edit the **DITA Map PDF - based on XSL-FO** transformation scenario *(on page 1090)*, 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 Documentation - Customizing PDF Output.

Related Information:
Using a Customization Directory *(on page 1714)*

Customizing the Header and Footer in PDF Output

This procedure should only be used for the **DITA Map PDF - based on XSL-FO** transformation scenario *(on page 1090)*.

The XSLT stylesheet *`DITA-OT-DIR`*`/plugins/org.dita.pdf2/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 1714)* (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 1090)*, go to the **Parameters** tab, and set the **customization.dir** parameter to point to the location of your customization directory.

Related Information:
https://github.com/oxygenxml/com.oxygenxml.pdf2.ug/wiki
Using a Customization Directory *(on page 1714)*

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 1090)*, follow this procedure:

1. Create a customization directory *(on page 1714)* (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 Developer and *uncomment* this line:

   ```
   <!--uri name="cfg:fo/xsl/custom.xsl" uri="fo/xsl/custom.xsl"/-->
   ```

   The uncommented line should look like this:

   ```
   <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 Developer 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:

```
<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>


</xsl:template>
```

- 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">
              <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>

   <!--<xsl:call-template name="createPreface"/>-->

      </fo:flow>
        </fo:page-sequence>
        <xsl:if test="not($retain-bookmap-order)">
            <xsl:call-template name="createNotices"/>
```

```
        </xsl:if>
    </xsl:template>
```

7. Edit the **DITA Map PDF - based on XSL-FO** transformation scenario *(on page 1090)*, go to the **Parameters** tab, and set the **customization.dir** parameter to point to the location of your customization directory.

Adding an Edit Link in PDF Output to Launch Oxygen XML Web Author

You can embed *Edit* links in the **DITA Map PDF - based on XSL-FO** transformation scenario *(on page 1090)* 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 1090)* 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 or Content Fusion 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

The following procedure works for the **DITA Map PDF - based on XSL-FO** transformation scenario *(on page 1090)*.

Suppose that in your DITA content you have two *block elements (on page 2303)*, such as two paragraphs:

```
<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 2307)* 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:

```
<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:

```
<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>
```

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 **DITA Map PDF - based on XSL-FO** transformation scenario *(on page 1090)* 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 2304)* is transformed using the **DITA Map PDF - based on XSL-FO** transformation scenario *(on page 1090)* 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 1176)*.

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:

```
<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:

```
<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:

```
<logical-font name="Monospaced">
      <physical-font char-set="default">
        <font-face>Courier New, Courier</font-face>
```

```
    </physical-font>
.............
</logical-font>
```

Important:

If no alias mapping is found for a font-family specified in the XSLT stylesheets, the processing defaults to **Helvetica**.

Related information

http://www.elovirta.com/2016/02/18/font-configuration-in-pdf2.html

Adding Libraries to the Built-in FO Processor (DITA-OT)

Adding Support for PDF Images

Starting with Oxygen XML Developer version 20.0, PDF image support is enabled by default in the built-in **DITA Map PDF - based on XSL-FO** transformation scenario *(on page 1090)*. For older versions of Oxygen XML Developer, use the following procedures to enable this support:

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.

Adding Hyphenation Support for DITA-OT Transformation Scenarios

1. Download the pre-compiled *JAR (on page 2306)* 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.

Debugging DITA PDF Transformations

To debug the **DITA Map PDF - based on XSL-FO** transformation scenario *(on page 1090)*, follow these steps:

1. Open the **Preferences** dialog box **(Options > Preferences)** *(on page 79)*, go to **XML > XML Catalog**, click **Add**, and select the file located at *DITA-OT-DIR*`\plugins\org.dita.pdf2\cfg\catalog.xml`. If there are other custom DITA-OT PDF customization plugins that contain XML catalogs, references to those XML catalogs might need to be added as well.

2. Open the map 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 481)* 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.fop\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 corresponding plugin folders.

   > **Note:**
   > For validation purposes, you need to add the main debugged stylesheet (usually `topic2fo_shell_fop.xsl`) to the Main Files folder *(on page 353)* 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:
   a. Set the `locale` parameter (e.g. with the value of `en_GB`).
   b. Set the `work.dir.url` parameter with the value `${cfdu}`.
   c. Set the `customizationDir.url` parameter to point to either your customization directory or to the default DITA-OT customization directory. Its value should have a URL syntax like this:

   ```
   file://c:/path/to/<term keyref="glossentry_dita_ot_dir"/>/plugins/org.dita.pdf2/cfg
   ```

9. You need to reference the extra commonly used JAR libraries by clicking the **Extensions** button and using wildcards to add a reference to all libraries in this folder: `DITA-OT\plugins\com.oxygenxml.common\lib\*`.
10. Apply the transformation to continue the debugging process.

    > **Note:**
    > For externally configured DITA Open Toolkit installations or when using custom plugins based on the base PDF2 plugin, the paths to resources described above may need to be adjusted accordingly.

Related information

Debugging XSLT Stylesheets and XQuery Documents *(on page 1826)*

How to Enable Debugging for FO Processor Transformations *(on page 1177)*

DocBook to PDF Output Customization

When the default layout and output of the DocBook to PDF transformation needs to be customized, follow these steps:

1. Create a custom version of the DocBook title spec file.

   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:

   ```
   <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 Developer and set the customization layer in the **XSL URL** property of the scenario *(on page 1102)*.

Related Information:

The book **DocBook XSL: The Complete Guide** by Bob Stayton contains more details about customizing the PDF output.

Video demonstration for creating a DocBook customization layer in Oxygen XML Developer.

12.

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.



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.

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.

Related Information:

Content Completion in XPath Expressions *(on page 634)*
Find/Replace in Multiple Files *(on page 368)*
Find/Replace Dialog Box *(on page 364)*

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 Developer provides an XPath toolbar to let you query XML documents fast and easy using XPath expressions.

Figure 415. 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 Developer 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 Developer can return a compilation error.

XPath scope menu

Oxygen XML Developer 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.
- **Opened archive** - Files that are opened in the **Archive Browser** view *(on page 1735)*.
- **Working sets** - The selected *working sets (on page 2309)*.

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 2309)*.
- **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 397)* 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 1729)*.

> **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 1729)*.

Related Information:
XPath Expression Results View *(on page 1732)*

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 Developer 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 1727)*.

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 Developer 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 Developer 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 474)* view.

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 Developer 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 397)* 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 Developer allows you to define a scope for the XPath operation to be executed. You can choose where the XPath expression will be executed:

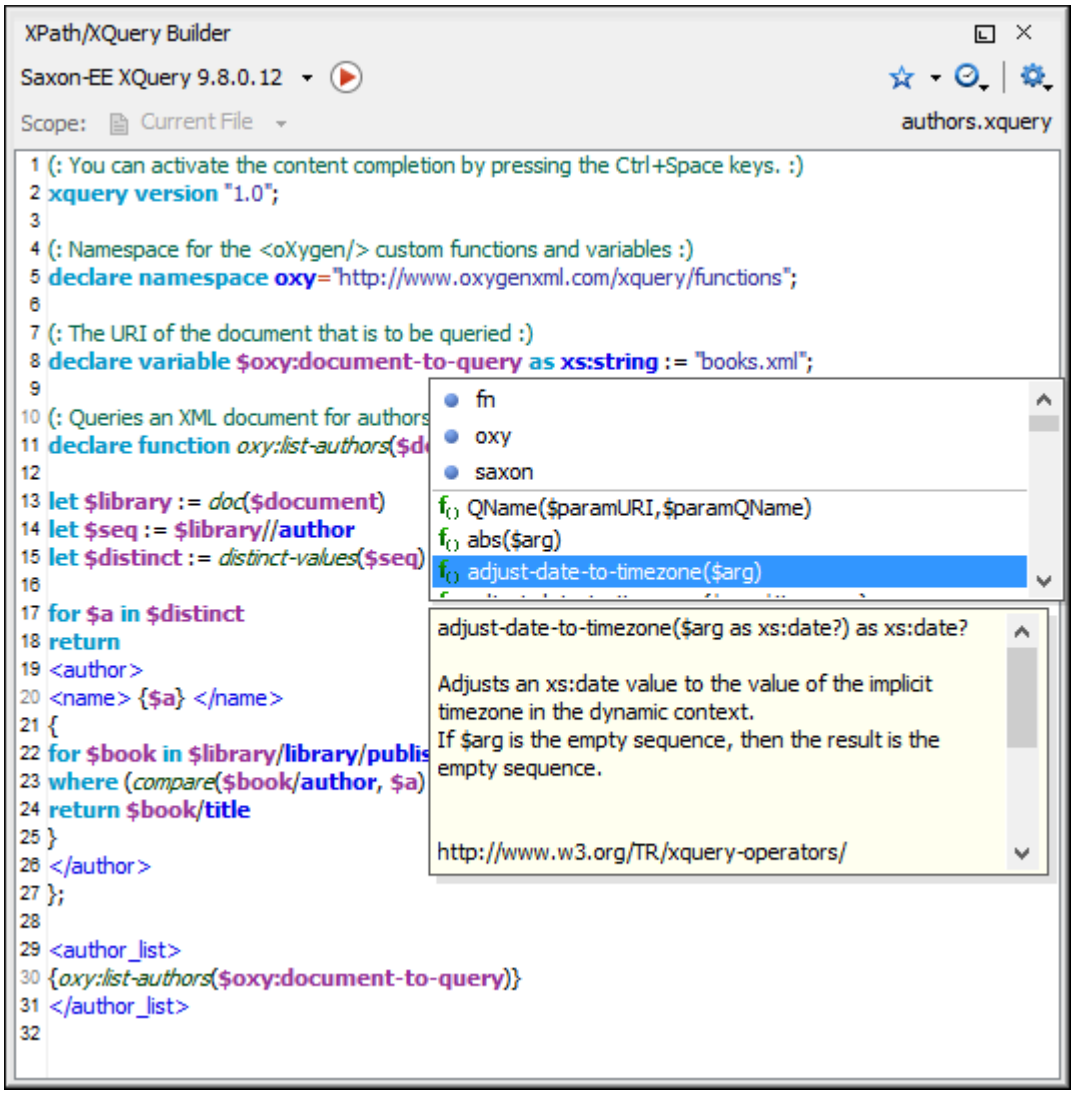
- **Current file** - Currently selected file only.
- **Project** - All the files in the project.
- **Selected project resources** - The files selected in the project.
- **All opened files** - All files that are opened in the application.

- **Opened archive** - Files that are opened in the **Archive Browser** view *(on page 1735)*.
- **Working sets** - The selected *working sets (on page 2309)*.

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 2309)*.
- **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 416. XPath/XQuery Builder View



While you edit an XPath or XQuery expression, Oxygen XML Developer assists you with the following features:

- *Content Completion Assistant (on page 2303)* - 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 79)* and go to **Editor > Syntax Highlight** *(on page 158)*.
- 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.

Related Information:
XPath Expression Results View *(on page 1732)*

XPath Expression Results View

When you run an XPath expression, Oxygen XML Developer displays the results of its execution in the **Results** view.

This view contains the following columns:

- **Description** - The result thatOxygen XML Developer displays when you run an XPath expression.
- **XPath location** - The path to the matched node.
- **Resource** - The name of the document that you run the XPath expression on.
- **System ID** - The path to the document itself.
- **Location** - The location of the result in the document.

To arrange the results depending on a column, click its header. To group the results by their resource, or by their system ID, right-click the header of any column in the **Results** view and select **Group by "Resource"** or **Group by "System ID"**. If no information regarding location is available, Oxygen XML Developer displays **Not available** in the **Location** column. Oxygen XML Developer displays the results in a valid XPath expression format.

```
- /node[value]/node[value]/node[value] -
```

The **Results** view also includes various toolbar and contextual menu actions. For more information, see Results View *(on page 474)*.

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.

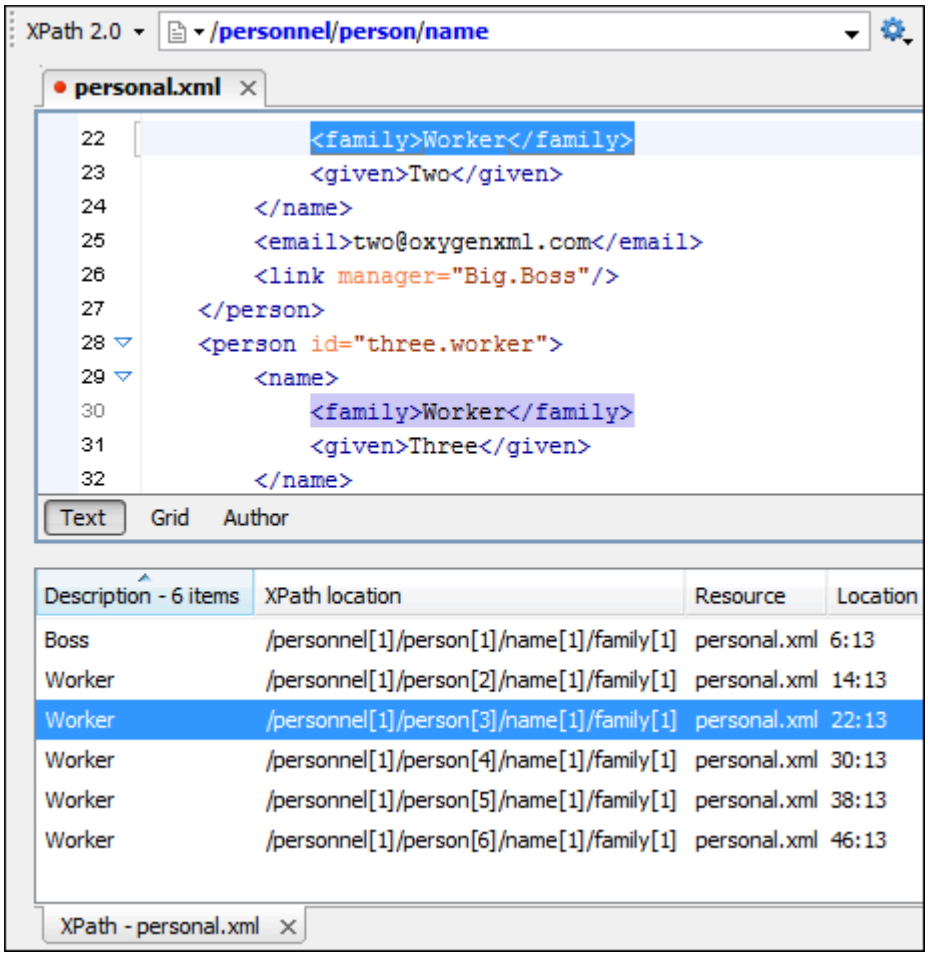
To find all `example` nodes contained in the `sect2` nodes of a DocBook XML document, use the following XPath expression: `//chapter/sect1/sect2/example`. Oxygen XML Developer adds a result in the **Results View** for each `example` node found in any `sect2` node.

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.

Figure 417. XPath Results Highlighted in Editor Panel with Character Precision



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 2309)*. These catalogs are configured in the **XML Catalog** preferences *(on page 168)* pages and the **XML Parser** preferences *(on page 171)*.

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:

```
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 1727)*) and namespace URIs go to the **XPath** preferences page *(on page 190)* 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 Developer 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 Developer includes a useful **Archive Browser** view *(on page 1735)* 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 2306)*
- Office Open XML (OOXML) files
- Open Document Format (ODF) files
- *IDML files (on page 2306)*

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.

Resources

For more information about working with an EPUB archive in Oxygen XML Developer, watch our video demonstration:

https://www.youtube.com/embed/OIGTNQwOCl8

Browsing Archives

Oxygen XML Developer 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.

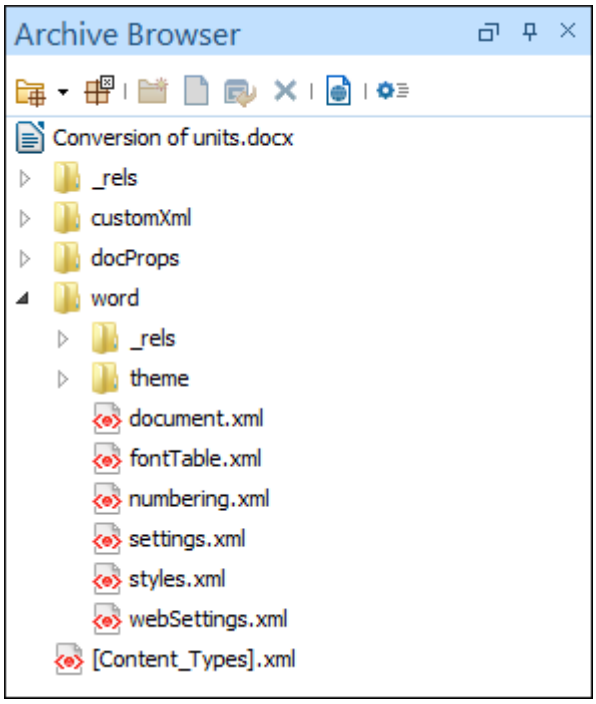
To open an archive in the **Archive Browser** view, use one of the following methods:

- Open an archive from the **Project** view *(on page 334)*.
- Select an archive in one of the file chooser dialog boxes in Oxygen XML Developer (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 Developer as a supported archive type, you can add it in the **Archive** preferences page *(on page 219)*.

Figure 418. 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 219)* 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 334)* 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 219)*.

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 file** 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 368)* 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 397)* 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.

Resources

For more information, watch our video demonstration about working with an EPUB in the **Archive Browser** view:

https://www.youtube.com/embed/OIGTNQwOCI8

Working with Archive Files

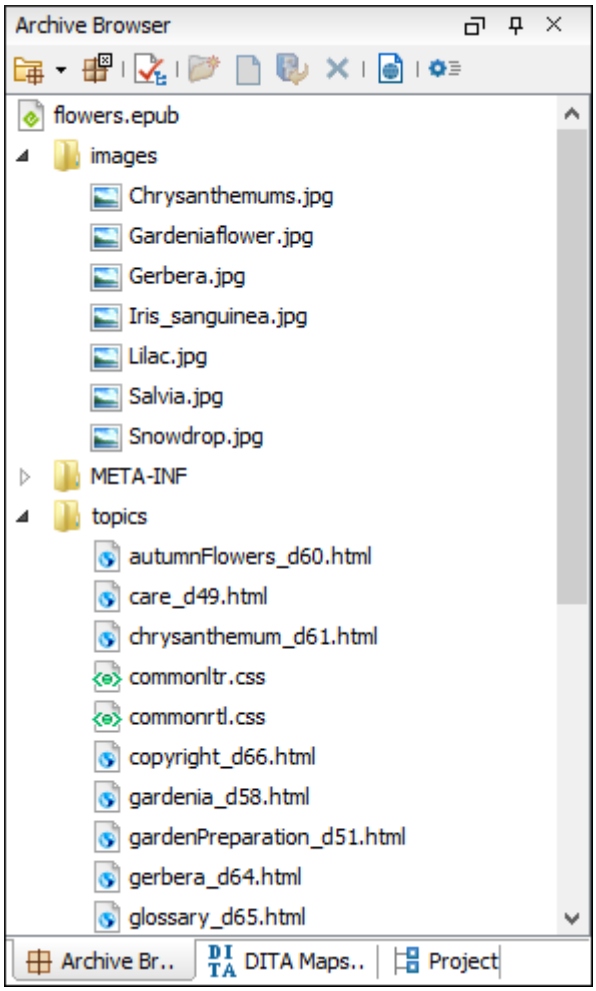
Oxygen XML Developer includes support for working with various types of archives, including the following:

- **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 1735)*, their internal components are expanded:

- Document content (XHTML and image files).
- Packaging files.
- Container files.

Figure 419. EPUB File Displayed in the Archive Browser View



When an archive is expanded in the **Archive Browser** view *(on page 1735)*, 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 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 1735)* includes a **Validate** action on the toolbar that checks the EPUB archive to make sure the structure and content are valid. Oxygen XML Developer 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.

Resources

For more information about working with an EPUB archive in Oxygen XML Developer, watch our video demonstration:

https://www.youtube.com/embed/OIGTNQwOCl8

Related information
The Archive Browser View *(on page 1735)*
EPUB Document Type (Framework) *(on page 1064)*

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 1735)*.

   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 1735)* 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 235)*.

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 1735)*.
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 Developer 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.

14.
Databases and SharePoint

Oxygen XML Developer provides support for connecting and integrating with various databases and Microsoft SharePoint. This section includes information about the database-related features in Oxygen XML Developer. 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 Developer 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 Developer 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 Developer include:

- IBM DB2 (Deprecated) *(on page 1784)*
- Microsoft SQL Server (Deprecated) *(on page 1748)*
- Oracle Database (Deprecated) *(on page 1752)*
- PostgreSQL (Deprecated) *(on page 1757)*
- eXist *(on page 1761)*
- MarkLogic (Deprecated) *(on page 1766)*
- MySQL (Deprecated) *(on page 1775)*
- Generic JDBC *(on page 1777)*
- JDBC-ODBC *(on page 1778)*
- BaseX *(on page 1780)*
- WebDAV *(on page 1789)*
- Microsoft SharePoint *(on page 1802)*

Related information

Integration with Microsoft SharePoint *(on page 1802)*

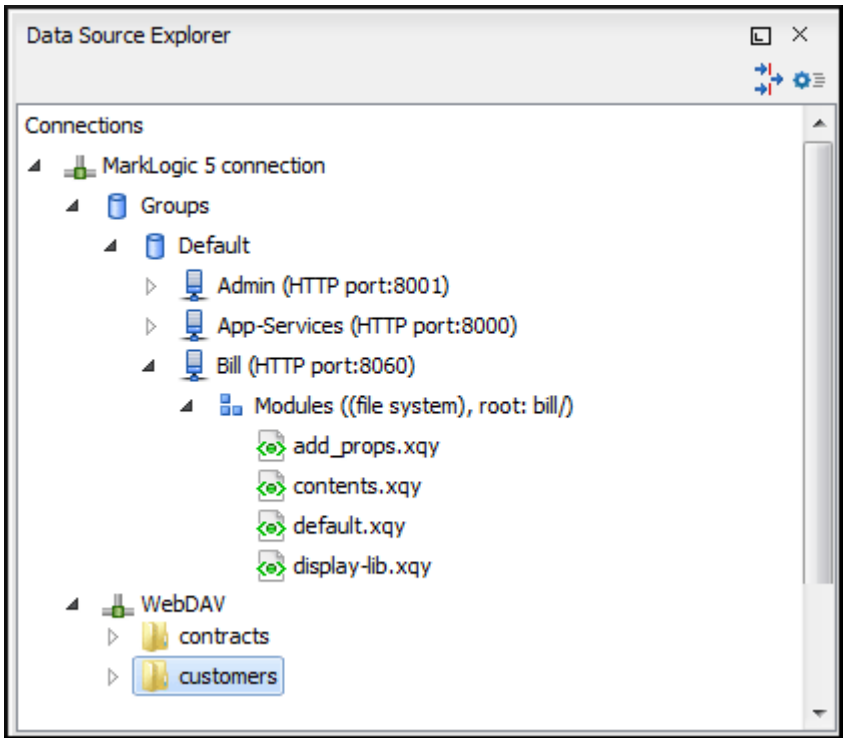
Data Source Explorer View

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 Developer supports multiple simultaneous database connections and the connection tree in the **Data Source Explorer** view provides an easy method for browsing them.

Figure 420. 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 334)*, 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 209)*, 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 205)* 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.

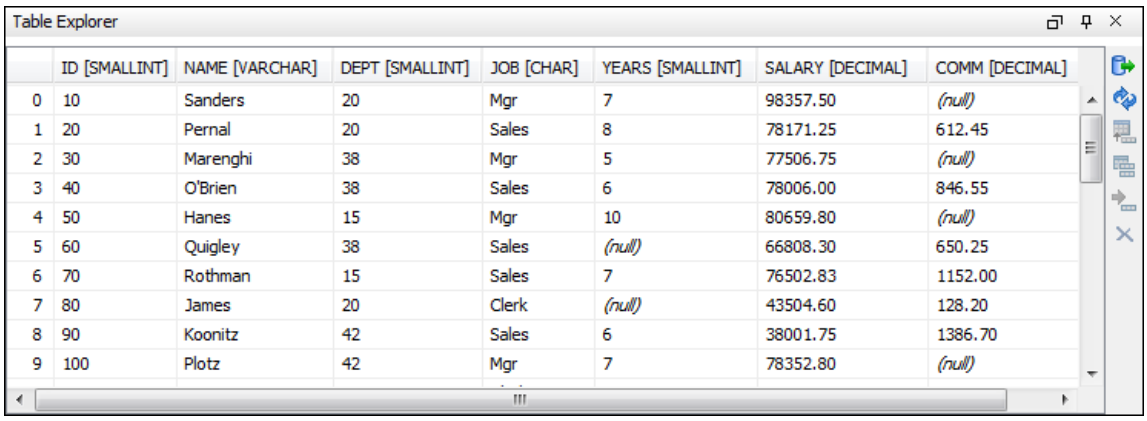
Related Information:

Data Sources Preferences *(on page 205)*

Table Explorer View

Relational databases tables in the **Data Source Explorer** view *(on page 1742)* 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 Developer attempts to update the database with the new cell content.

Figure 421. Table Explorer View



	ID [SMALLINT]	NAME [VARCHAR]	DEPT [SMALLINT]	JOB [CHAR]	YEARS [SMALLINT]	SALARY [DECIMAL]	COMM [DECIMAL]
0	10	Sanders	20	Mgr	7	98357.50	*(null)*
1	20	Pernal	20	Sales	8	78171.25	612.45
2	30	Marenghi	38	Mgr	5	77506.75	*(null)*
3	40	O'Brien	38	Sales	6	78006.00	846.55
4	50	Hanes	15	Mgr	10	80659.80	*(null)*
5	60	Quigley	38	Sales	*(null)*	66808.30	650.25
6	70	Rothman	15	Sales	7	76502.83	1152.00
7	80	James	20	Clerk	*(null)*	43504.60	128.20
8	90	Koonitz	42	Sales	6	38001.75	1386.70
9	100	Plotz	42	Mgr	7	78352.80	*(null)*

You can sort the content of a table by one of its columns by clicking its column header.

Note the following:

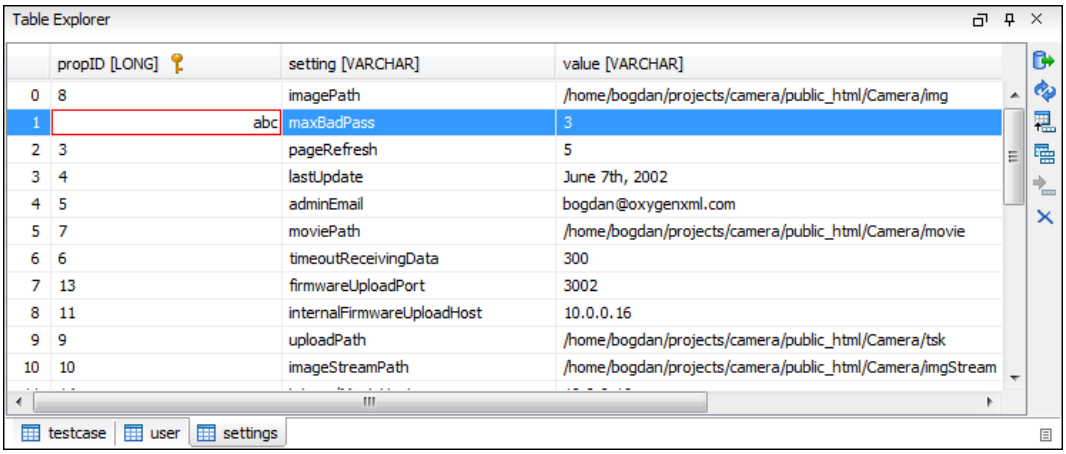
- The first column is an index (not part of the table structure).
- Every column header contains the field name and its data type.
- The primary key columns are marked with this symbol: .
- Multiple tables are presented in a tabbed manner.

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 209)*). 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 422. 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 423. Duplicate Entry for Primary Key

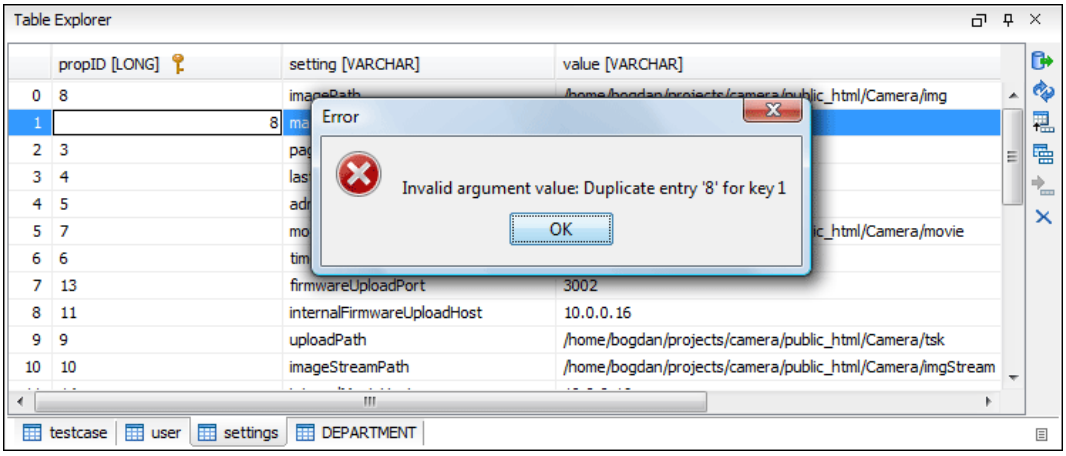


Table Explorer Contextual Menu Actions

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**.

Insert row

Inserts an empty row in the table.

Duplicate row

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.

Commit row

Commits the selected row.

Delete row

Deletes the selected row.

Copy

Copies the content of the cell.

Paste

Pastes copied content into the selected cell.

Table Explorer Toolbar Actions

The toolbar of the **Table Explorer** view also includes the following actions:

Export to XML

Opens the **Export Criteria** dialog box (a thorough description of this dialog box can be found in the Import from database *(on page 1820)* chapter) .

Refresh

Performs a refresh for the sub-tree of the selected node.

Insert row

Inserts an empty row in the table.

Duplicate row

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.

Commit row

Commits the selected row.

Delete row

Deletes the selected row.

Related Information:

Data Source Explorer View *(on page 1742)*

Database Connection Support

Oxygen XML Developer 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 205)* and once configured, the database connections can be viewed and managed in the **Data Source Explorer** view *(on page 1742)*. Oxygen XML Developer also includes a **Database** *perspective (on page 276)* that helps you to manage databases.

The database support in Oxygen XML Developer offers a variety of capabilities, including:

- Browsing the structure of databases in the **Data Source Explorer** view *(on page 1742)*.
- Viewing relational tables in the **Table Explorer** view *(on page 1744)*.
- 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 Developer offers support for the most commonly used relational databases, including:

- IBM DB2 (Deprecated)
- Oracle 11g (Deprecated)
- Microsoft SQL Server (Deprecated)
- PostgreSQL (Deprecated)
- MySQL (Deprecated)

Oxygen XML Developer 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 Developer offers support for the most commonly used native XML databases, including:

- eXist
- MarkLogic (Deprecated)
- Oracle XML DB (Deprecated)
- Base X

Related information

WebDAV Connections *(on page 1789)*

Integration with Microsoft SharePoint *(on page 1802)*

Microsoft SQL Server Database Connections (Deprecated)

Oxygen XML Developer includes support for Microsoft SQL Server database connections. Oxygen XML Developer allows you to browse the structure of a SQL Server database in the **Data Source Explorer** view *(on page 1742)*, open tables in the **Table Explorer** view *(on page 1744)*, 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:

1. Download the appropriate MS SQL JDBC driver from the Microsoft website: https://docs.microsoft.com/en-us/sql/connect/jdbc/release-notes-for-the-jdbc-driver?view=sql-server-ver16#102.
2. Configure MS SQL Server *Data Source* drivers *(on page 1748)*.
3. Configure a MS SQL Server *Connection (on page 1749)*.
4. To view your connection, go to the **Data Source Explorer** view *(on page 1742)* (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 2307)*.

How to Configure Microsoft SQL Server Data Source Drivers

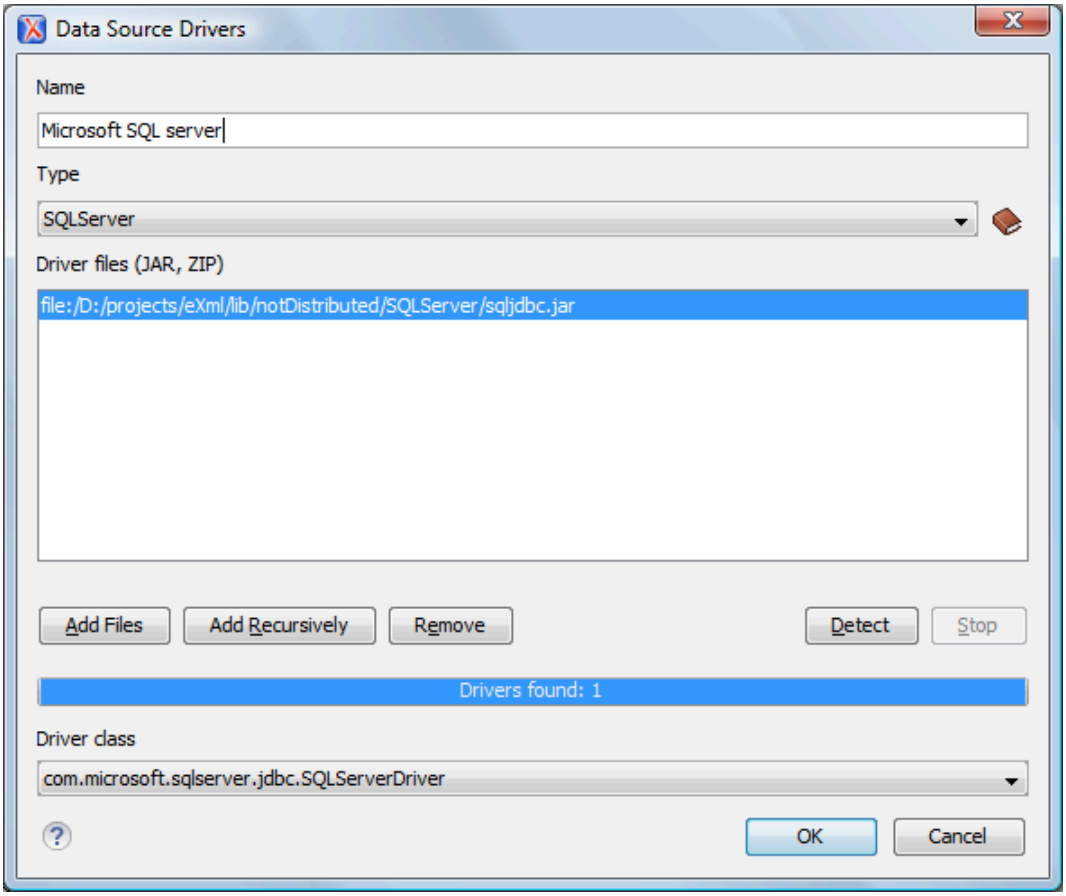
Note:
Available in the Enterprise edition only.

To configure a data source for connecting to a Microsoft SQL server, follow these steps:

1. Download the appropriate MS SQL JDBC driver from the Microsoft website: https://docs.microsoft.com/en-us/sql/connect/jdbc/release-notes-for-the-jdbc-driver?view=sql-server-ver16#102.
2. Open the **Preferences** dialog box **(Options > Preferences)** *(on page 79)* 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 424. 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 1749)*.

How to Configure a Microsoft SQL Server Connection

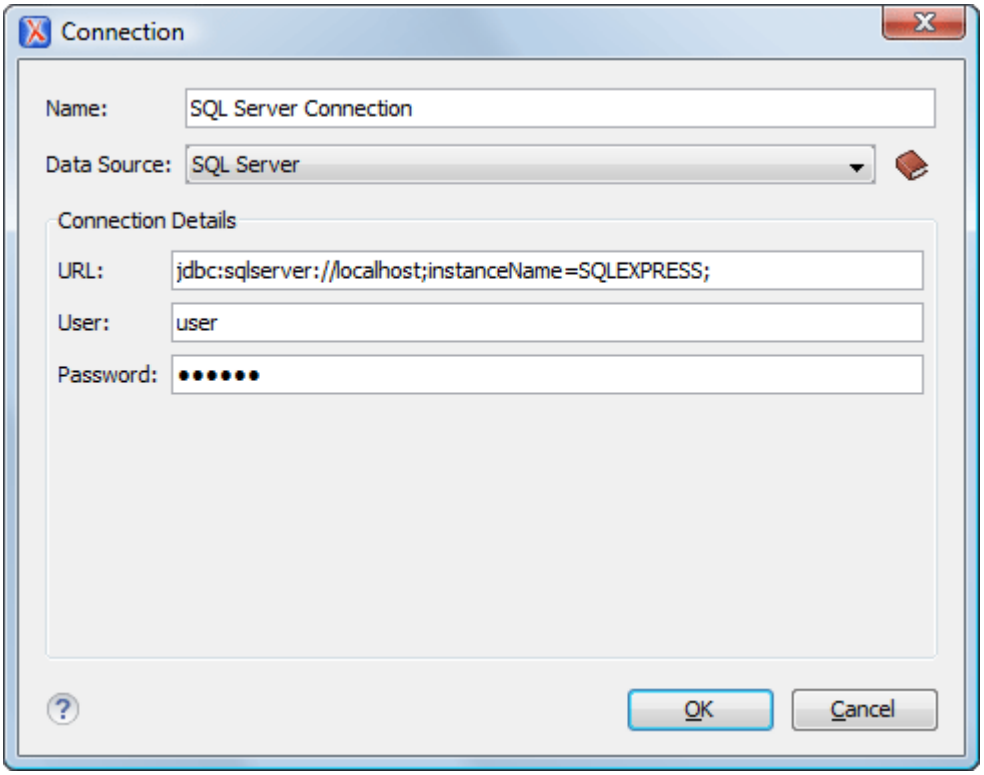
Note:
The support to configure a Microsoft SQL Server connection is available in the Enterprise edition only.

To configure a connection to a Microsoft SQL Server, follow these steps:

1. Open the **Preferences** dialog box **(Options > Preferences)** *(on page 79)* 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 425. Connection Configuration Dialog Box



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 1742)* (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 2307)*.

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 1742)*, 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 205)* 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 1744)*.

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 1820)* chapter).

Database-Specific Contextual Menu Actions

In addition to the general contextual menu actions in the **Data Source Explorer** view *(on page 1742)*, 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.

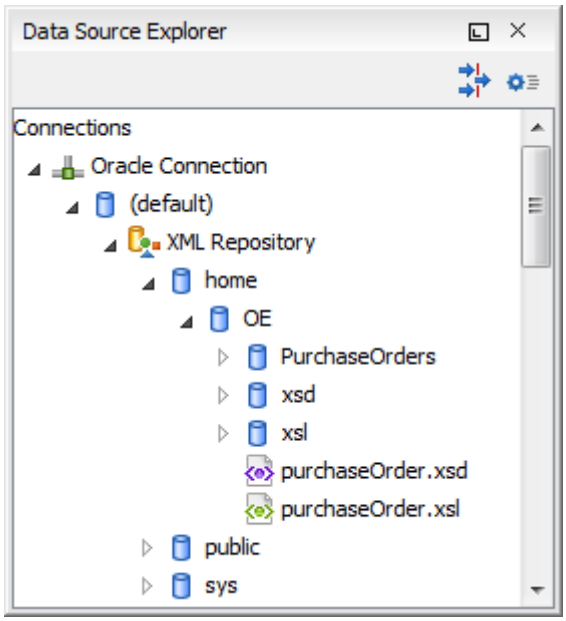
View

Opens the selected schema in Oxygen XML Developer.

Oracle Database Connections (Deprecated)

The Oracle database is a common relational type of database system. Oxygen XML Developer 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 Developer allows you to browse Oracle repositories in the **Data Source Explorer** view *(on page 1742)*, open tables in the **Table Explorer** view *(on page 1744)*, and perform various operations on the resources in the repository.

Figure 426. Oracle Database Connection



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:

1. Go to http://www.oracle.com/technetwork/database/enterprise-edition/jdbc-112010-090769.html and download the Oracle 11g JDBC driver called `ojdbc6.jar`.
2. Configure Oracle 11g *Data Source* drivers *(on page 1753)*.

3. Configure an Oracle 11g *Connection (on page 1754)*.
4. To view your connection, go to the **Data Source Explorer** view *(on page 1742)* (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 2307)*.

How to Configure Oracle 11g Data Source Drivers

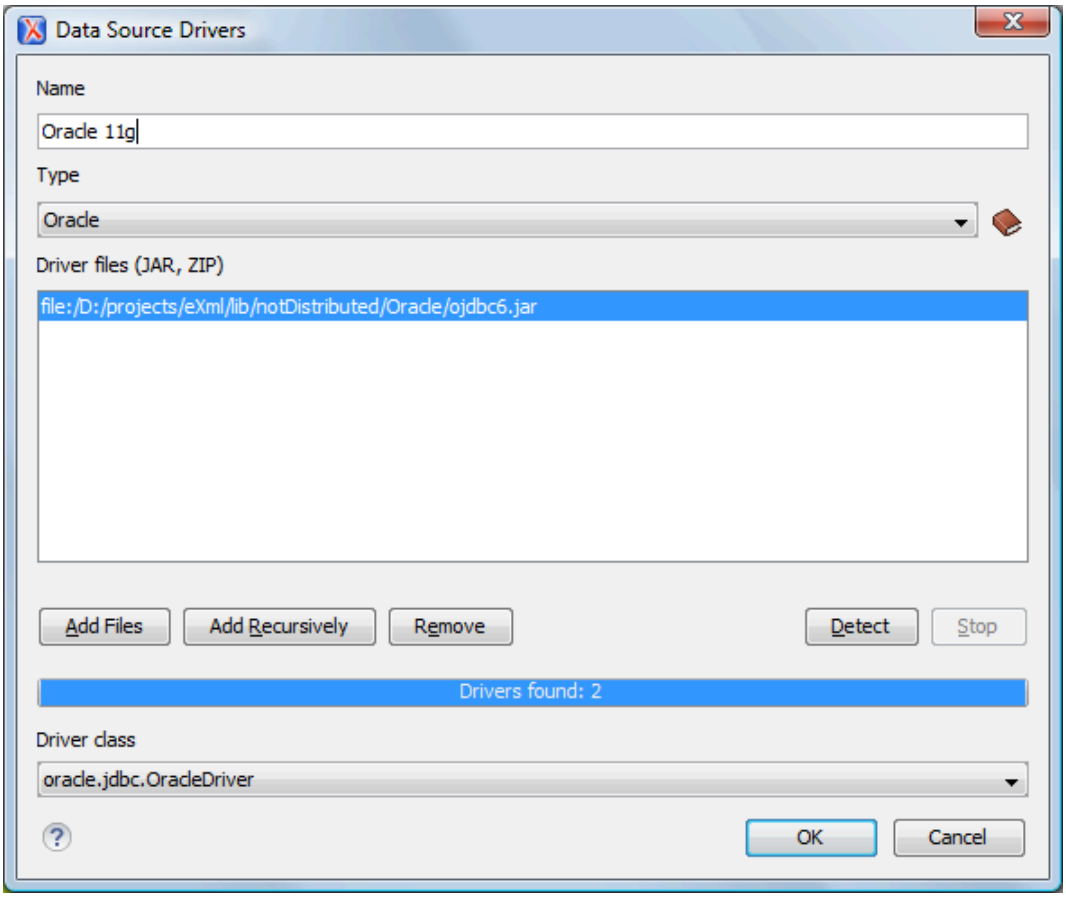
Note:
Available in the Enterprise edition only.

To configure a data source for connecting to an Oracle 11g server, follow these steps:

1. Go to http://www.oracle.com/technetwork/database/enterprise-edition/jdbc-112010-090769.html and download the Oracle 11g JDBC driver called `ojdbc6.jar`.
2. Open the **Preferences** dialog box **(Options > Preferences)** *(on page 79)* 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 427. 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 1754)*.

How to Configure an Oracle 11g Connection

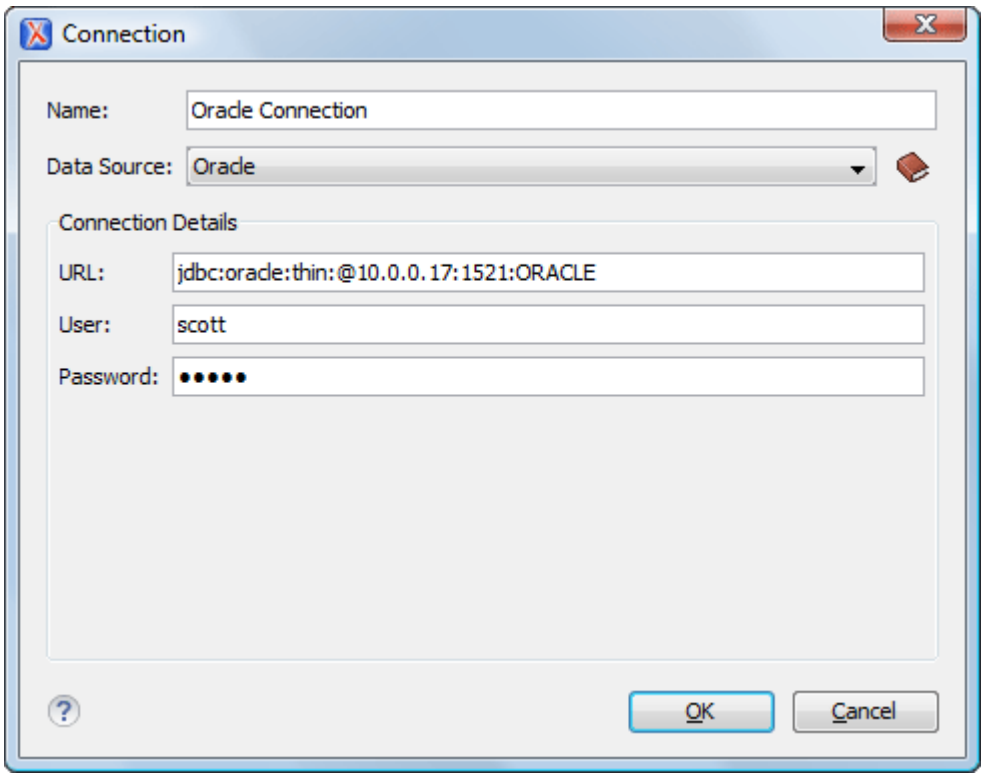
Note:
Available in the Enterprise edition only.

To configure a connection to an Oracle 11g server, follow these steps:

1. Open the **Preferences** dialog box **(Options > Preferences)** *(on page 79)* 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 428. Connection Configuration Dialog Box



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 1742)* (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 2307)*.

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 1742)*, 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 205)* 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 1744)*.

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 1820)* chapter).

Database-Specific Contextual Menu Actions

In addition to the general contextual menu actions in the **Data Source Explorer** view *(on page 1742)*, 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 Developer 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 Developer 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

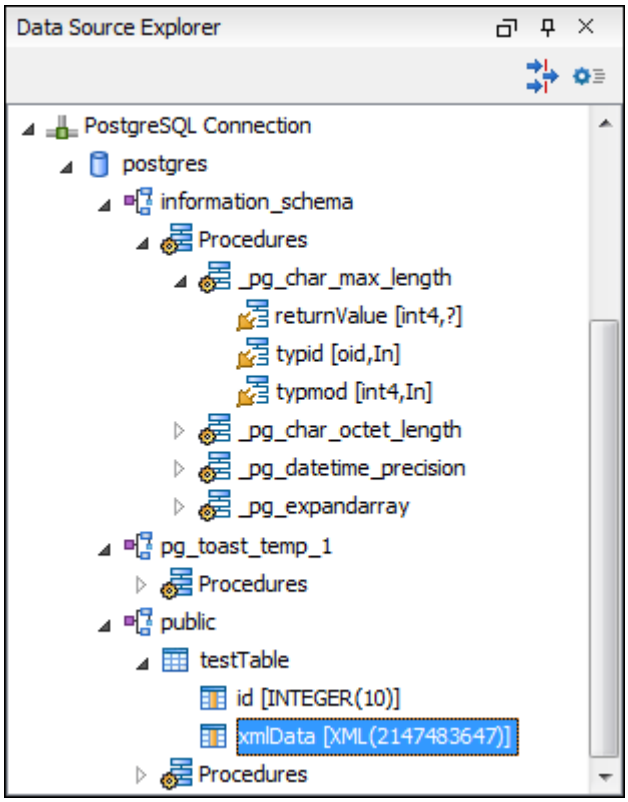
Compares two selected resources using the **Compare Files** tool *(on page 402)*.

PostgreSQL Database Connections (Deprecated)

Oxygen XML Developer includes support for PostgreSQL database connections. Oxygen XML Developer allows you to browse the structure of a PostgreSQL database in the **Data Source Explorer** view *(on page*

1742), open tables in the **Table Explorer** view *(on page 1744)*, and perform various operations on the resources in the repository.

Figure 429. PostgreSQL Database Connection



Configuring a PostgreSQL Database Connection

To configure the support for a PostgreSQL database, follow this procedure:

1. Go to https://jdbc.postgresql.org/download/ and download the PostgreSQL JDBC driver specific for your server version.
2. Configure PostgreSQL *Data Source* drivers *(on page 1758)*.
3. Configure a PostgreSQL *Connection (on page 1759)*.
4. To view your connection, go to the **Data Source Explorer** view *(on page 1742)* (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 2307)*.

How to Configure PostgreSQL Data Source Drivers

To configure a data source for connecting to a PostgreSQL server, follow these steps:

1. Go to https://jdbc.postgresql.org/download/ and download the PostgreSQL JDBC3 driver specific for your server version.
2. Open the **Preferences** dialog box **(Options > Preferences)** *(on page 79)* 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 1759)*.

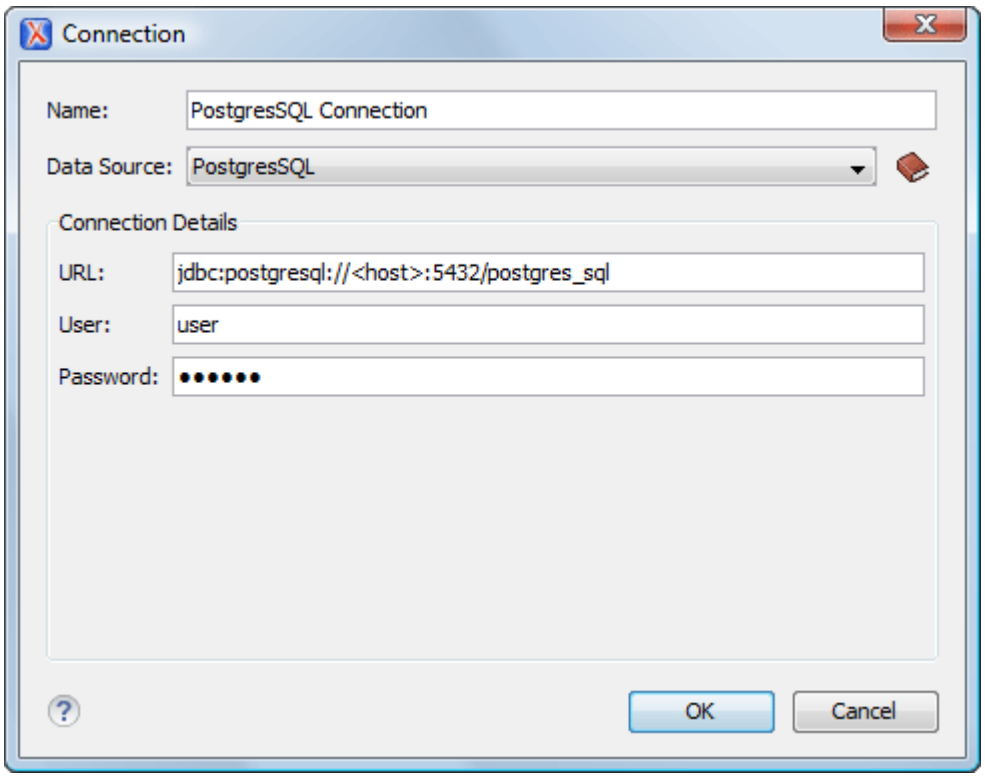
How to Configure a PostgreSQL Connection

To configure a connection to a PostgreSQL server, follow these steps:

1. Open the **Preferences** dialog box **(Options > Preferences)** *(on page 79)* 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 430. Connection Configuration Dialog Box



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 1742)* (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 2307)*.

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 1742)*, 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 205)* 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 1744)*.

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 1820)* chapter).

Database-Specific Contextual Menu Actions

In addition to the general contextual menu actions in the **Data Source Explorer** view *(on page 1742)*, the resource level nodes in PostgreSQL connections include the following additional contextual menu action:

Resource Level Nodes

Compare

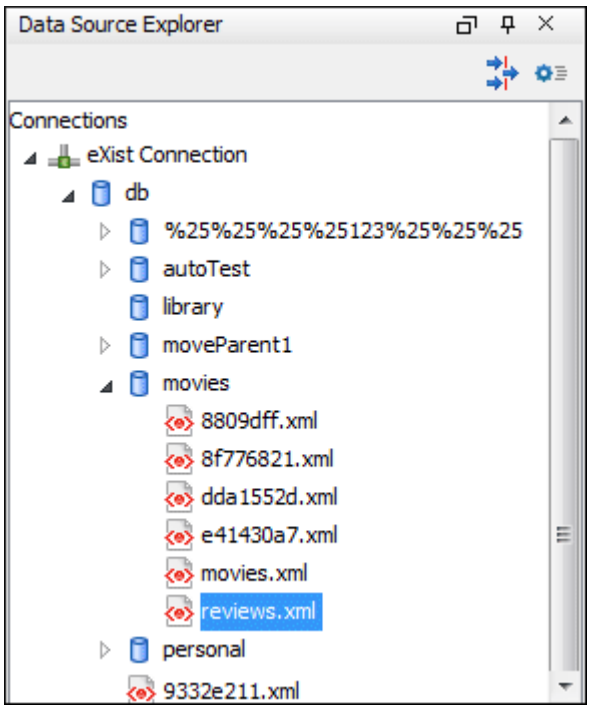
Compares two selected resources using the **Compare Files** tool *(on page 402)*.

eXist Database Connections

Attention:
Oxygen XML Developer 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 Developer includes support for *eXist* database connections. Oxygen XML Developer allows you to browse the structure of a *eXist* database in the **Data Source Explorer** view *(on page 1742)* and perform various operations on the resources in the repository.

Figure 431. eXist Database Connection



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 79)*, 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 Developer expects the `exist/webstart/exist.jnlp` path to be accessible at the provided **Host** and **Port**.

Attention:
The connection details dialog box has a **Libraries** path where it will download JAR libraries from the eXist server. If you are using Oxygen XML Developer version 25.0 or older with eXist version 6.2.0 or newer and the final connection to the server does not work, you need to remove all libraries with the pattern `log4j-*.jar` from the folder of downloaded libraries because they may interfere with the logging in Oxygen XML Developer.

3. To view your connection, go to the **Data Source Explorer** view *(on page 1742)* (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 2307)*.

Attention:
Oxygen XML Developer 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 Developer 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 1761)*.

Step 1: Configure eXist Data Source Drivers

Oxygen XML Developer 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 79)* 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 Developer.
    - The `exist.jar` file located in the base directory (if present, depending on the server version).
    - All JAR files in the `lib/core/` directory (if present) or all JAR files located in the `lib` directory except for the JAR libraries with the pattern `log4j-*.jar`, which may interfere with the logging in Oxygen XML Developer.
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 79)* 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 1742)* (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 2307)*.

Resources

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 1742)*, the various nodes include the following contextual menu actions:

Connection Level Nodes

Configure Database Sources

Opens the **Data Sources** preferences page *(on page 205)* 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 368)* 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 368)* 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 Developer 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 Developer 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.

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 368)* 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 402)*.

MarkLogic Database Connections (Deprecated)

Oxygen XML Developer Enterprise edition includes support for MarkLogic database connections. Once you configure a MarkLogic connection *(on page 1767)*, you can use the **Data Source Explorer** view *(on page 1742)* 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 1742)* 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/.

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 to you to spot possible issues without the need of actually executing the XQuery script.

When debugging XQuery files with MarkLogic *(on page 1770)*, you can use the **Data Source Explorer** view *(on page 1742)* to open the files from the application server that is involved in the debugging process. By using the **Data Source Explorer** view *(on page 1742)*, any imported modules are better identified by the MarkLogic server. You can also use step actions and breakpoints *(on page 1772)* in the modules to help identify problems.

Modules Container

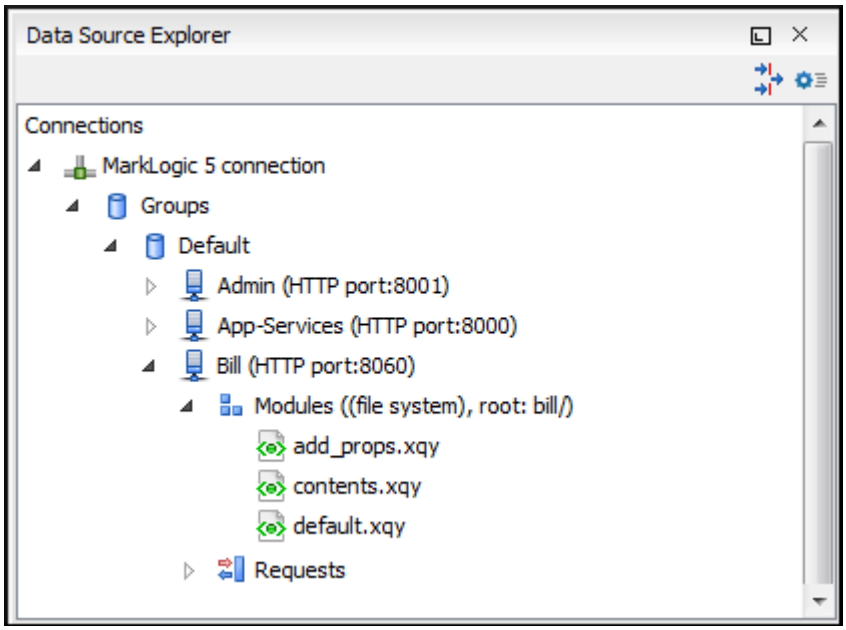
For each Application server (for example: *Bill (HTTP port:8060)*), you have access to the XQuery modules that are visible to that server. When editing, executing, or debugging XQuery it is recommended to open the XQuery files from this **Modules** container.

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 Developer 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 1774)*.

Figure 432. MarkLogic Connection in Data Source Explorer



Configuring a MarkLogic Database Connection

Note that this feature is available in Oxygen XML Developer 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 1768)*.
3. Configure a MarkLogic *Connection (on page 1768)*.
4. To view your connection, go to the **Data Source Explorer** view *(on page 1742)* (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 2307)*.

Related Information:

MarkLogic Development in Oxygen XML Developer *(on page 1769)*

How to Configure MarkLogic Data Source Drivers

Notes:

- Available in the Enterprise edition only.
- Oxygen XML Developer supports MarkLogic version 4.0 or later.

To configure a data source for MarkLogic, follow this procedure:

1. Download the **XCC Java distribution** zip file from: http://developer.marklogic.com/products/xcc.
2. Unzip the downloaded archive.
3. Open the **Preferences** dialog box **(Options > Preferences)** *(on page 79)* 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 1768)*.

How to Configure a MarkLogic Connection

Notes:

- Available in the Enterprise edition only.
- Oxygen XML Developer 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 79)* 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 Developer 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 1774)* 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 1742)*.
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 1742)* (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 2307)*.

MarkLogic Development in Oxygen XML Developer

The Oxygen XML Developer 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 779)* 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 778)* collects and displays all the functions from all imported modules. The *Content Completion Assistant (on page 2303)* 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 1742)*. 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 Developer 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 Developer 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 Developer 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 1767)*.
2. Expand the MarkLogic connection in the **Data Source Explorer** view *(on page 1742)* and open the *library* modules. The *main* module must also be opened from the **Data Source Explorer** view *(on page 1742)*.
3. Configure a validation scenario *(on page 526)* 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 2303)* and the **Outline** view *(on page 778)* should now present the functions from all the modules.

Related Information:
Debugging with MarkLogic *(on page 1770)*
Configuring a MarkLogic Database Connection *(on page 1767)*

Debugging with MarkLogic

Oxygen XML Developer 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 1768)* and a MarkLogic connection *(on page 1768)*.
2. Make sure that the debugging support is enabled in the MarkLogic server that Oxygen XML Developer 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 Developer uses to process XQuery expressions by selecting the **Use it to execute queries** action *(on page 1774)* from the contextual menu in the **Data Source Explorer** view *(on page 1742)*.

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 1742)* 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 275)*. If you have a MarkLogic validation scenario configured for the XQuery file, you can choose to debug the scenario *(on page 1845)* directly.
   - Otherwise, switch to the **XQuery Debugger** *perspective (on page 2307)*, open the XQuery file in the editor, and select the MarkLogic connection in the XQuery engine selector from the debug control toolbar *(on page 1828)*.

     For general information about how a debugging session is started and controlled, see the Working with the Debugger *(on page 1845)* section.

In a MarkLogic debugging session, you can use step actions and *breakpoints (on page 1849)* to help identify problems. When you add a *breakpoint (on page 1850)* on a line where the debugger never stops, Oxygen XML Developer 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 1772)*.

Remote Debugging with MarkLogic

Oxygen XML Developer allows you to debug remote applications that use XQuery (for example, web applications that trigger XQuery executions). Oxygen XML Developer 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 Developer 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.

Resources

For more information about the XQuery debugger for MarkLogic, watch our video demonstration:

https://www.youtube.com/embed/eQ4ThDZq1bk

Related Information:
MarkLogic Development in Oxygen XML Developer *(on page 1769)*
Configuring a MarkLogic Database Connection *(on page 1767)*

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 (on page 1849)* 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 *(on page 1742)*, open all the modules from the **Modules** container of the XDBC application server *(on page 1768)* that performs the debugging.
2. Set *breakpoints (on page 1850)* in the module as needed.
3. Continue debugging *(on page 1845)* the query.

If you get a warning that the *breakpoints* failed to initialize, try the following solutions:

- Check the **Breakpoints** view *(on page 1833)* 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.

Related Information:
MarkLogic Database Connections (Deprecated) *(on page 1766)*
MarkLogic Development in Oxygen XML Developer *(on page 1769)*

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 value** action from the contextual menu of the **Variables** view *(on page 1843)* and pasting it in the **XWatch** view *(on page 1835)*.
- There is no support for output to source mapping *(on page 1846)*.
- There is no support for showing the trace *(on page 1840)*.
- You can only set breakpoints *(on page 1833)* 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 1742)*.
  - When the debugger automatically opens the modules in the Editor.
- No *breakpoints (on page 1849)* are set in modules from the same server that are not involved in the current debugging session.
- No support for profiling *(on page 1850)* 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 1742)*, the various nodes include the following contextual menu actions:

Connection Level Nodes

Configure Database Sources

Opens the **Data Sources** preferences page *(on page 205)* 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 368)* 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 1770)*.

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 Developer 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 Developer 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 402)*.

Related Information:
Configuring a MarkLogic Database Connection *(on page 1767)*
MarkLogic Development in Oxygen XML Developer *(on page 1769)*
Debugging with MarkLogic *(on page 1770)*

MySQL Database Connections (Deprecated)

Oxygen XML Developer includes support for MySQL database connections. Oxygen XML Developer allows you to browse the structure of a SQL Server database in the **Data Source Explorer** view *(on page 1742)*, open tables in the **Table Explorer** view *(on page 1744)*, 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 1775)*.
2. Configure a MySQL *Connection*. *(on page 1776)*
3. To view your connection, go to the **Data Source Explorer** view *(on page 1742)* (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 2307)*.

How to Configure MySQL Data Source Drivers

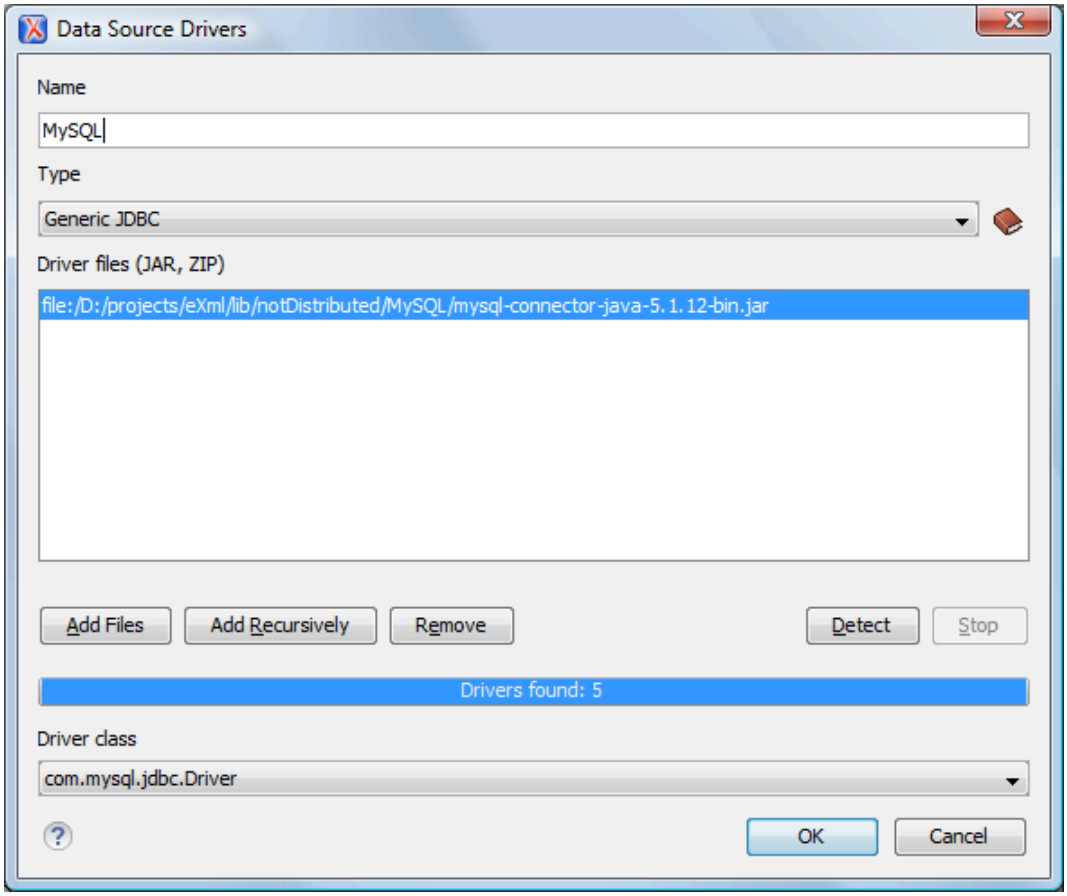
To connect to a MySQL server, you need to create a generic JDBC type data source based on the MySQL JDBC driver available on the MySQL website.

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 79)* 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 433. Data Source Drivers Configuration Dialog Box



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 1776)*.

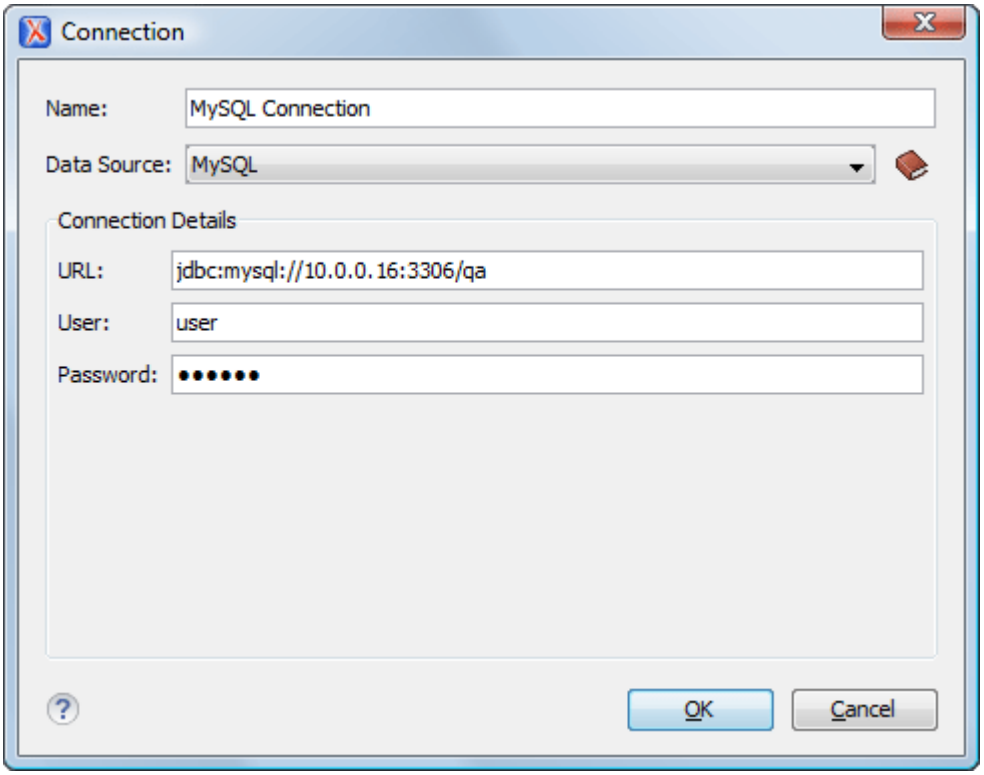
How to Configure a MySQL Connection

To configure a connection to a MySQL server, follow these steps:

1. Open the **Preferences** dialog box **(Options > Preferences)** *(on page 79)* 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 434. Connection Configuration Dialog Box



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 1742)* (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 2307)*.

Generic JDBC Database Connections

Oxygen XML Developer 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 1778)*.
2. Configure a Generic JDBC *Connection (on page 1778)*.
3. To view your connection, go to the **Data Source Explorer** view *(on page 1742)* (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 2307)*.

How to Configure Generic JDBC Data Source Drivers

Starting with version 17, Oxygen XML Developer comes bundled with Java 11, 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 Developer 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 79)* 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 1778)*.

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 79)* 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 1742)* (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 2307)*.

JDBC-ODBC Database Connections

Oxygen XML Developer includes support for JDBC-ODBC database connections.

How to Configure a JDBC-ODBC Connection

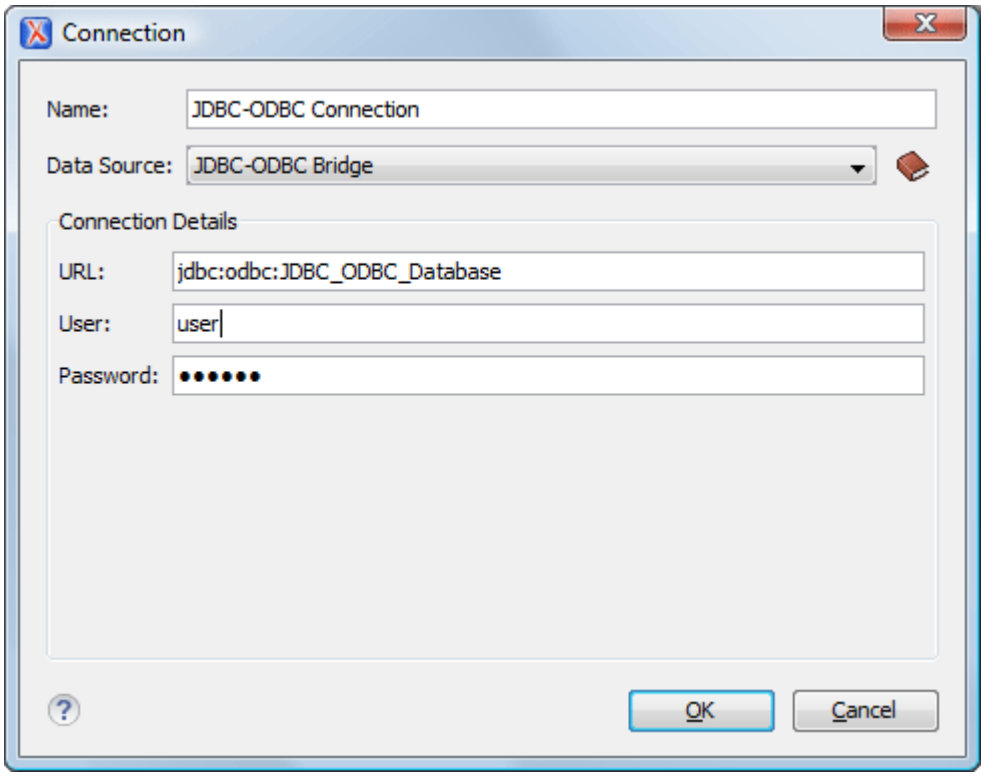
Starting with version 17, Oxygen XML Developer comes bundled with Java 11, which does not provide built-in access to JDBC-ODBC data sources. To access such sources, you need to find an alternative JDBC-ODBC bridge or use a platform-independent distribution of Oxygen XML Developer along with a Java VM 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 79)* 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 435. 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 1742)* (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 2307)*.

BaseX Database Connections

Oxygen XML Developer includes support for BaseX database connections using a WebDAV connection. BaseX is a light-weight XML database engine and XQuery processor. Oxygen XML Developer allows you to browse the structure of a BaseX database in the **Data Source Explorer** view *(on page 1742)* 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 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 Developer as described in How to Configure a WebDAV Connection *(on page 1789)*. 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 1742)* (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 2307)*.

BaseX Contextual Menu Actions

While browsing BaseX connections in the **Data Source Explorer** view *(on page 1742)*, the various nodes include the following contextual menu actions:

Connection Level Nodes

Configure Database Sources

Opens the **Data Sources** preferences page *(on page 205)* 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 368)* 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 368)* 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 Developer 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 Developer 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 368)* 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 402)*.

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 scrips, 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 1783)*. 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 1784)*.

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 Developer and create an XQuery Transformation on XML *(on page 1190)*. 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.

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 Developer after configuring the environment variables.
2. Open the **Preferences** dialog box **(Options > Preferences)** *(on page 79)* 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 Developer detects any implementation of *javax.xml.xquery.XQDataSource* and presents it in **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 1784)*.

How to Configure an XQJ Connection

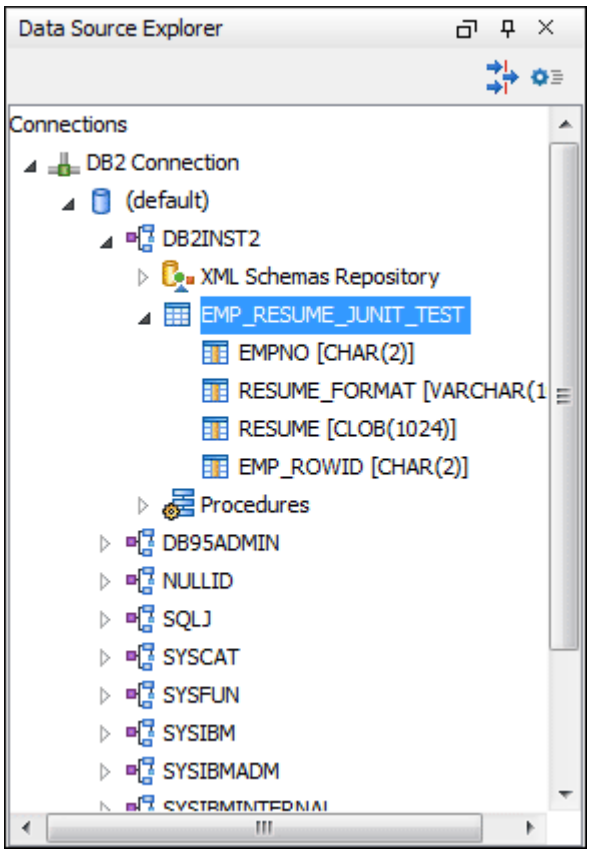
The steps for configuring an XQJ connection are the following:

1. Open the **Preferences** dialog box **(Options > Preferences)** *(on page 79)* 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 1783)* 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.

IBM DB2 Database Connections (Deprecated)

Oxygen XML Developer includes support for IBM DB2 database connections. Oxygen XML Developer allows you to browse the structure of an IBM DB2 database in the **Data Source Explorer** view *(on page 1742)*, open tables in the **Table Explorer** view *(on page 1744)*, and perform various operations on the resources in the repository.

Figure 436. IBM DB2 Database Connection



Configuring an IBM DB2 Database Connection (Deprecated)

To configure the support for the IBM DB2 database, follow this procedure:

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. Unzip the zip file and use the `db2jcc.jar` and `db2jcc_license_cu.jar` files in Oxygen XML Developer for configuring a DB2 data source *(on page 1785)*.
2. Configure IBM DB2 *Data Source* drivers *(on page 1785)*.
3. Configure an IBM DB2 Server *Connection (on page 1787)*.
4. To view your connection, go to the **Data Source Explorer** view *(on page 1742)* (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 2307)*.

How to Configure IBM DB2 Data Source Drivers (Deprecated)

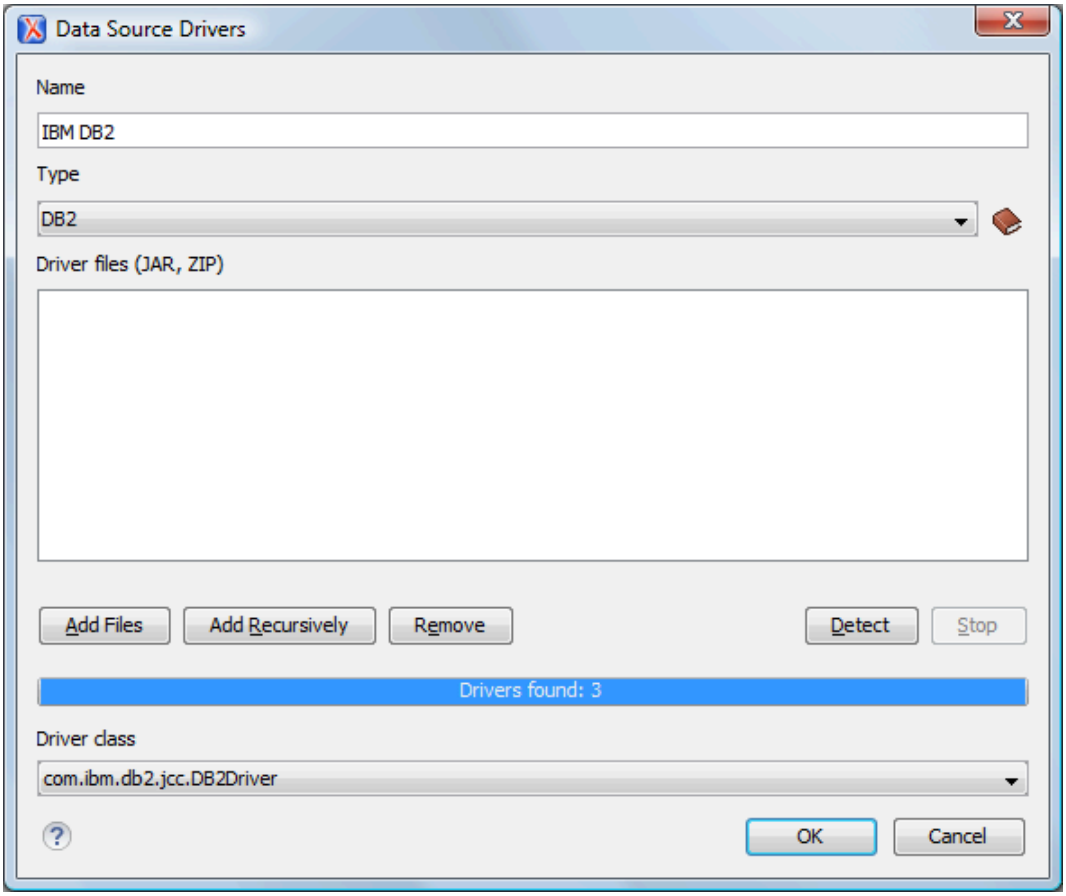
Note:
Available in the Enterprise edition only.

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 79)* 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 437. 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 1787)*.

How to Configure an IBM DB2 Connection (Deprecated)

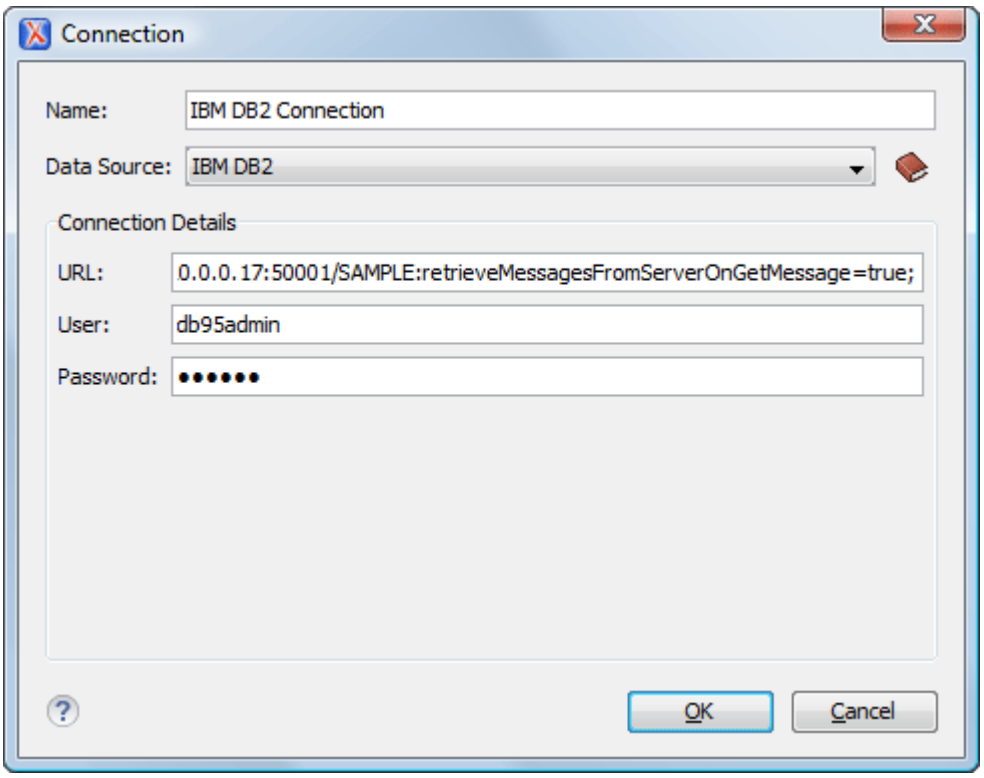
Note:
The support to create an IBM DB2 connection is available in the Enterprise edition only.

To configure a connection to an IBM DB2 server, follow these steps:

1. Open the **Preferences** dialog box **(Options > Preferences)** *(on page 79)* 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 438. 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 1742)* (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 2307)*.

IBM DB2 Contextual Menu Actions (Deprecated)

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 1742)*, 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 205)* 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 1744)*.

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 1820)* chapter).

Database-Specific Contextual Menu Actions

In addition to the general contextual menu actions in the **Data Source Explorer** view *(on page 1742)*, 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 Developer.

WebDAV Connections

Oxygen XML Developer includes support for WebDAV server connections. Oxygen XML Developer allows you to browse the structure of a WebDAV connection in the **Data Source Explorer** view *(on page 1742)* and perform various operations on the resources in the repository.

How to Configure a WebDAV Connection

By default, Oxygen XML Developer 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 1742)*.

To configure a WebDAV connection, follow these steps:

1. Open the **Preferences** dialog box **(Options > Preferences)** *(on page 79)* 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 1742)* (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 2307)*.
   For more information about the WebDAV support in Oxygen XML Developer, watch our video demonstration:

   https://www.youtube.com/embed/vDXO36CqbvM

WebDAV Contextual Menu Actions

While browsing WebDAV connections in the **Data Source Explorer** view *(on page 1742)*, the various nodes include the following contextual menu actions:

Connection Level Nodes

Configure Database Sources

Opens the **Data Sources** preferences page *(on page 205)* 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 368)* 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 368)* 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 Developer 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 Developer 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 368)* 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 402)*.

SQL Execution Support

The database support in Oxygen XML Developer includes support for writing SQL statements, syntax highlighting, *folding (on page 2305)*, and dragging and dropping from the **Data Source Explorer** view *(on page 1742)*. It also includes transformation scenarios for executing the statements, and the results are displayed in the **Table Explorer** view *(on page 1744)*.

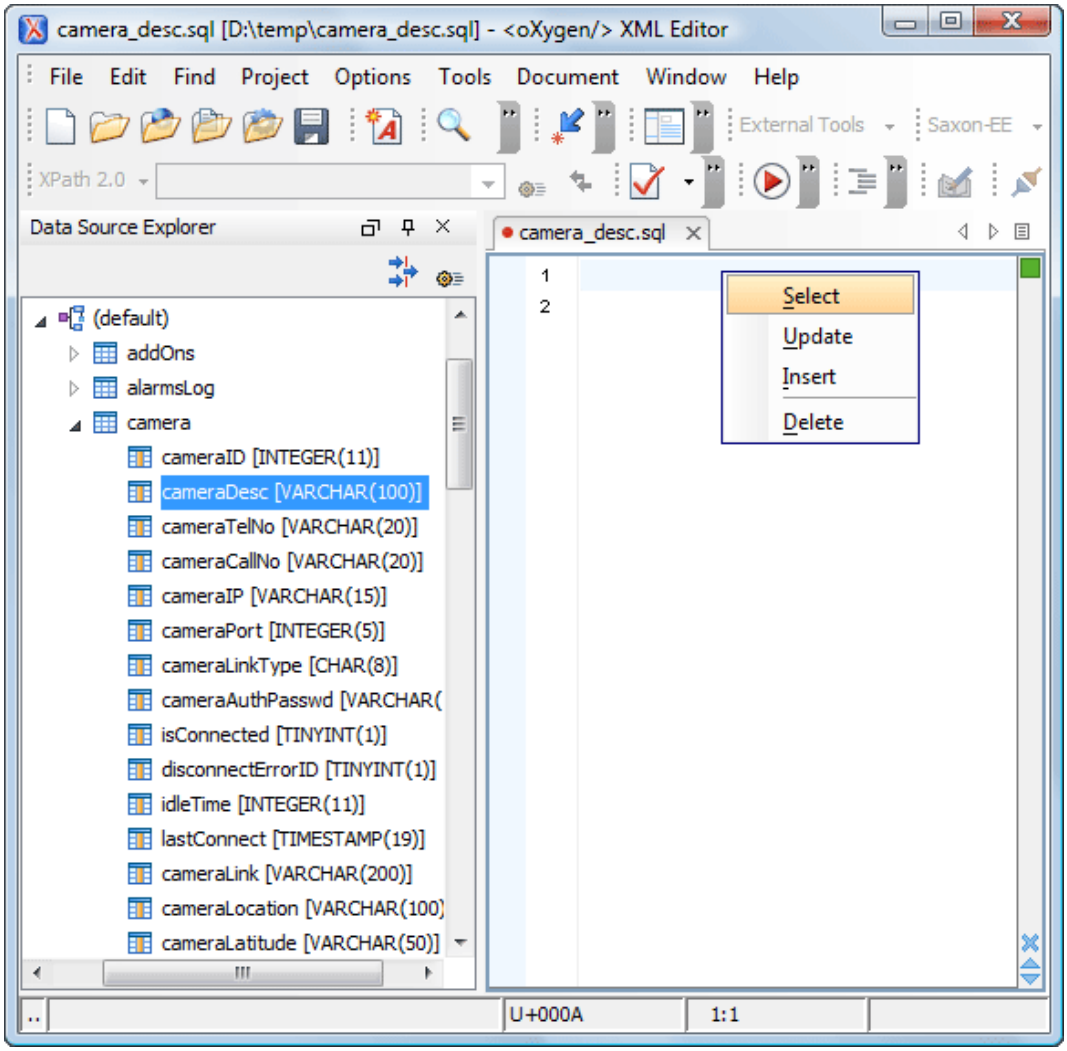
Drag and Drop from Data Source Explorer View

Dragging operations from the **Data Source Explorer** view *(on page 1742)* 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 1747)* 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.

Figure 439. SQL Statement Editing with Drag and Drop



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 1070)* 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 place holders (?) 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 474)* at the bottom of the Oxygen XML Developer 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 Developer.

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 Developer offer XQuery support:

- *Native XML Databases:*
  - eXist
  - MarkLogic (validation support available starting with version 5)
- *Relational Databases:*
  - IBM DB2
  - Microsoft SQL Server (validation support not available)
  - Oracle (validation support not available)

Related information
Editing XQuery Documents *(on page 774)*

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 1742)* and dropping them into the editor panel.

1. Configure the data source drivers *(on page 1747)* for the particular relational database in the **Data Sources** preferences page *(on page 205)*.
2. Configure the connection *(on page 1747)* for the particular relational database in the **Data Sources** preferences page *(on page 205)*.
3. Browse the connection in the **Data Source Explorer** view *(on page 1742)*, 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 Developer, 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 Developer 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 774)*

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 1747)* 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**.

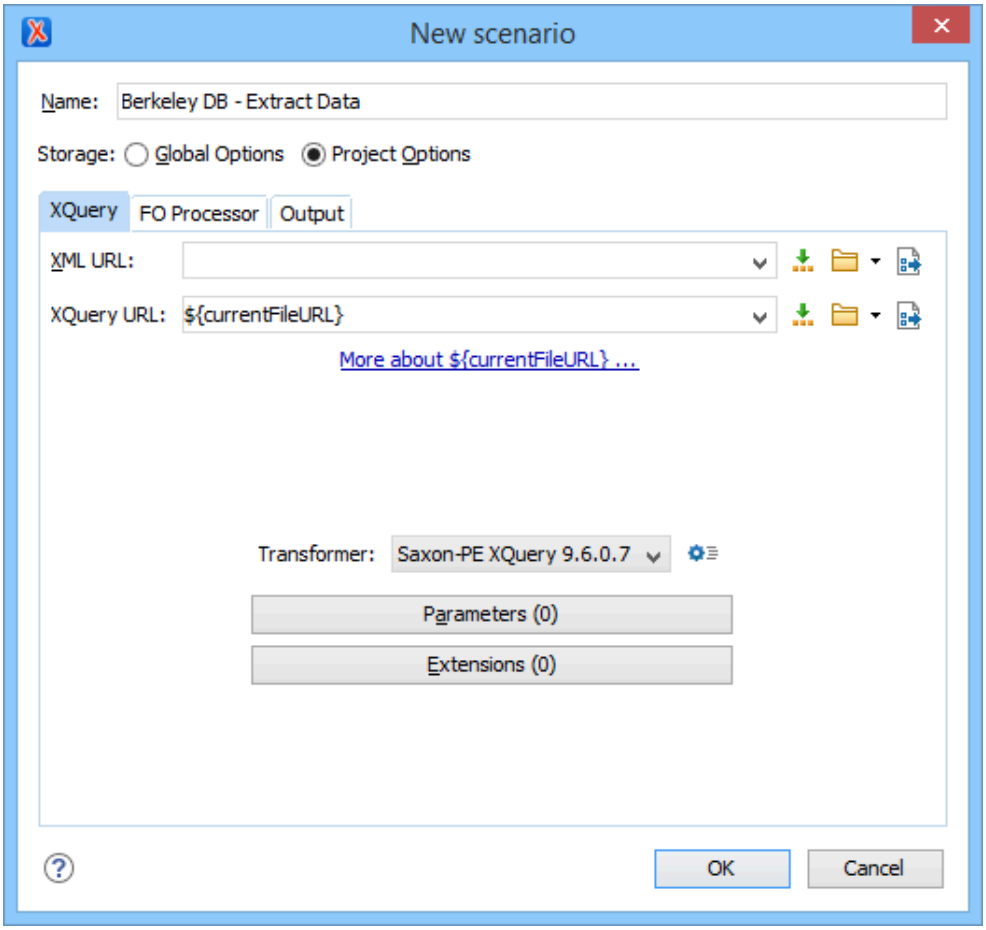
    The **Configure Transformation Scenario** dialog box *(on page 1210)* is opened.

    b. Click the **New** button toward the bottom of the dialog box.

    c. Select **XML Transformation with XQUERY** *(on page 1117)*.

    The **New Scenario** dialog box for configuring an XQuery scenario is opened.

Figure 440. New Scenario Dialog Box



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 185)* 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 2303)* 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 Developer 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 Developer).

Related information

XML Transformation with XQuery *(on page 1117)*
XQuery XQJ Transformation *(on page 1798)*

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 scrips, therefore XQuery 3.0 and 3.1 scripts are not supported.

Oxygen XML Developer 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 1783)*.
2. Configure an XQJ *Connection (on page 1784)*.
3. To view your connection, go to the **Data Source Explorer** view *(on page 1742)* (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 2307)*.

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.

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 Developer after configuring the environment variables.

2. Open the **Preferences** dialog box **(Options > Preferences)** *(on page 79)* 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 Developer detects any implementation of *javax.xml.xquery.XQDataSource* and presents it in **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 1784)*.

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 79)* 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 1783)* 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 Developer 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 Developer, see Debugging XSLT Stylesheets and XQuery Documents *(on page 1826)*.

Debugging with MarkLogic

Oxygen XML Developer 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 1768)* and a MarkLogic connection *(on page 1768)*.
2. Make sure that the debugging support is enabled in the MarkLogic server that Oxygen XML Developer 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 Developer uses to process XQuery expressions by selecting the **Use it to execute queries** action *(on page 1774)* from the contextual menu in the **Data Source Explorer** view *(on page 1742)*.

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 1742)* 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 275)*. If you have a MarkLogic validation scenario configured for the XQuery file, you can choose to debug the scenario *(on page 1845)* directly.
   - Otherwise, switch to the **XQuery Debugger** *perspective (on page 2307)*, open the XQuery file in the editor, and select the MarkLogic connection in the XQuery engine selector from the debug control toolbar *(on page 1828)*.

     For general information about how a debugging session is started and controlled, see the Working with the Debugger *(on page 1845)* section.

In a MarkLogic debugging session, you can use step actions and *breakpoints (on page 1849)* to help identify problems. When you add a *breakpoint (on page 1850)* on a line where the debugger never stops, Oxygen XML Developer 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 1772)*.

Remote Debugging with MarkLogic

Oxygen XML Developer allows you to debug remote applications that use XQuery (for example, web applications that trigger XQuery executions). Oxygen XML Developer 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 Developer 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.

Resources

For more information about the XQuery debugger for MarkLogic, watch our video demonstration:

https://www.youtube.com/embed/eQ4ThDZq1bk

Related Information:
MarkLogic Development in Oxygen XML Developer *(on page 1769)*
Configuring a MarkLogic Database Connection *(on page 1767)*

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 (on page 1849)* 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 *(on page 1742)*, open all the modules from the **Modules** container of the XDBC application server *(on page 1768)* that performs the debugging.
2. Set *breakpoints (on page 1850)* in the module as needed.
3. Continue debugging *(on page 1845)* the query.

If you get a warning that the *breakpoints* failed to initialize, try the following solutions:

- Check the **Breakpoints** view *(on page 1833)* 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.

Related Information:
MarkLogic Database Connections (Deprecated) *(on page 1766)*
MarkLogic Development in Oxygen XML Developer *(on page 1769)*

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 value** action from the contextual menu of the **Variables** view *(on page 1843)* and pasting it in the **XWatch** view *(on page 1835)*.
- There is no support for output to source mapping *(on page 1846)*.
- There is no support for showing the trace *(on page 1840)*.
- You can only set breakpoints *(on page 1833)* 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 1742)*.
  - When the debugger automatically opens the modules in the Editor.
- No *breakpoints (on page 1849)* are set in modules from the same server that are not involved in the current debugging session.
- No support for profiling *(on page 1850)* when an XQuery transformation is executed in the debugger.

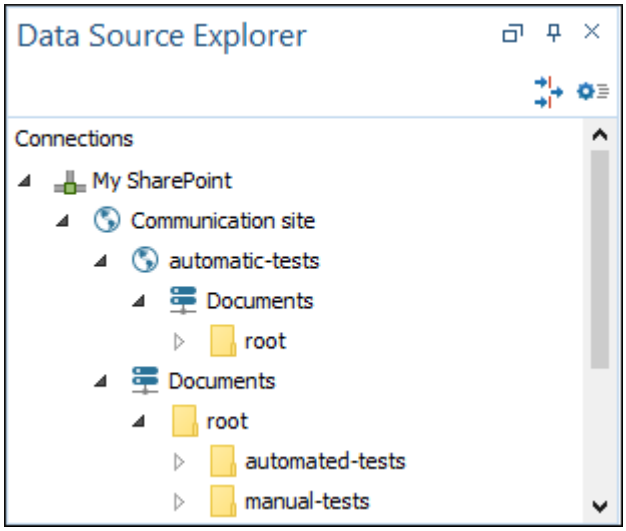
Integration with Microsoft SharePoint

Restriction:
The SharePoint integration is only available in the Enterprise edition of Oxygen XML Developer.

Oxygen XML Developer provides support for browsing and managing SharePoint connections in the **Data Source Explorer** view *(on page 1742)* and there is also a specialized **SharePoint Browser** view *(on page 1807)*. 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.

The **SharePoint Online** implementation uses the SharePoint REST API v.2 and it supports OAuth credentials (access tokens).

Figure 441. SharePoint Connection in Data Source Explorer View



Related Information:
Working with Databases *(on page 1742)*

How to Configure a SharePoint Connection

By default, Oxygen XML Developer 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 1807)*.

The **SharePoint Online** connection that uses the SharePoint REST API v.2 and it supports 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 1807)*).
   - Open the **Preferences** dialog box **(Options > Preferences)** *(on page 79)*, 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 1807)*, 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. If you have problems with the log-in process, see Troubleshooting SharePoint Online Connections *(on page 1804)*.

Note:
If you are still logged in when you close Oxygen XML Developer, 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 Developer by using the **Log in with Microsoft account** action.

Troubleshooting SharePoint Online Connections

Allowed SharePoint Online Sites

SharePoint Online sites supported by Oxygen XML Developer have the following syntax:

- `https://tenant.SharePoint.com`
- `https://tenant.SharePoint.com/sites/siteName`
- `https://tenant.SharePoint.com/sites/siteName/subsiteName1/..../subsiteNameK`
- `https://tenant.SharePoint.com/teams/siteName`
- `https://tenant.SharePoint.com/teams/siteName/subsiteName1/..../subsiteNameK`

Authentication Workflow Problems

Once you configure the SharePoint Online Connection *(on page 1803)* and use the *Log in with Microsoft account* action, the action will open the default browser for authentication. If this is the fist time you access this SharePoint site, you will have to Grant Permissions to Oxygen XML Developer *(on page 1805)*.

If the authentication is successful, the browser should display the **Authentication complete** page:



Attention:
If you cannot get past the *This page isn't working* response while using *Google Chrome*, try using a different browser.

Once you go back to Oxygen XML Developer, in the *SharePoint Browser* you should see your username details and be able to browse the repository.

Grant Permissions to Oxygen XML Developer

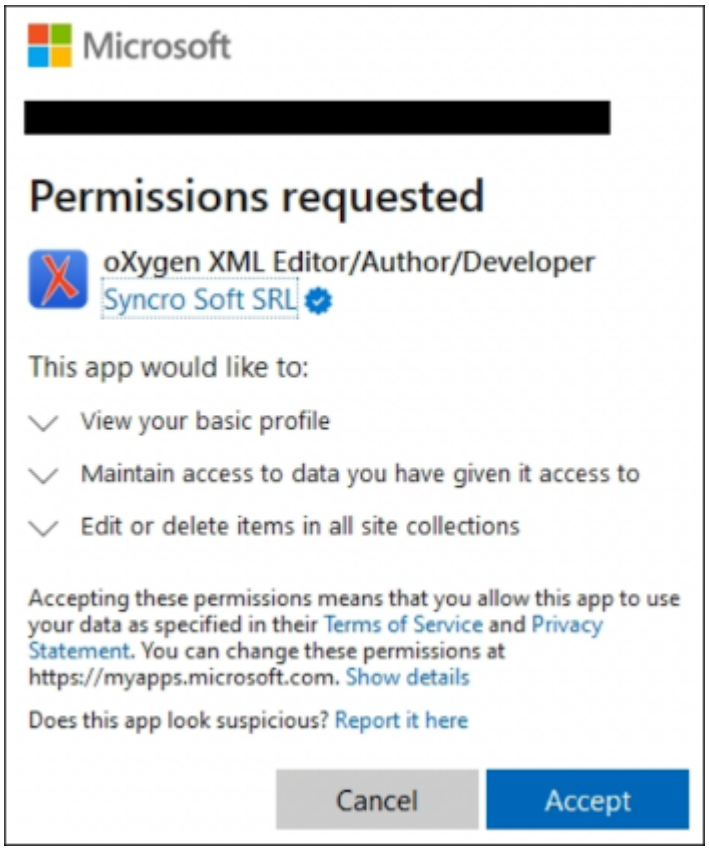
When you first **Log in** and access your **SharePoint** server, you may need to grant **SharePoint Enterprise Application** permissions for the Oxygen XML Developer application to:

- View your basic profile.
- Maintain access to data you have given access to.
- Edit or delete items in all site collections.

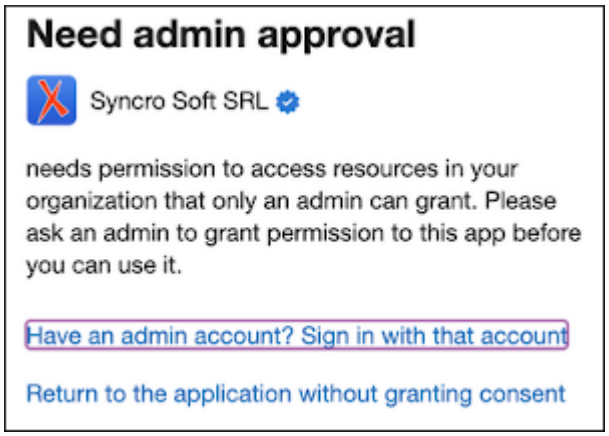
The **Permissions requested** form will appear when you first authenticate to your **SharePoint** server from Oxygen XML Developer.

If you have administrative privileges, you are able to grant permissions directly from the form:

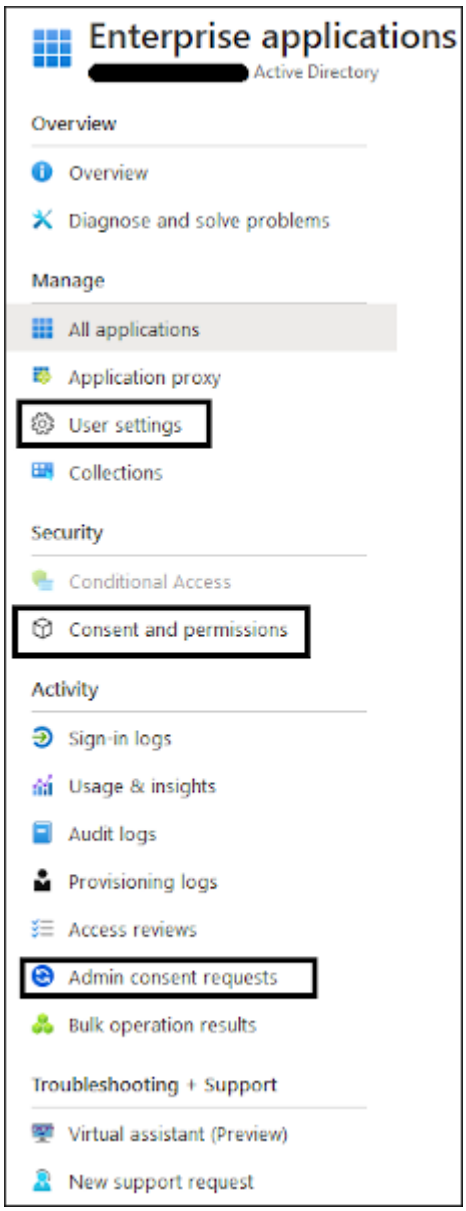
Figure 442. Permissions Requested Form



If you do not have admin rights to give Oxygen XML Developer permissions to access the **SharePoint** server, the **Permissions requested** form will suggest that you contact the administrator for your **SharePoint** account so that they can grant the permissions.



The **SharePoint** global administrator should log in to https://portal.azure.com/, navigate to **Manage Azure Active Directory > Enterprise applications**, and approve the request under the **Admin consent requests** category.



The administrator can also check the **User settings** category and configure the **Users can request admin consent to apps they are unable to consent to** options and policies.

More consent settings can be configured by the admin under the **User consent settings** category.

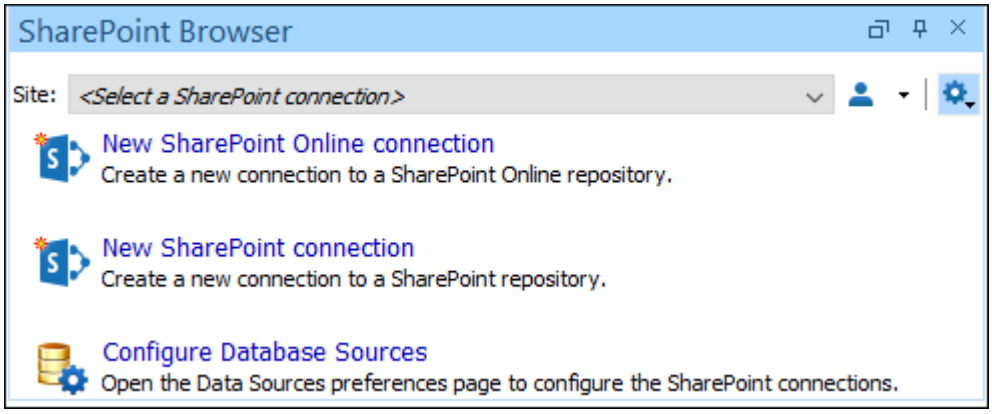
SharePoint Browser View

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 1803)* (these actions are also available in the **Settings** drop-down menu).

Figure 443. 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 1803)*.

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 Developer.
- **Log out** (if logged in) - Logs out of your Microsoft account and the authorization between Oxygen XML Developer 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.
- **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**.

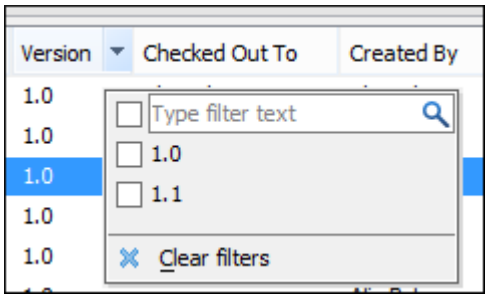
Once you are connected, the view is separated into two panes. The left pane is a navigation area that presents the SharePoint site structure in a tree-like fashion with the following node types:

- **Site**
- **Sub-site**
- **Folders**

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 444. Column Filter



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**. You have the option to enter a comment and click the **Check In** button to process it.

Related Information:
How to Configure a SharePoint Connection *(on page 1803)*

SharePoint Contextual Menu Actions

While browsing SharePoint connections in the **Data Source Explorer** view *(on page 1742)* or the **SharePoint Browser** view *(on page 1807)*, 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 205)* 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 Developer.

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 368)* 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 368)* 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.

Refresh

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 368)* 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 Developer 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 Developer 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. You have the option to enter a comment and click the **Check In** button to process it.

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.

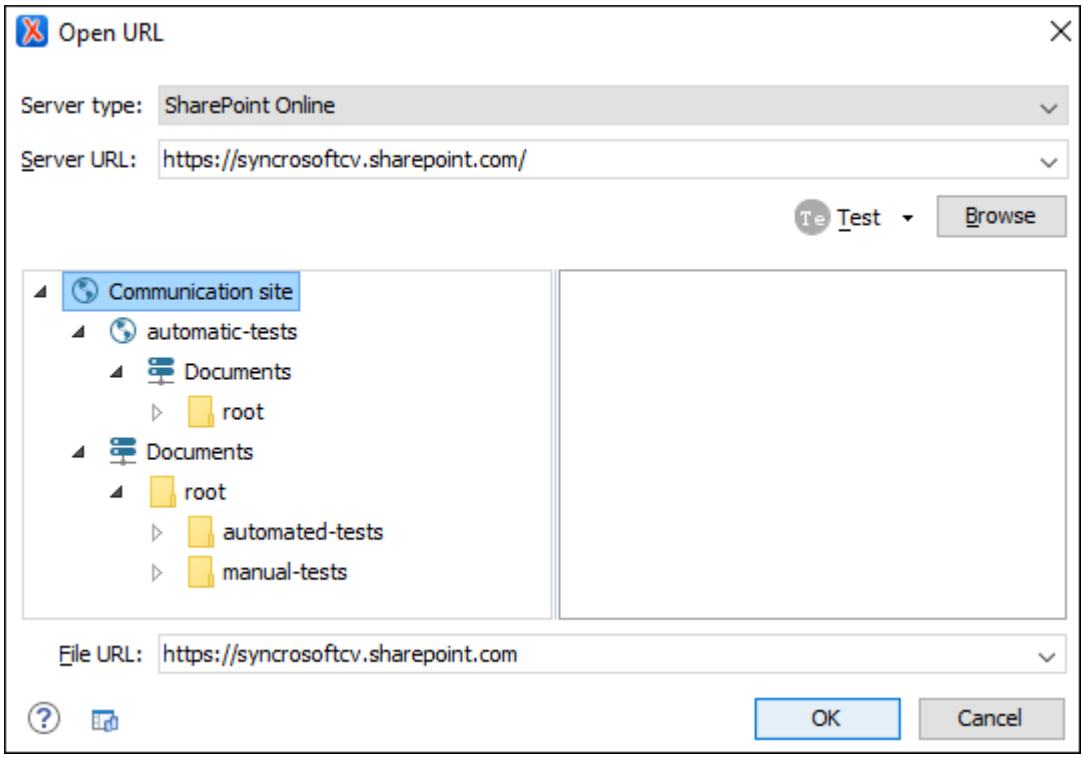
Compare

Compares two selected resources using the **Compare Files** tool *(on page 402)*.

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 1807)*. 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 445. 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 318)*.

Server URL

Specifies the protocol 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 Developer.
- **Log out** (if logged in) - Logs out of your Microsoft account and the authorization between Oxygen XML Developer 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 261)*:

- `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 41. API Permissions - Microsoft Graph

Microsoft Graph			
Name	**Type**	**Description**	**Admin consent**
email	Delegated	View users' email address	No
Files.ReadWrite.All	Delegated	Have full access to all files user can access	No
offline_access	Delegated	Maintain access to data you have given it access to	No
openid	Delegated	Sign users in	No
profile	Delegated	View users' basic profile	No
Sites.ReadWrite.All	Delegated	Edit or delete items in all site collections	No

Table 41. API Permissions - Microsoft Graph (continued)

Microsoft Graph			
User.Read	Delegated	Sign in and read user profile	No

Table 42. API Permissions - SharePoint

SharePoint			
Name	**Type**	**Description**	**Admin consent**
AllSites.Read	Delegated	Read items in all site collections	No
AllSites.Write	Delegated	Read and write items in all site collections	No

Notice:
The *Redirect URI* should be set to: `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 Developer 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 1070)*.

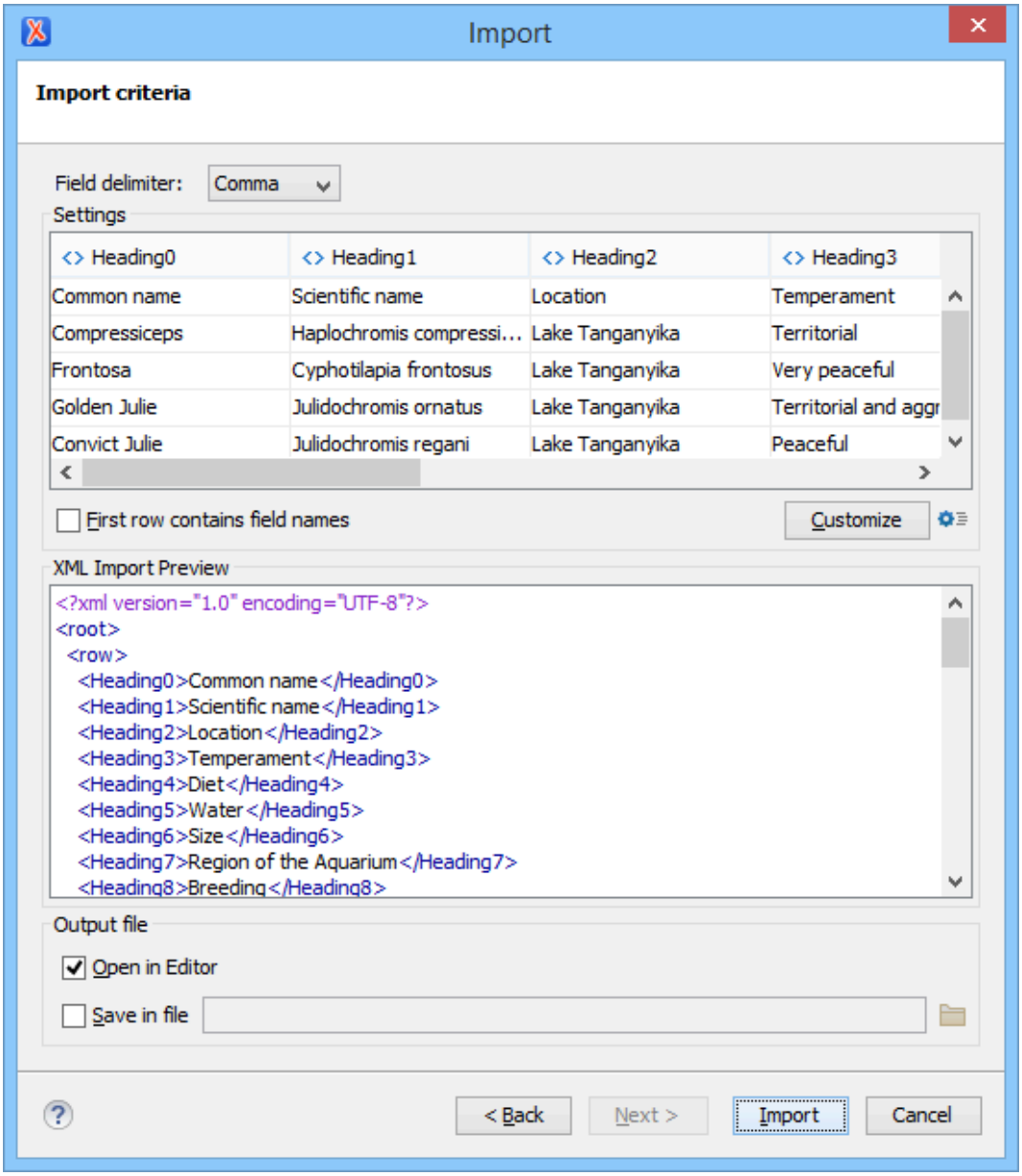
Import from Text Files

Oxygen XML Developer 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.

Figure 446. Import Criteria Dialog Box



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 197)* 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 Developer provides several methods for importing MS Excel files into an XML file. You can copy data from Excel and paste it into inserted cells in **Grid** mode. If you want to import an entire Excel file, Oxygen XML Developer also offers a configurable import wizard that works with any type of XML document.

Grid Mode Method

The **Grid** mode in Oxygen XML Developer 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.

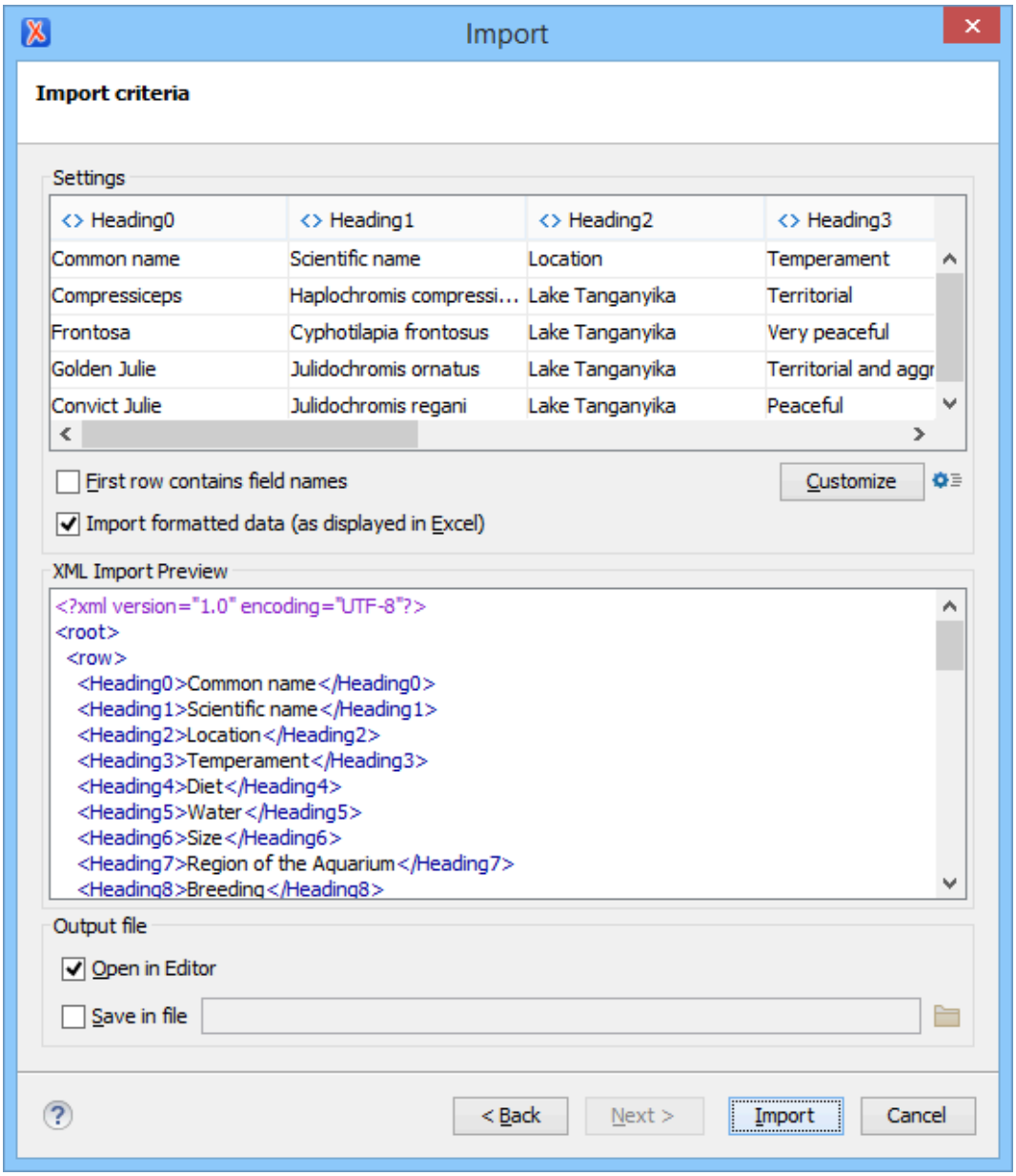
1. Copy the particular cells from your Excel spreadsheet that you want to import into an XML file.
2. Switch to **Grid** mode in Oxygen XML Developer.
3. Expand the particular nodes and locate the nested table where you want to insert the copied cells.
4. Right-click a particular row or column where 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 Developer 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 447. Import Wizard



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 197)* 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.

Resources

For more information about exchanging data between Oxygen XML Developer and spreadsheet applications, watch our video demonstration:

https://www.youtube.com/embed/8VwsF58zLkU

Related information

Exporting XML Content to Excel *(on page 513)*

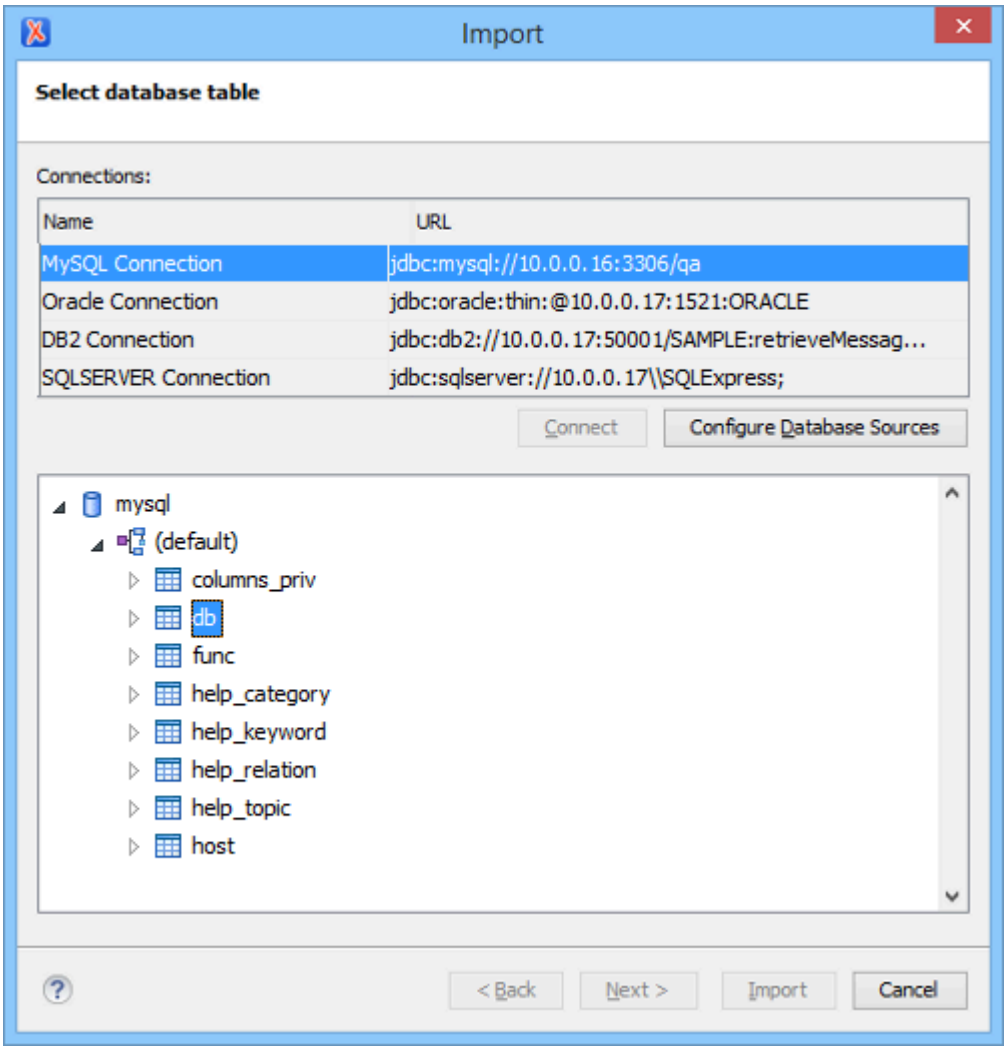
Import Database Data as an XML Document

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 448. Select Database Table Dialog Box



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.

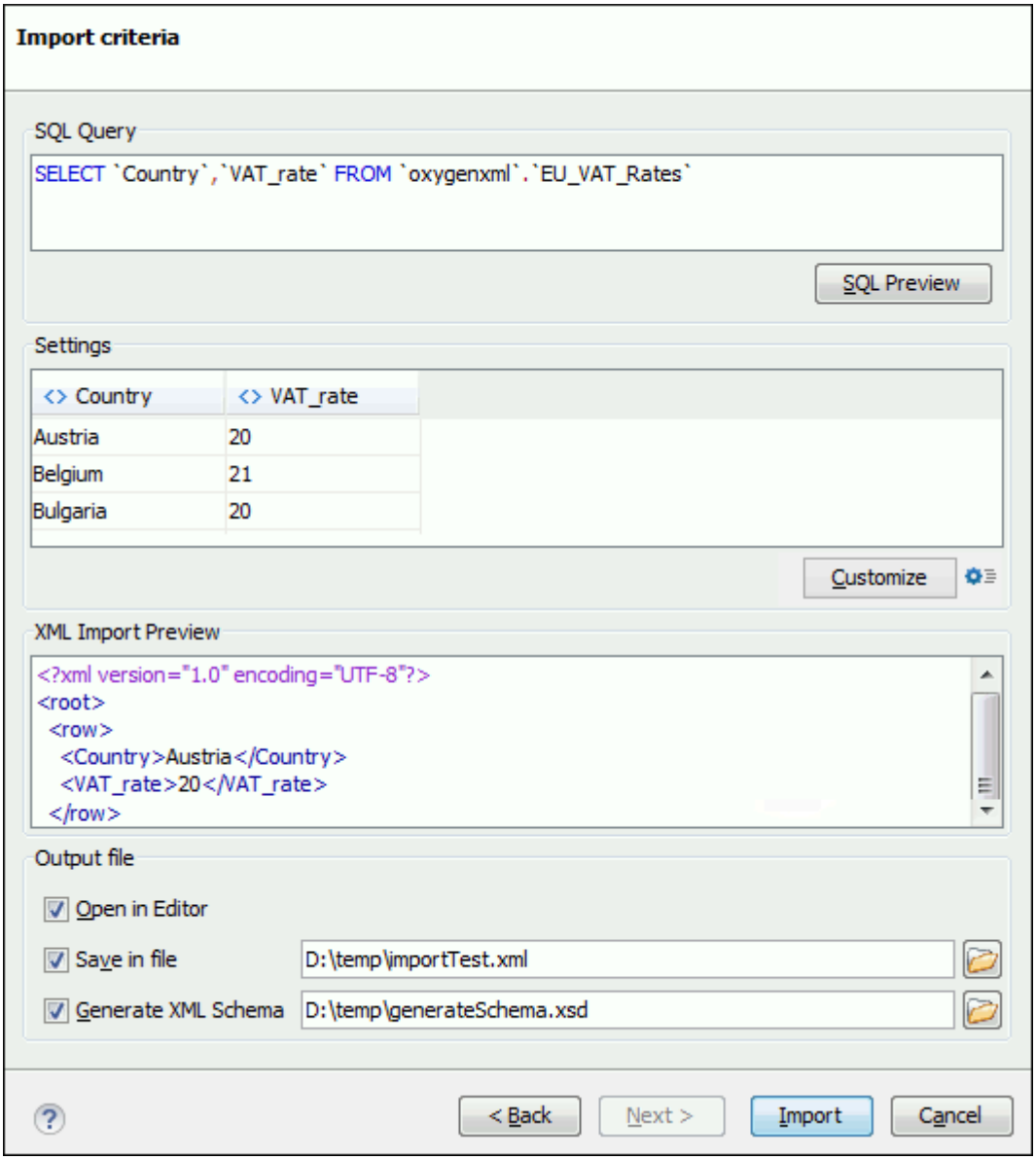
4. Click **Connect**.

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 449. Import from Database Criteria Dialog Box



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 (<> 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.
   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 197)* 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 Developer offers support for importing HTML files into an XML document.

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 HTTPS or SFTP), Oxygen XML Developer also provides special support for a *convert* protocol that can be used to chain predefined processors to dynamically import content from various sources.

Important:
Starting with version 26, the dynamic conversion protocol is disabled by default. To enable it, you must set the `com.oxygenxml.enable.convert.url.protocol` system property *(on page 269)* to the value of `true`.

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 2309)* 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 2304)* 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 Developer. 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.

```
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.

```
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.

```
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 2306)* 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:

```
 public void convert(String systemID, String originalSourceSystemID,
InputStream is, OutputStream os, LinkedHashMap<String, String> properties)
throws IOException
```

```
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.

```
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.

```
convert:/processor=json!/urn:files:personal.json
```

- **`xhtml` Processor** - Converts HTML content to well-formed XHTML. It has no parameters.

```
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 Developer 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 **Libaries** button in the **Advanced** tab of the transformation scenario.

Related Information:
https://github.com/oxygenxml/dita-glass

16.

Debugging XSLT Stylesheets and XQuery Documents

Oxygen XML Developer 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 2307)* 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 2307)* 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 2307)* 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 12.9 XSLT engine) and XQuery 1.0 / 3.0 (using Saxon 12.9 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 1849)* 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.

Resources

For even more information, watch our video demonstration:

https://www.youtube.com/embed/m9d8c4V-LJw

Debugger Layout

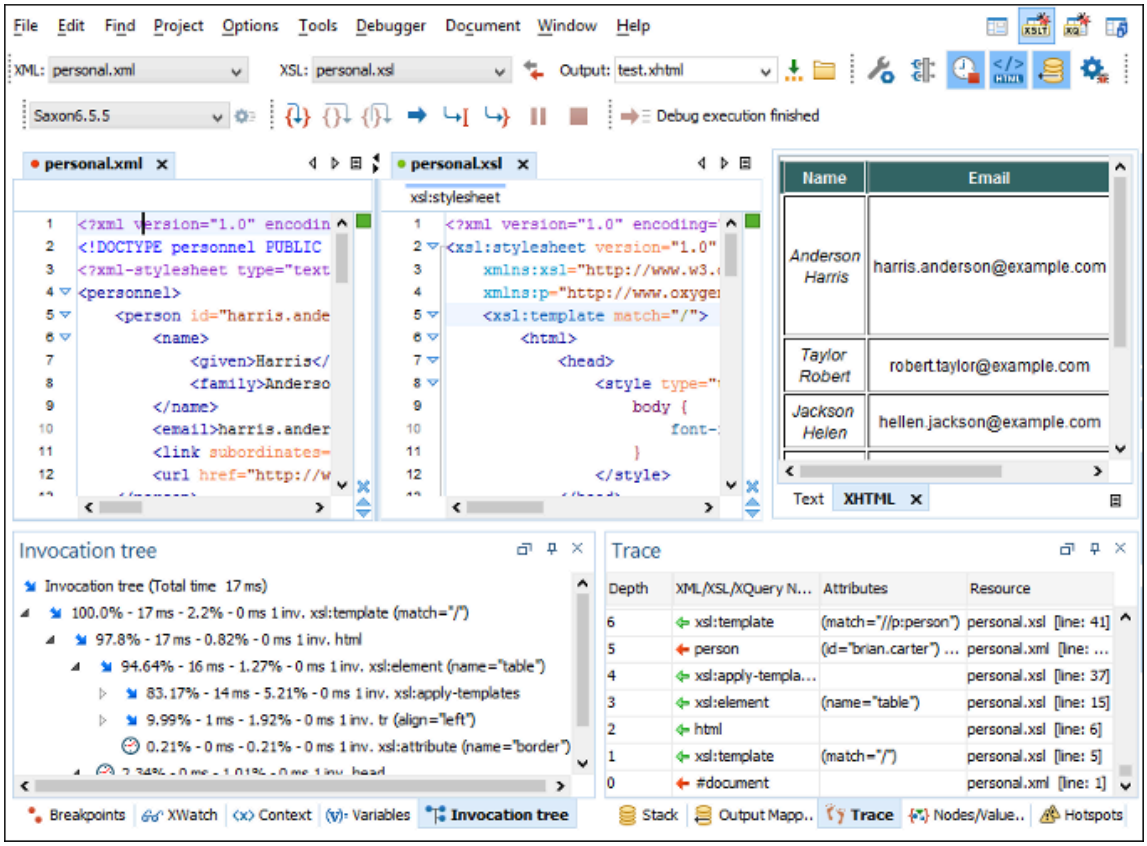
The XML and XSL files are displayed in **Text** mode *(on page 442)*. The **Grid** mode *(on page 279)* is available only in the Editor *perspective (on page 272)*.

The **XSLT/XQuery Debugger** *perspective (on page 2307)* 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 1831)*). 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 188)*).
- **Control Toolbar** *(on page 1828)* - Contains a variety of actions to help you configure and control the debugging process.
- **Information Views** *(on page 1832)* - 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 2304)* so that you can configure the workspace according to your preferences.

Figure 450. Debugger Interface



XML documents and XSL stylesheets or XQuery documents that were opened in the **Editor** *perspective (on page 2307)* 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 444) are replaced in the **Debugger** *perspective* by breakpoints *(on page 1849)*.

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 1845)*

Identify the XSLT / XQuery Expression that Generated Particular Output *(on page 1846)*

Supported Processors for XSLT / XQuery Debugging *(on page 1856)*

Performance Profiling of XSLT Stylesheets and XQuery Documents *(on page 1850)*

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.

Figure 451. Control Toolbar



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 252)* 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 2306)* 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 1827)* 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 2307)* 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.

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.

Debugger preferences

Quick link to Debugger preferences page *(on page 188)*.

XSLT / XQuery engine selector

Lists the processors available for debugging XSLT and XQuery transformations *(on page 1856)*.

XSLT / 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 1106)*.

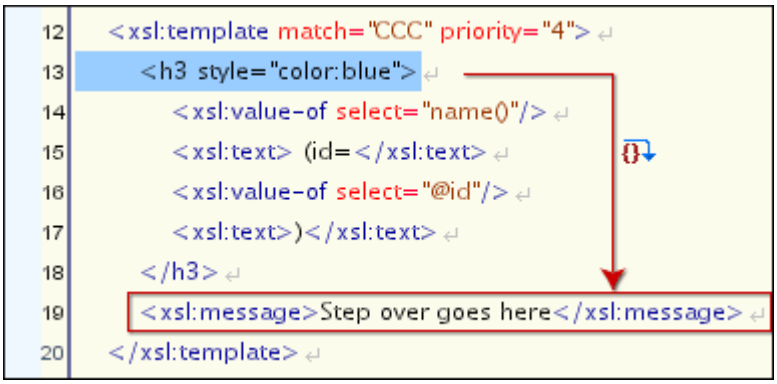
Step into

Starts the debugging process and runs until the next instruction is encountered.

Step over

Run until the current instruction and its sub-instructions are over. Usually this will advance to the next sibling instruction.

Figure 452. Step over



Step out

Run until the parent of the current instruction is over. Usually this will advance to the next sibling of the parent instruction.

Figure 453. Step out



Run Shift + F5

Starts the debugging process. The execution of the process is paused when a breakpoint *(on page 1833)* 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 1849)* is reached or the execution ends.

Run to end

Runs the transformation until the end, without taking into account enabled *breakpoints (on page 1849)*, 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 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 1120)*

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 1833)*
- **XWatch** view *(on page 1835)*
- **Context** view *(on page 1834)*
- **Messages** view *(on page 1836)* (XSLT only)
- **Variables** view *(on page 1843)*
- **Invocation Tree** view *(on page 1852)*

Right side information views

- **Stack** view *(on page 1837)*
- **Output Mapping Stack** view *(on page 1838)*
- **Trace** view *(on page 1840)*
- **Templates** view *(on page 1841)* (XSLT only)
- **Nodes/Values Set** view *(on page 1842)*
- **Hotspots** view *(on page 1853)*

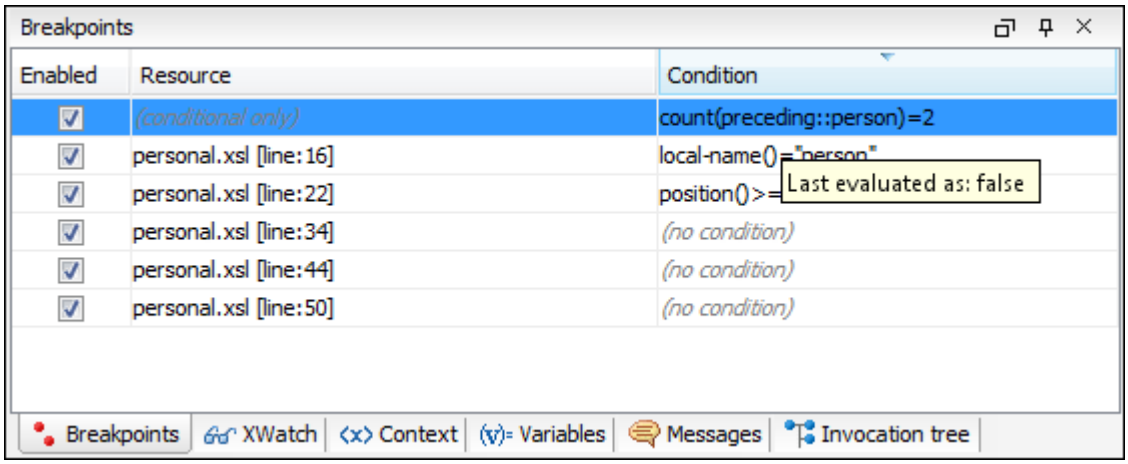
Tip:
The information views are dockable *(on page 2304)* so that you can configure the workspace according to your preferences.

Breakpoints View

The **Breakpoints** view lists all *breakpoints (on page 1849)* 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 1850)* 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**.

Figure 454. Breakpoints View



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.

Note:
The *breakpoints* list is not deleted at the end of a transformation (it is preserved between debugging sessions).

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

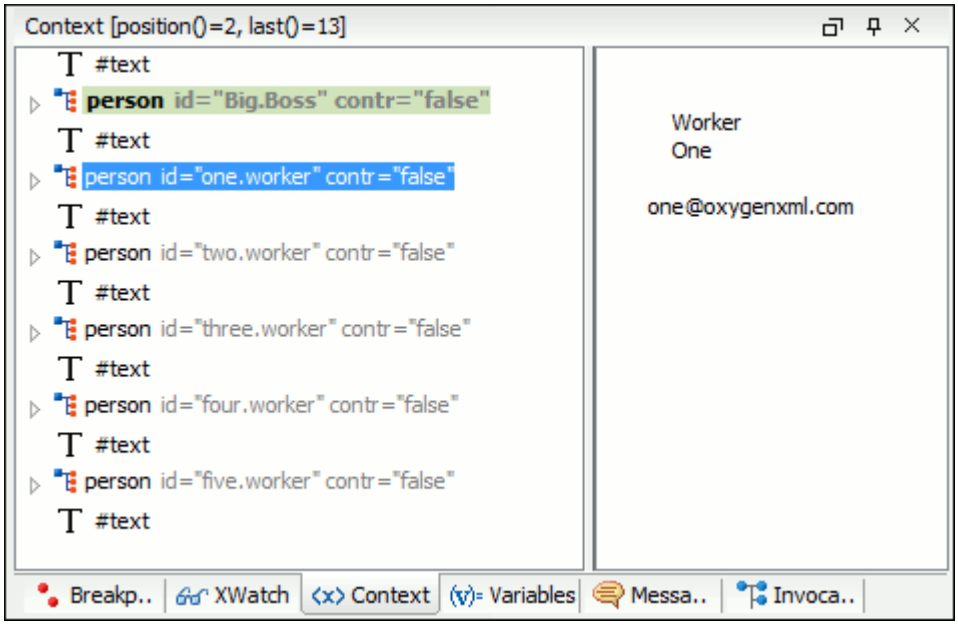
Removes all *breakpoints*.

Related Information:
Using Breakpoints *(on page 1849)*

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 1835)*. 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.

Figure 455. Context node view



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 Developer supports you with syntax highlight and content completion assistance *(on page 634)*.

Figure 456. XPath Watch View

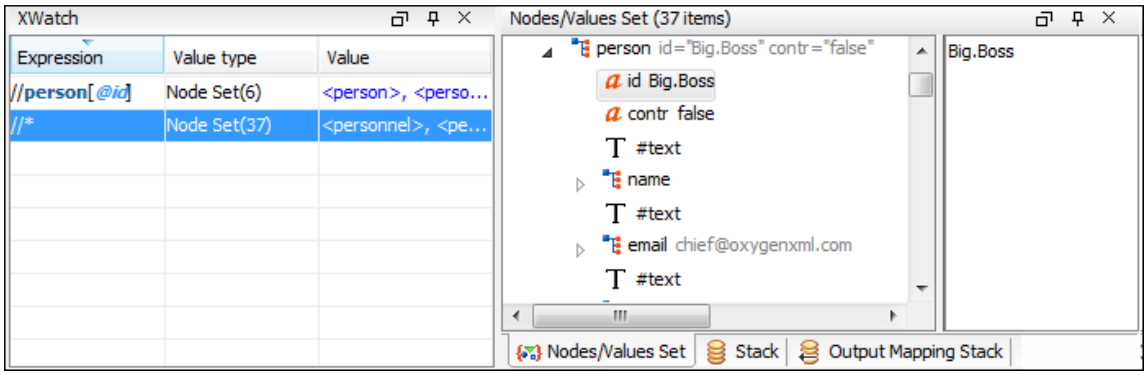


Table 43. XWatch columns

Column	Description
Expression	XPath expression to be evaluated (XPath 1.0 or 2.0 / 3.0 compliant).
Value	Result of XPath expression evaluation. Value has a type (see the possible values *(on page 1843)* in the Variables View *(on page 1843)*) section. For *Node Set* results, the number of nodes in the set is shown in parenthesis.

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 1842)*.
- 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 457. Messages View

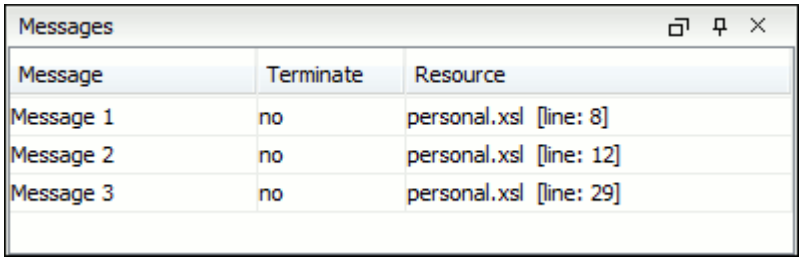


Table 44. Messages columns

Column	Description
Message	Message content.
Terminate	Signals whether or not the processor terminates the transformation once it encounters the message (yes/no respectively).
Resource	Resource file where *xsl:message* instruction is defined and the message line number. The complete path of the resource is available as tooltip.

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:

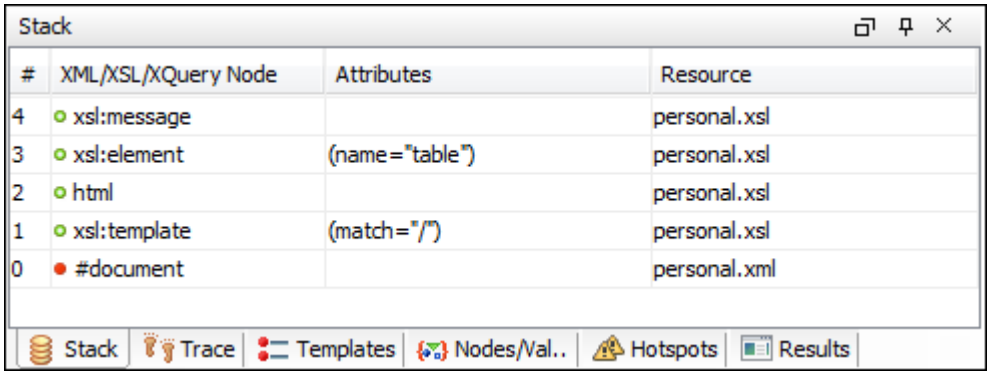
- Clicking a record from the table highlights the `xsl:message` declaration line.
- Message table values can be sorted by clicking the corresponding column header. Clicking the column header switches the sorting order between: ascending, descending, no sort.

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 Developer 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 458. 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 45. Stack Columns

Column	Description
#	Order number, represents the depth of the node (0 is the stack base).
XML/XSLT/XQuery Node	Node from source or stylesheet document currently being processed. One particular stack node is the document root, noted as **#document**.
Attributes	Attributes of the node (a list of *id="value"* pairs).
Resource	Resource file where the node is located. The entire path is available as tooltip.

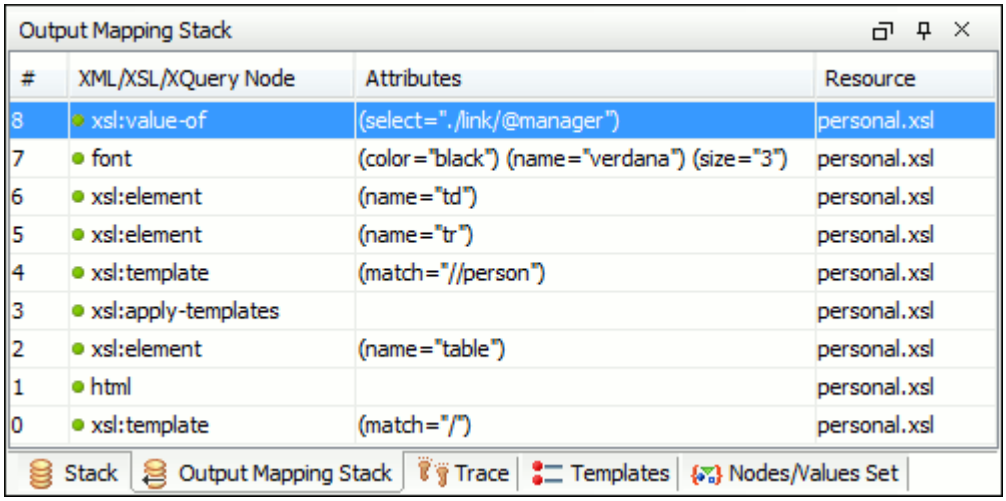
Important:
Remarks:

- 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 1846)* 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.

Figure 459. Output Mapping Stack view



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 46. Output Mapping Stack Columns

Column	Description
#	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.
XSL/XQuery Node	The name of an XSLT template/XQuery element that participated in the generation of the selected output area.
Attributes	The attributes of the XSLT template/XQuery node.
Resource	The name of the file containing the XSLT template/XQuery element.

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.

Related Information:

Identify the XSLT / XQuery Expression that Generated Particular Output *(on page 1846)*
Stack View *(on page 1837)*
Trace View *(on page 1840)*
Templates View *(on page 1841)*

Trace View

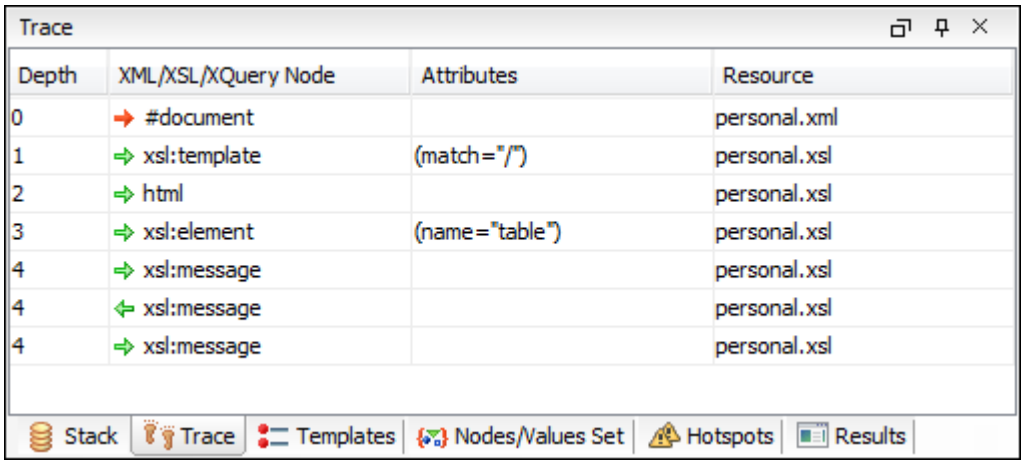
Usually, the XSLT/XQuery processors signal the following events during transformation:

- - 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 460. 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 47. Trace History Columns

Column	Description
Depth	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.
XML/XSLT/XQuery Node	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).
Attributes	Attributes of the node (a list of *id="value"* pairs).
Resource	Resource file where the node is located. The complete path of the resource file is provided as tooltip.

Important:
Remarks:

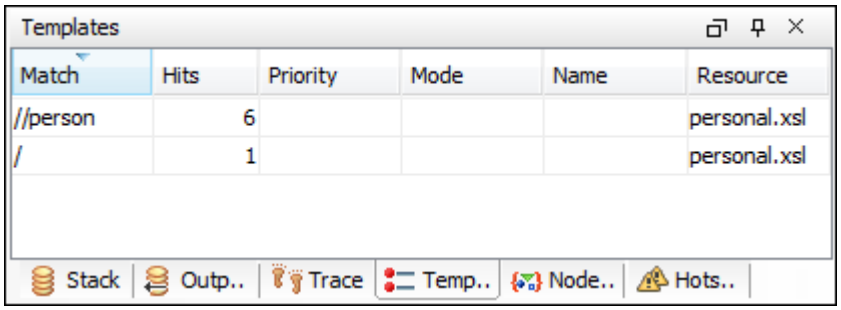
- Clicking a record highlights that node's location inside the resource.
- Only the Saxon processor shows the element attributes.
- The Xalan processor shows also the built-in rules.

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 461. 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 48. Templates columns

Column	Description
Match	The *match* attribute of the *xsl:template*.
Hits	The number of hits for the *xsl:template*. Shows how many times the XSLT processor used this particular template.
Priority	The template priority as established by XSLT processor.
Mode	The *mode* attribute of the *xsl:template*.
Name	The *name* attribute of the *xsl:template*.
Resource	The resource file where the template is located. The complete path of the resource file is available as tooltip.

Important:
Remarks:

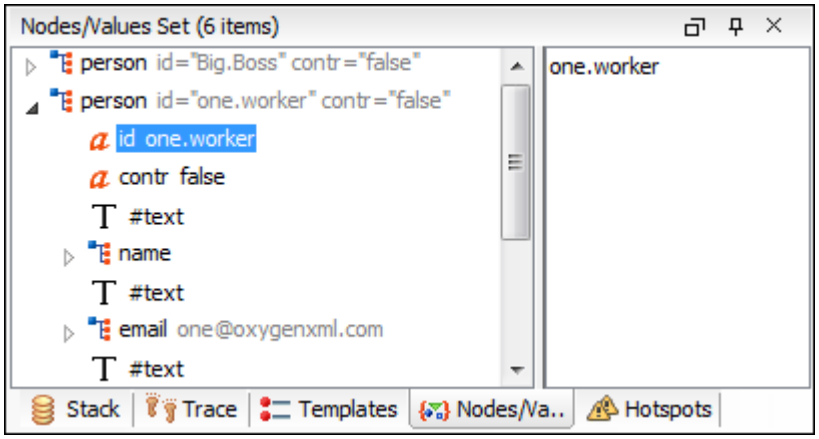
- Clicking a record highlights that template definition inside the resource.
- Saxon only shows the applied templates having at least one hit from the processor. Xalan shows all defined templates, with or without hits.
- Template table values can be sorted by clicking the corresponding column header. 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 1843)* and **XWatch** view *(on page 1835)*. 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 1843)* or **XWatch** view *(on page 1835)*.
- You click a tree fragment in the **Variables** *(on page 1843)* or **XWatch** view *(on page 1835)*.
- You click an XPath expression evaluated to a node set in the **Variables** *(on page 1843)* or **XWatch** view *(on page 1835)*.

Figure 462. Node Set view



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 Developer uses the following icons to differentiate variables and parameters:

- - Global variable.
- - Local variable.
- - Global parameter.
- - 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 463. Variables View

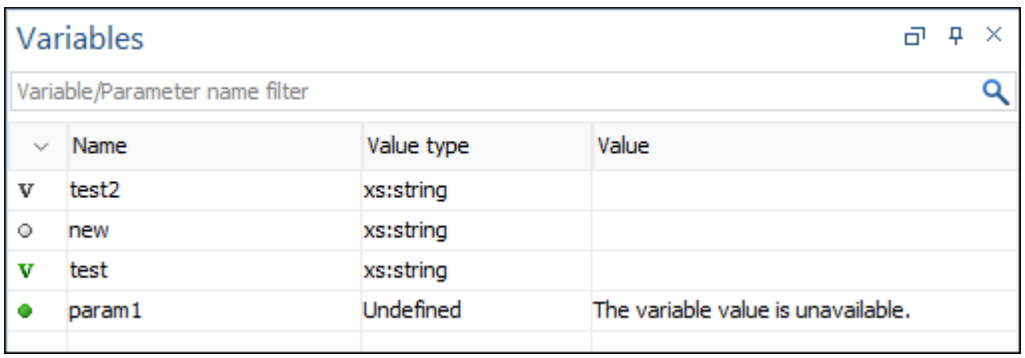


Table 49. Variables Columns

Column	Description
Name	Name of variable / parameter.
Value Type	Type of variable/parameter.

Table 49. Variables Columns (continued)

Column	Description
Value	Current value of variable / parameter.

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 *(on page 1842)* 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 *(on page 313)* and the XSLT/XQuery document. *(on page 313)*
2. If you are in the **Editor** *perspective (on page 2307)*, switch to the **XSLT Debugger** or **XQuery Debugger** *perspective (on page 2307)* 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 2307)* 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 1829)*. 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 1829)*.
5. Set XSLT/XQuery parameters using the **Configure parameters** button on the Control toolbar *(on page 1829)*.
6. Set one or more breakpoints *(on page 1849)*.
7. Step through the stylesheet using the following buttons available on the Control toolbar *(on page 1830)*:
   - **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 1846)*.

Related Information:
Identify the XSLT / XQuery Expression that Generated Particular Output *(on page 1846)*

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 2307)* 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 2307)* 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 1829)*. 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 1829)*.
4. Select the appropriate engine in the XSLT/XQuery engine selector of the Control toolbar *(on page 1830)*.
5. Set XSLT/XQuery parameters using the **Configure parameters** button on the Control toolbar *(on page 1829)*.
6. Apply the XSLT stylesheet or XQuery transformation using the **Run to end** button that is available on the Control toolbar *(on page 1831)*.
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 274)*.

Figure 464. XHTML Output to Source Mapping

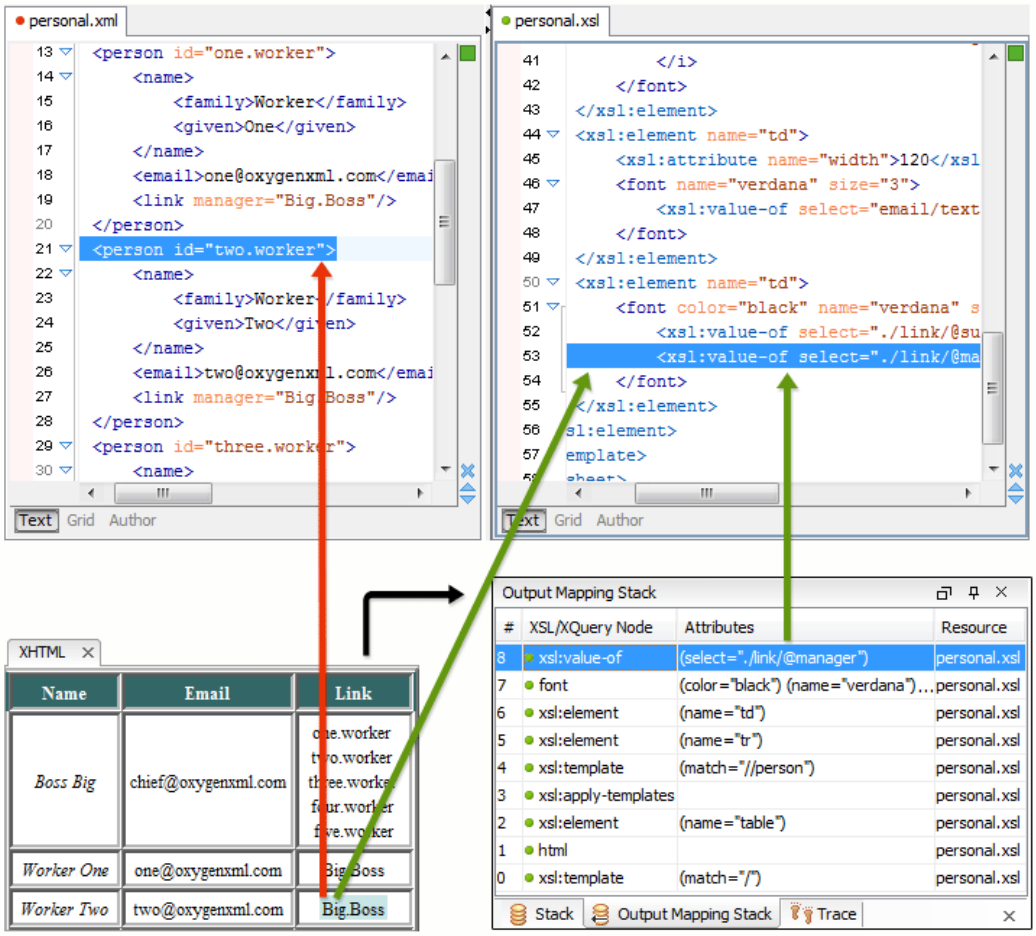
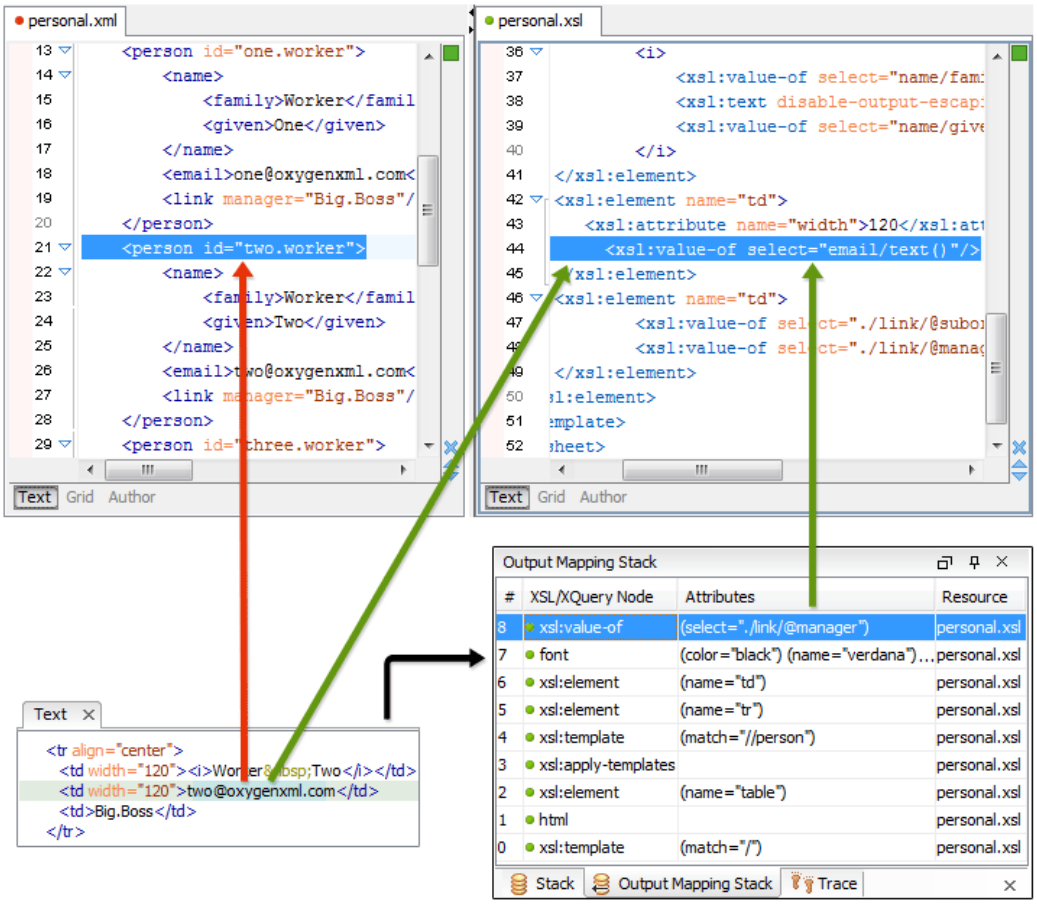


Figure 465. Text Output to Source Mapping



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 1838)*.

Related Information:

Output Mapping Stack View *(on page 1838)*

Trace View *(on page 1840)*

Templates View *(on page 1841)*

Using Breakpoints

The Oxygen XML Developer 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

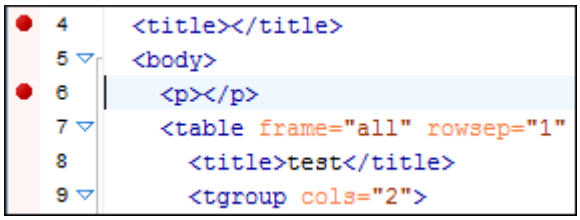
To insert a *breakpoint*, follow these steps:

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 466. Example: Breakpoints



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 **Renove** or **Remove all**.

Related Information:

Breakpoints View *(on page 1833)*

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 2307)*.

Enabling the Profiler

Enabling and disabling the profiler is controlled by the **Profiler** button from the debugger Control toolbar *(on page 1829)*. 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 1845)*.

Profiling Information Views

Immediately after enabling the profiler, two new information views are added to the current debugger information views *(on page 1832)*:

- **Invocation tree** view *(on page 1852)* on left side
- **Hotspots** view *(on page 1853)* on right side

Profiling data is available only after the transformation ends successfully.

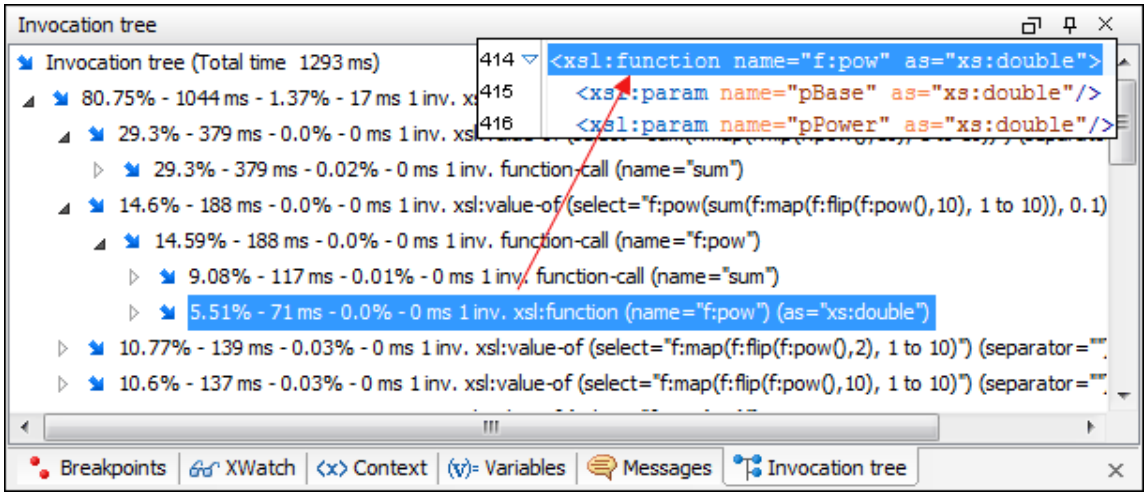
On the left side (**Invocation tree** view *(on page 1852)*), you can examine how style instructions are processed. This result view is also named call-tree, as it represents the order of style processing. The profiling result shows the duration time for each of the style-instruction including the time needed for its called children.

On the right side (**Hotspots** view *(on page 1853)*), 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 1852)* or **Hotspots** view *(on page 1853)*, you can use the backmapping feature to find the XSLT stylesheet or XQuery expression definition. Clicking the selected item causes Oxygen XML Developer to highlight the XSLT stylesheet or XQuery expression source line where the instruction is defined.

Figure 467. 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 1852)* or **Hotspots** view *(on page 1853)*, 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 Developer 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 189)*, 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.

Resources

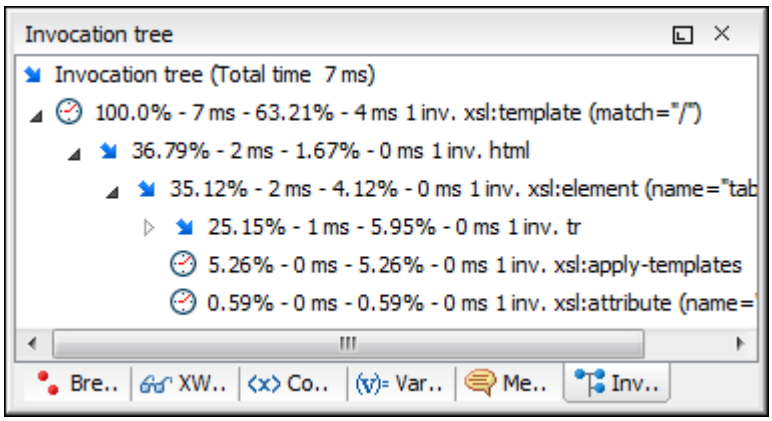
For more information about the XSLT/XQuery Profiler, watch our video demonstration:

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 468. 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 189)*:

- 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.

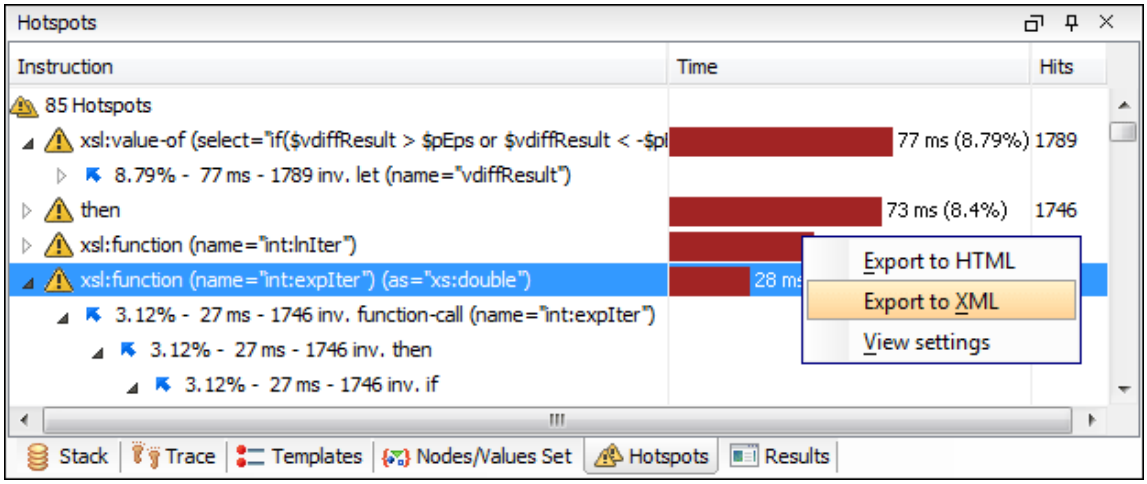
View settings

Opens the XSLT/XQuery Profiler preferences page *(on page 189)* 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 189)*. If the view is not displayed, it can be opened by selecting it from the **Window > Show View** menu.

Figure 469. Hotspots View



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 189)*:

- 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_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 189)* 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 on XML scenario *(on page 1159)* for your XSLT document (select **Configure Transformation Scenario(s)** action from the toolbar, then click **New**, and select **XSLT transformation on XML**).
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 1845)* by specifying the Java extensions using the **Edit extensions** button on the debugger control toolbar *(on page 1829)*.

Related Information:
Validating XSLT Stylesheets that Call Java Extensions *(on page 629)*

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 1849)* 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. 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 1829)* of the XSLT/XQuery transformation in the debugging *perspective (on page 2307)*.

    b. 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 1828)*:

- **Saxon 12.9 HE (Home Edition)** - a limited version of the Saxon 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 12.9 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 12.9 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** (Deprecated) - capable of running only XSLT 1.0 transformations, available only in the XSLT debugger.

17.

Extending Oxygen With the SDK

Oxygen XML Developer has an SDK that can be used as a base to develop *frameworks (on page 2305)* and *plugins (on page 2307)*. 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 Developer and ONLY if such extensions are used from inside Oxygen XML Developer. To use such extensions outside of Oxygen XML Developer (for example, a 3rd-party application that has Oxygen XML Developer built in to it), an additional license must be purchased to use the SDK according the **Oxygen XML SDK Licensing Policy** .

Extending Oxygen XML Developer with Plugins

A *plugin (on page 2307)* 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 Developer. 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 Developer Eclipse plugin you can look at the Eclipse IDE Integration Sample Project to see how an Eclipse plugin can interact with the Oxygen XML Developer APIs.

General Configuration of an Oxygen XML Developer Plugin

The Oxygen XML Developer functionality can be extended with *plugins (on page 2307)* that implement a clearly specified API. On the Oxygen XML Developer 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. The allowed format is: `MAJOR.MINOR.PATCH` (for example, `1.0.2`).
- **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 Developer `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 Developer `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 Developer 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 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">` - 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 *${oxygenInstallDir}* editor variable *(on page 259)* as part of the value of the `@name` attribute. You can also use a system variable (`${system(var.name)}`) or environment variable (`${env(VAR_NAME)}`).

- `<librariesFolder name="path/libraryFolderPath">` - 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).

Dependency Injection for Plugins

If you want to share a single instance of a certain class between plugin extensions and custom operations (to prevent instances from being repeated), you can declare a `<context>` element in your `plugin.xml` file:

```
<context class="my.package.ContextClass"/>
```

Important:
The `my.package.ContextClass` class should have a no-arguments constructor that will be called when the class is instantiated.

This will result in an instance being automatically generated. You can access this instance in an extension class by defining a field of that type and annotated with the `ro.sync.exml.plugin.PluginContext` annotation:

```
@PluginContext

ContextClass contextInstance;
```

The defined field is automatically populated with the single instance.

Tip:
By default, an instance of the `PluginDescriptor` class is also injectable.

Installing an Oxygen XML Developer Plugin

Choose one of the following methods to install a *plugin (on page 2307)* in Oxygen XML Developer:

Manual Method

To manually install a *plugin* in Oxygen XML Developer, follow these steps:

1. Go to the Oxygen XML Developer installation directory and locate the `plugins` directory.

   Note:
   The `plugins` directory contains all the *plugins* available to Oxygen XML Developer.

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 Developer.

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 Developer 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.

DANGER:
Installing from update sites other than the official **Oxygen** sites may cause security risks. Please verify that all your installed third-party add-ons come from companies or individuals that you have verified and that you implicitly trust. As a way to help with identifying possible add-on security problems, Oxygen XML Developer issues warnings for unsigned plugins.

Note:
All add-ons are installed in the `extensions` directory inside the Oxygen XML Developer preferences directory *(on page 80)*.

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 2307)* 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 2307)* 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 requires 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 *WorkspaceAccessPluginExtension* interface. The callback method *applicationStarted* of this interface allows access to a parameter of the *StandalonePluginWorkspace* type (allows for API access to the application workspace).

Important:
For security reasons, if a plugin's *applicationStarted* callback delays the startup of the entire application by more than 15 seconds, it will continue to be loaded on a separate thread and may no longer properly contribute UI customizations.

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 version="1.0" encoding="UTF-8"?>
<!DOCTYPE plugin SYSTEM "../plugin.dtd">
<plugin name="CustomWorkspaceAccess" ..............>
 <runtime>
  ........
 </runtime>

 <extension type="WorkspaceAccess" ...............>
 ...............
  <toolbar id="SampleID" initialSide="NORTH" initialRow="1"/>
</plugin>
```

The `<toolbar>` element adds a toolbar in the Oxygen XML Developer 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 version="1.0" encoding="UTF-8"?>
<!DOCTYPE plugin SYSTEM "../plugin.dtd">
<plugin name="CustomWorkspaceAccess" ..............>
 <runtime>
  ........
 </runtime>
```

```
<extension type="WorkspaceAccess" ...../>
.....
<view id="SampleID" initialSide="WEST" initialRow="0"/>
</plugin>
```

The `<view>` element adds a view in the Oxygen XML Developer 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 Developer, the **Project** view *(on page 334)* has an initial row of `0` and the **Outline** view *(on page 465)* 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. 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*).

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.
- 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.

Example: Adding a Custom View in Oxygen XML Developer

To add a custom view in Oxygen XML Developer, 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 1861)* implementation, where the `applicationStarted` callback is received, add a view component customizer like this:

   ```
   pluginWorkspaceAccess.addViewComponentCustomizer(new ViewComponentCustomizer() {
    public void customizeView(ViewInfo viewInfo) {
      if(
        //The view ID defined in the "plugin.xml"
        "SampleWorkspaceAccessID".equals(viewInfo.getViewID())) {
        cmsMessagesArea = new JTextArea("CMS Session History:");
        viewInfo.setComponent(new JScrollPane(cmsMessagesArea));
        viewInfo.setTitle("CMS Messages");
        viewInfo.setIcon(Icons.getIcon(Icons.CMS_MESSAGES_CUSTOM_VIEW_STRING));
      }
    }
   });
   ```

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 2307)* 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 1861)*, 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:

```
<!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 1865)*) 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:

```
function applicationStarted(pluginWorkspaceAccess) {
.........
}

function applicationClosing(pluginWorkspaceAccess) {
.........
}
```

With regard to the `applicationStarted` callback, besides using the *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.

Important:
For security reasons, if a plugin's *applicationStarted* callback delays the startup of the entire application by more than 15 seconds, it will continue to be loaded on a separate thread and may no longer properly contribute UI customizations.

Below is a much larger example with a JavaScript Workspace Access *plugin* extension implementation that adds a new action in the contextual menu. The action starts the `notepad.exe` application and passes the reference to the currently selected `<topicref>` to it.

```
function applicationStarted(pluginWorkspaceAccess) {
 Packages.java.lang.System.err.println("Application started "
      + pluginWorkspaceAccess);
 edChangedListener = {
  /*Called when a DITA Map is opened*/
   editorOpened: function (editorLocation) {
   Packages.java.lang.System.err.println("\nrunning " + editorLocation);
  /*Get the opened DITA Map*/
   editor = pluginWorkspaceAccess.getEditorAccess(editorLocation,
   Packages.ro.sync.exml.workspace.api.PluginWorkspace.DITA_MAPS_EDITING_AREA);
   ditaMapPage = editor.getCurrentPage();
  /*Add listener called when right-click is done in the DITA Maps manager*/
   customizerObj = {
   customizePopUpMenu: function (popUp, ditaMapDocumentController) {
   Packages.java.lang.System.err.println("RIGHT CLICK" + popUp);
   tree = ditaMapPage.getDITAMapTreeComponent();
  /*Selected tree path*/
   sel = tree.getSelectionPath();
   if (sel != null) {
   selectedElement = sel.getLastPathComponent();
  /*Reference attribute*/
   href = selectedElement.getAttribute("href");
   if (href != null) {
     try {
  /*Create absolute reference*/
   absoluteRef = new Packages.java.net.URL(selectedElement.getXMLBaseURL(),
           href.getValue());
   Packages.java.lang.System.err.println("Computed absolute reference "
         + absoluteRef);
   mi = new Packages.javax.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 (e1) {
       Packages.java.lang.System.err.println(e1);
      }
     }
    }
   }

   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:

```
<!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 2307)* can be used by developers to automatically allow or reject remote connections that Oxygen XML Developer would normally ask the user for confirmation.

The name of the *plugin* extension is ***TrustedHosts***. For security reasons, Oxygen XML Developer 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 this 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:

  ```
  <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 437)*.

Example implementation:

```
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;
        }
}
```

Additional Framework Plugin Extension

This type of *plugin (on page 2307)* 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:

```
<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 73)*, the new framework will be set as read-only and editing it will only be possible if you *duplicate* it *(on page 94)*. 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 2307)* 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 1189)*. 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:

```
<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 2307)* 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 Developer:

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 Developer 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 2307)* 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 Developer 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 2307)* allows you to contribute new translation languages to the Oxygen XML Developer UI.

The ***AdditionalUITranslation*** *plugin* extension provides the ability to contribute new translation languages to the interface in Oxygen XML Developer.

A sample `plugin.xml` file looks like this:

```
<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:

```
<translation>
    <languageList>
        <language description="Romanian" lang="ro_RO" localeDescription="Romana"/>
    </languageList>
    <key value="Error">
        <val lang="ro_RO">Eroare</val>
    </key>
.........................
</translation>
```

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 81)* 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 2307)* allows you to contribute an external DITA-OT distribution to Oxygen XML Developer.

Oxygen XML Developer comes bundled with DITA-OT version 4.4.0. 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 Developer.

Example

For instance, if you wanted to use a DITA-OT version 1.8, your `plugin.xml` file might look like this:

```
<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 Developer 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 200)*.

Custom Protocol Plugin Extension

This type of *plugin (on page 2307)* 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 1861)* 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 Developer.

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 button.

Lock Handler Plugin Extension

This type of *plugin extension (on page 2307)* 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 version="1.0" encoding="UTF-8"?>
<!DOCTYPE plugin SYSTEM "../plugin.dtd">
<plugin name="CustomLockHandler" ..............>
 <runtime>
  ........
 </runtime>

 <extension type="LockHandlerFactory"
        class="LockHandlerFactoryPluginExtensionImpl"/>
 ...............
```

```
</plugin>
```

Open Redirect Plugin Extension

This type of *plugin (on page 2307)* 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 1735)* 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 Developer 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 2307)* allows you to add custom **Preferences** pages.

The extension must implement the *ro.sync.exml.plugin.option.OptionPagePluginExtension* class. The provided callbacks allows 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:

```
<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, perform these steps:

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 Developer instance *(on page 1860)*.
3. In the **Project** view *(on page 334)*, create a project or open an existing one.
4. Open the **Preferences** dialog box **(Options > Preferences)** *(on page 79)*.
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 2308)* 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.

Option Page Group Plugin Extension

This type of *plugin extension (on page 2307)* allows you to add a group of custom **Preferences** pages from a plugin.

The extension must implement the *ro.sync.exml.plugin.OptionsPageGroupPluginExtension* class. The base method `OptionsPageGroupPluginExtension.addOptionPagePluginExtension(...)` allows adding multiple implementations of the `OptionPagePluginExtension` *(on page 1875)* base class.

All preferences pages that are contributed by this extension are listed as descendents of the plugin specific preferences page in the **Preferences** dialog box in the **Plugins** category.

The `plugin.xml` configuration file can specify one or more such extensions using constructs like this:

```
<extension type="OptionPageGroup" class="my.pack.CustomOptionPageGroupPluginExtension"/>
```

Resource Locking Custom Protocol Plugin Extension

This type of *plugin (on page 2307)* 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 Developer.

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 Developer.

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 `sftp`, `http`, `https`, or `customName`).

Targeted URL Stream Handler Plugin Extension

This type of *plugin (on page 2307)* 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`, or `sftp`. Oxygen XML Developer 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 Developer *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 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 Developer *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 Developer) 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 2307)* 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 Developer 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 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 2307)* 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 1102)* 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:

  ```
  <extension type="Transformer" class="my.xslt.plugin.extension"/>
  ```

Validator Plugin Extension

This type of *plugin (on page 2307)* 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:

```
<extension type="DocumentValidator" class="my.validator.plugin.extension"/>
```

Saxon XSLT Transformer Plugin Extension

This type of *plugin (on page 2307)* 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 1102)* 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:

```
<extension type="Transformer" class="my.saxon.xslt.plugin.extension"/>
```

An add-on that implements the Saxon XSLT transformer can be found here: Saxon Transformer Add-on. For more information, see the Oxygen XML SDK Add-on Repositories web page.

XQuery Transformer Plugin Extension

This type of *plugin (on page 2307)* 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 1190)* 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:

```
<extension type="XQueryTransformer" class="my.xquery.plugin.extension"/>
```

Saxon XQuery Transformer Plugin Extension

This type of *plugin (on page 2307)* 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 1190)* 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 Transformer add-on. 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 2307)* 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 1861)* 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 2307)* 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 Developer.

Selection Plugin Extension

This type of *plugin (on page 2307)* 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 252)* 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 Developer.
- `getDocumentURL()` - Returns the URL of the currently edited document.

Related information
Editor Variables *(on page 252)*

Example - Uppercase Plugin

The following *plugin (on page 2307)* is called **UppercasePlugin** and is an example of a Selection *plugin*. *(on page 1882)*. It is used in Oxygen XML Developer for capitalizing the characters in the current selection. This example consists of two Java classes and the *plugin* descriptor file (`plugin.xml`):

- **UppercasePlugin.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**:

```
<!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 2307)* 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 Developer.

How to Write a CMS Integration Plugin

To have a complete integration between Oxygen XML Developer and a CMS, you usually have to write a *plugin (on page 2307)* that combines the following two available *plugin* extensions:

- Workspace Access *(on page 1861)*
- Custom protocol *(on page 1890)*

The usual set of requirements for an integration between Oxygen XML Developer and the CMS are as follows:

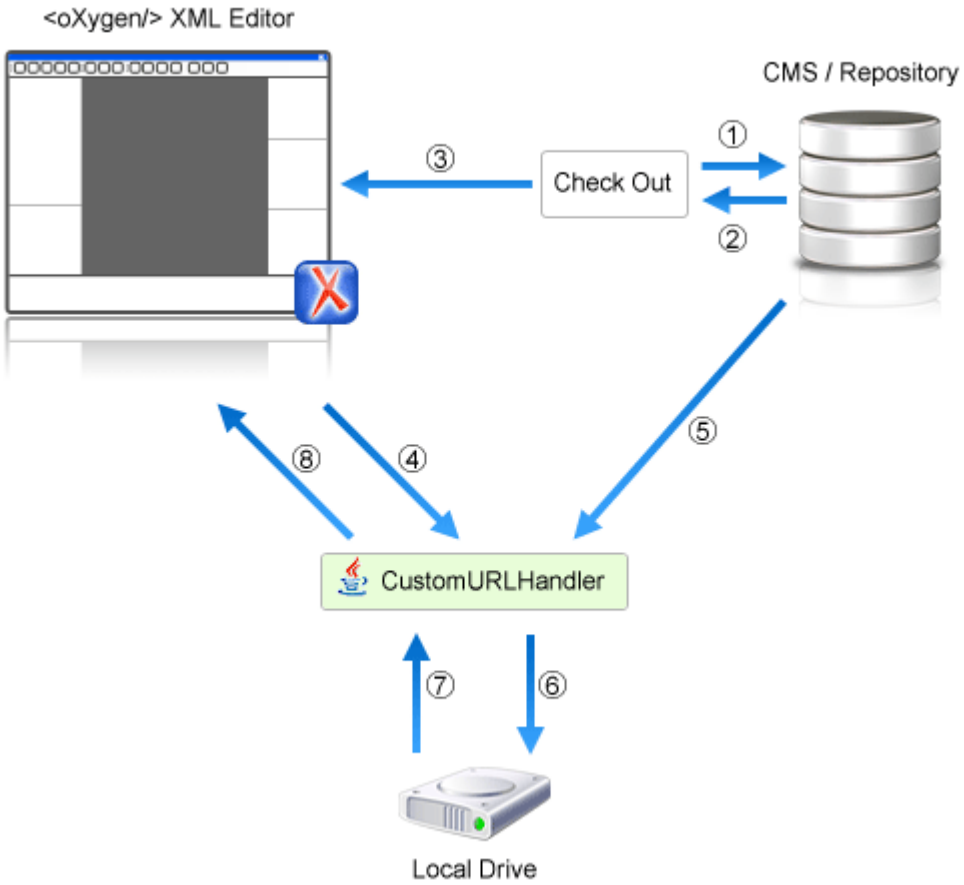
1. Contribute to the Oxygen XML Developer toolbars and main menu with your custom **Check Out** and **Check In** actions:
   - **Check Out** triggers your custom dialog boxes that allow you to browse the remote CMS and choose the resources you want to open.
   - **Check In** allows you to send the modified 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 Developer 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 Developer. 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:

```
  /**
   * 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 470. 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 Developer where a URL can be chosen to perform a special action (such as the **Reuse Content** or **Insert Image** action). Sample code:

```
      //Add an additional browse action to all dialog boxes/places
      //where Oxygen allows selecting a URL.
      pluginWorkspaceAccess.addInputURLChooserCustomizer
(new InputURLChooserCustomizer() {
        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 Developer, 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:

```
      //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.
          return null;
        }
      });
...
```

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:

```
<!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 2306)* archive containing your implementation classes.
6. Copy your new *plugin* directory in the `plugins` subfolder of the Oxygen XML Developer install folder (for example, *`[OXYGEN_INSTALL_DIR]`*`/plugins/myPlugin`) and start Oxygen XML Developer.

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 Developer. To instruct the *plugin* to prefer its libraries over the ones used by Oxygen XML Developer, you can add the following attribute on the `<plugin>` root element: `classLoaderType="preferReferencedResources"` from the `plugin.xml` descriptor file.

A **Late Delegation Class Loader** (the main class loader in Oxygen XML Developer) 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 Developer 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 Developer 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 Developer `LateDelegationClassLoader`.

If you experience additional problems, such as:

```
java.lang.LinkageError: ClassCastException:

attempting to cast

jar:file:/C:/jdk1.6.0_06/jre/lib/rt.jar!/

javax/xml/ws/spi/Provider.class

tojar:file:/D:/Program

 Files/Oxygen XML Editor

 12/plugins/wspcaccess/../../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:

```
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 2307)*, 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 Developer 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 (on page 2306)* 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 Developer both through a *framework (on page 2305)* and through a *plugin (on page 2307)*. 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 79)*, 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 1858)*.

If you have created the framework using a *Framework Extension Script*, then edit the script and specify the plugin ID on the classpath.

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 Developer, 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 2307)* package or those who want to automate some of the steps:

1. Pack the *plugin (on page 2307)* as a ZIP file or a *Java Archive (on page 2306)*. 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 2303)*, 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 Developer 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*.

Deploying an Add-on

To deploy an add-on, copy the `ZIP` or *Java Archive (on page 2306)* 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

Testing Plugins and Java Extensions

In the various procedures for creating a *plugin (on page 2307)*, 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. Close the Oxygen XML Developer application if it is running.
2. Create in the folder *[OXYGEN_INSTALL_DIR]*`/plugins/` a subfolder named for example `myPlugin` (but it can have any other name).
3. In the `myPlugin` subfolder create a new text file named `plugin.redirect` that contains the full file path references to your project (for example, `C:\Users\john_doe\Documents\sample-plugin-folder`).

   Step Result: Oxygen XML Developer will automatically load the plugin from your project location when it starts.
4. 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 Developer and test your changes.

Creating and Running Automated Tests

If you have developed complex custom *plugins (on page 2307)* or *frameworks (on page 2305)* (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 Developer standalone installation includes a main `oxygen.jar` library located in the *[OXYGEN_INSTALL_DIR]*. That *JAR (on page 2306)* 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 under the `oxygen` sub-directory.
2. Add all *JAR* libraries present in the `./oxygen/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. According to step 1, it should be `oxygen`.
    - **File frameworksFolder** - The file path to the `frameworks` directory. It can point to a custom *framework* directory where it resides. According to step 1, it should be `oxygen/frameworks`.
    - **File pluginsFolder** - The file path to the `plugins` directory. It can point to a custom *plugin* directory where it resides. According to step 1, it should be `oxygen/plugins`.
    - **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`, or `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:

```
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" +
    "\n" +
    "Date=09-04-2012\n" +
    "\n" +
    "Trial=31\n" +
```

```
    "\n" +
    "SGN=MCwCFGNoEGJSeiC3XCYIyalvjzHhGhhqAhRNRDpEu8RIWb8icCJO7HqfVP4++A\\=\\=\n" +
    "\n" +
"-------END-LICENSE-KEY-------",
   PluginWorkspaceTCBase.XML_AUTHOR_PRODUCT);
}


/**
  * <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\"?>\n" +
        "<!DOCTYPE task PUBLIC \"-//OASIS//DTD DITA Task//EN\" \"task.dtd\">\n" +
        "<task id=\"taskId\">\n" +
        "    <title>Task <b>title</b></title>\n" +
        "    <prolog/>\n" +
        "    <taskbody>\n" +
        "        <context>\n" +
        "            <p>Context for the current task</p>\n" +
        "        </context>\n" +
        "        <steps>\n" +
        "            <step>\n" +
        "                <cmd>Task step.</cmd>\n" +
        "            </step>\n" +
        "        </steps>\n" +
        "    </taskbody>\n" +
        "</task>\n" +
        "", getCurrentEditorXMLContent());
  }
}
```

Debugging a Plugin Using IntelliJ IDEA

To use *IntelliJ IDEA* to debug problems in the code of a *plugin (on page 2307)* without having to re-bundle the plugin's Java classes in a *JAR (on page 2306)* library, follow these steps:

1. Download and install Oxygen XML Developer.
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:

   ```
   <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 > Dependences**), 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
   -Dcom.oxygenxml.editor.plugins.dir=D:\projects\MyPluginProject
   ```

   Note:
   If you need to configure the *plugin* for Oxygen XML Developer, set the `com.oxygenxml.app.descriptor` to `ro.sync.exml.DeveloperFrameDescriptor`.
8. Add a *breakpoint (on page 1850)* in the source of one of your Java classes.
9. Debug the created configuration. When the code reaches your *breakpoint (on page 1849)*, 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 2307)* without having to re-bundle the plugin's Java classes in a *JAR (on page 2306)* library, follow these steps:

1. Download and install Oxygen XML Developer.
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 334)*), 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
   -Dcom.oxygenxml.editor.plugins.dir=D:\projects\MyPluginProject
   ```

   Note:
   If you need to configure the *plugin* for Oxygen XML Developer, set the `com.oxygenxml.app.descriptor` to `ro.sync.exml.DeveloperFrameDescriptor`.

   In the **Dependencies** tab, you should only add dependencies to two JAR libraries: *`[OXYGEN_INSTALL_DIR]`*`/lib/oxygen.jar` and *`[OXYGEN_INSTALL_DIR]`*`/lib/oxygen-basic-utilities.jar`.
8. Add a *breakpoint (on page 1850)* in the source of one of your Java classes.
9. Debug the created configuration. When the code reaches your *breakpoint (on page 1849)*, 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 2307)* without having to bundle its Java classes in a *JAR (on page 2306)* library, perform the following steps:

1. Download and install Oxygen XML Developer.
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 Developer from the *`[OXYGEN_INSTALL_DIR]`* and in the **Document Type Association** preferences page *(on page 93)*, edit the document type (for example, **DITA**) to open the **Document Type** configuration dialog box *(on page 95)*. 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
   ```
6. Add a *breakpoint (on page 1850)* in one of the source Java classes.
7. Debug the created configuration. When the code reaches your *breakpoint (on page 1849)*, the Eclipse IDE debugging perspective should take over.

Disabling a Plugin

To disable a *plugin (on page 2307)*, use one of the following two methods:

- Open the **Preferences** dialog box **(Options > Preferences)** *(on page 79)*, 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.

Developer Quick Start Guide

Oxygen XML Developer 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 1891)*. For more information, see the **The Oxygen SDK (Part 1: Plugins)** 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. 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.

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 Developer. For details, see: 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 Developer 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 1861)* but the *Workspace Access* plugin type *(on page 1861)* 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 1869)*.

The Maven-based Oxygen XML SDK 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.

The *Workspace Access* plugin API can also be used with a JavaScript-based plugin *(on page 1864)*. Small plugin samples can be found here: https://github.com/oxygenxml/wsaccess-javascript-sample-plugins.

You can create automated tests *(on page)* for your plugins and debug them using the Eclipse IDE *(on page 1895)*.

A plugin can either be installed manually or packed as an add-on and installed using **Help->Install new add-ons** *(on page 1860)*.

Sample Plugins

A sample Maven-based Workspace Access plugin can be found here: https://github.com/oxygenxml/sample-plugin-workspace-access.

There is also a sample project which contains various JavaScript-based plugins: https://github.com/oxygenxml/wsaccess-javascript-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.

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 Extension

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 extension *(on page 1861)* 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 1869)*.

Java or JavaScript?

Oxygen XML Developer 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:

- Workspace Access Plugin Extension *(on page 1861)*
- Workspace Access Plugin Extension (JavaScript-Based) *(on page 1864)*

API Overview

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 1900)*.
- Import or set global settings in Oxygen XML Developer. See: https://www.oxygenxml.com/InstData/Editor/SDK/javadoc/ro/sync/exml/workspace/api/options/GlobalOptionsStorage.html.

- 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 1861)* 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 1861)* type can contribute a new side view to Oxygen XML Developer. 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 Developer). 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 1875)* to the **Preferences** dialog box in Oxygen XML Developer. 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.
- **Set global options by importing an options XML file:** (https://www.oxygenxml.com/InstData/Editor/SDK/javadoc/ro/sync/exml/workspace/api/options/GlobalOptionsStorage.html#importGlobalOptions(java.io.File,boolean)) Such an options XML file can be generated by using the **Options->Export Global Options** action in Oxygen XML Developer. A sample JavaScript-based *WorkspaceAccess* plugin implementation that imports such an options XML document into the application can be found here: https://github.com/oxygenxml/wsaccess-javascript-sample-plugins/tree/master/impose-options.

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 1900)* 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 Developer.
- Your plugin can add a specific side view *(on page 1901)*.
- 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 266)* supported by Oxygen XML Developer. 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.

Comparing Documents

Using a Workspace Access plugin extension, you have access to various APIs that allow the comparison of XML documents:

- Display the **Compare Files** tool with a given set of URLs for two-way or three-way comparisons: https://www.oxygenxml.com/InstData/Editor/SDK/javadoc/ro/sync/exml/workspace/api/standalone/DiffAndMergeTools.html.
- Compare documents in-memory (for example, to generate reports): https://www.oxygenxml.com/InstData/Editor/SDK/javadoc/ro/sync/exml/workspace/api/util/CompareUtilAccess.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 Developer 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:

```
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

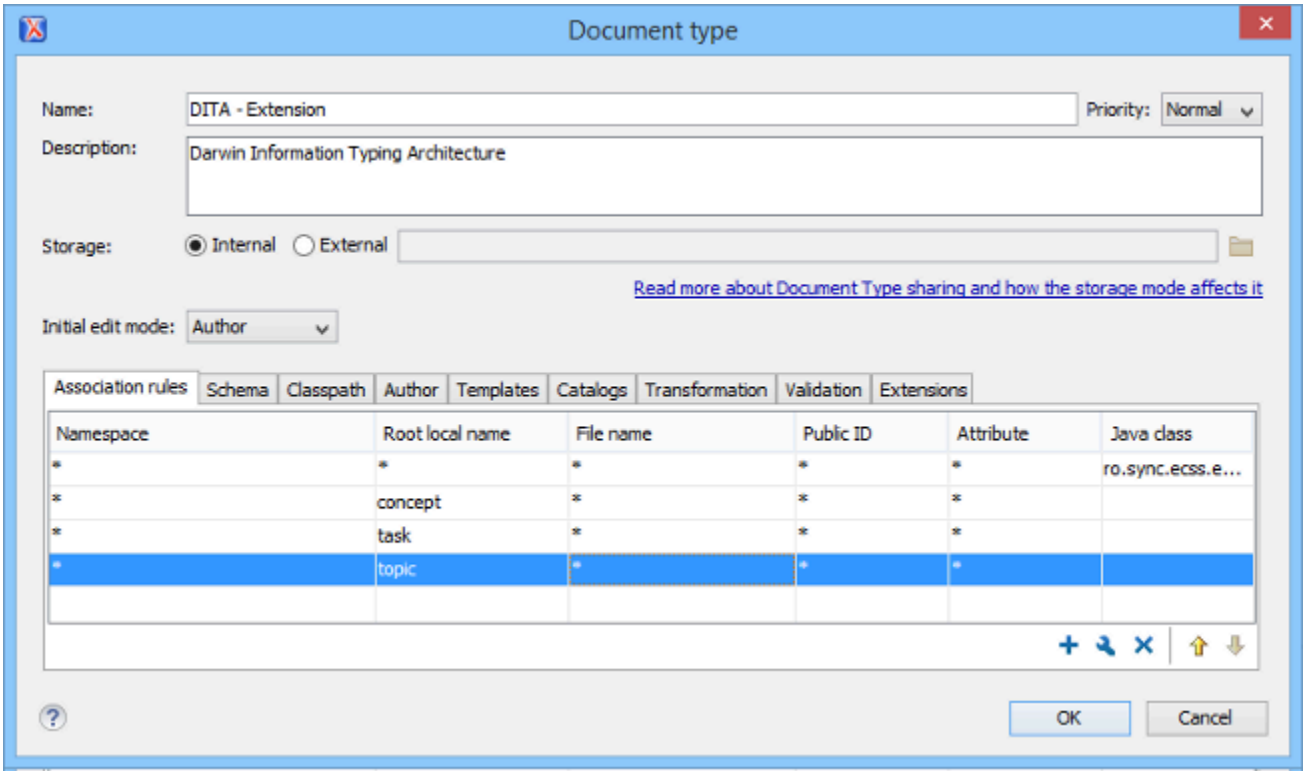
There is a particular plugin extension to contribute a new language to Oxygen XML Developer: https://www.oxygenxml.com/doc/ug-editor/topics/contribute-new-languages-extension.html.

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 Developer 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 Developer 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 1869)*.

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 Developer.
- 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 299)* (**File->New** or the **New** button on the toolbar) presents custom file templates gathered from all frameworks installed in Oxygen XML Developer. 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 Developer 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 1904)* 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 103)* to configure the name, description, icons, menu shortcuts, and various XPath-enabled activation operations *(on page 106)*.

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.

Related information

Sample DITA (framework) extension that sets a custom `ExtensionsBundle` implementation for customizing links

Customizing the Smart Paste Mapping

The *Smart Paste* feature in Oxygen XML Developer 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 2305)* and a *Plugin (on page 2307)* 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 Developer.

   For the Oxygen XML Developer Plugins API, Javadoc, samples, and documentation, go to https://www.oxygenxml.com/oxygen_sdk.html#Developer_Plugins

2. Create or modify the document type 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 93)* 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 95)*. The subtabs in this tab can be used to define custom DITA actions and add them to the toolbars, main menus, or contextual menus.

   If you look on disk in the `[OXYGEN_INSTALL_DIR]\frameworks\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 93)*. Then you can share that updated file with all users.

   The same folder contains some *JAR (on page 2306)* 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

SDK Common Use Cases

This section contains details for specific use cases regarding customizations using the *Oxygen SDK*, or plugins *(on page 1857)*.

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 *WSTextEditorPage.addPopUpMenuCustomizer* API method allows you to customize the contextual menu shown in **Text** mode.

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 2307)* for Oxygen XML Developer 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:

```
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);
```

Customize the Outline View in Text Mode

Use Case

You want to customize the **Outline** view *(on page 465)* in **Text** mode.

Solution

Suppose that you have the following XML document:

```
<doc startnumber="15">
    <sec counter="no">
        <info/>
        <title>Introduction</title>
      </sec>
    <sec>
        <title>Section title</title>
      <para>Content</para>
```

```
        <sec>
            <title>Section title</title>
                <para>Content</para>
        </sec>
    </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:

```
/**
 * 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;


  /**
   * Disable CaretListener when we select from the CaretListener.
   */
  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(
            //The view ID defined in the "plugin.xml"
            "SampleWorkspaceAccessID".equals(viewInfo.getViewID())) {
          customOutlineList = new JList();
          //Render the content in the Outline.
          customOutlineList.setCellRenderer(new DefaultListCellRenderer() {
       /**
        * @see javax.swing.DefaultListCellRenderer#getListCellRendererComponent
(javax.swing.JList, java.lang.Object, int, boolean, boolean)
        */
      @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() + "\"";
         }
       }
        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) {
   //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();
                }
                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.findElementsByXPath("//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;
  }
}
```

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 Developer.

Solution

The Oxygen XML Developer 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:

```
open edit-in-oxygen:protocol://host/path/file.xml
```

For example, if you start the following from the command line:

```
open edit-in-oxygen:http://www.oxygenxml.com/index.html
```

Oxygen XML Developer 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 315)*.

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:

```
     InputSource is = new org.xml.sax.InputSource
(URLUtil.correct(new File("test/personal.xsl")).toString());
     xslSrc = new SAXSource(is);
     javax.xml.transform.Transformer transformer =
authorAccess.getXMLUtilAccess().createXSLTTransformer
(xslSrc, 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: *ro.sync.exml.workspace.api.PluginWorkspace.getXMLUtilAccess()*.

Save a New Document with a Predefined File Name Pattern

Use Case

You want Oxygen XML Developer 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 2307)* for Oxygen XML Developer 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:

```
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) {
          final File chosenFile =
new File(chosenDir, UUID.randomUUID().toString() + ".dita");
        SwingUtilities.invokeLater(new Runnable() {
            @Override
              public void run() {
              try {
                saving = true;
                editor.saveAs(new URL(chosenFile.toURI().toASCIIString()));
              } catch (MalformedURLException e) {
                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);

.............................................
```

18.

Add-ons

Oxygen XML Developer offers various default add-ons that can be installed to provide additional functionality to Oxygen XML Developer. 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 Developer, see Oxygen XML Add-on Repositories.

Documentation and installation details for the latest versions of each default add-on.

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.

19.
Tools

Oxygen XML Developer 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 Developer 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 334)*.



Note:
The built-in refactoring operations are also available from the **Refactoring** submenu in the contextual menu of **Text** mode. This is useful because by selecting the operations from the contextual menu, Oxygen XML Developer 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 1925)* or previewed *(on page 1925)* also appear in the **Refactoring** submenu of the contextual menu in the **Project** view.

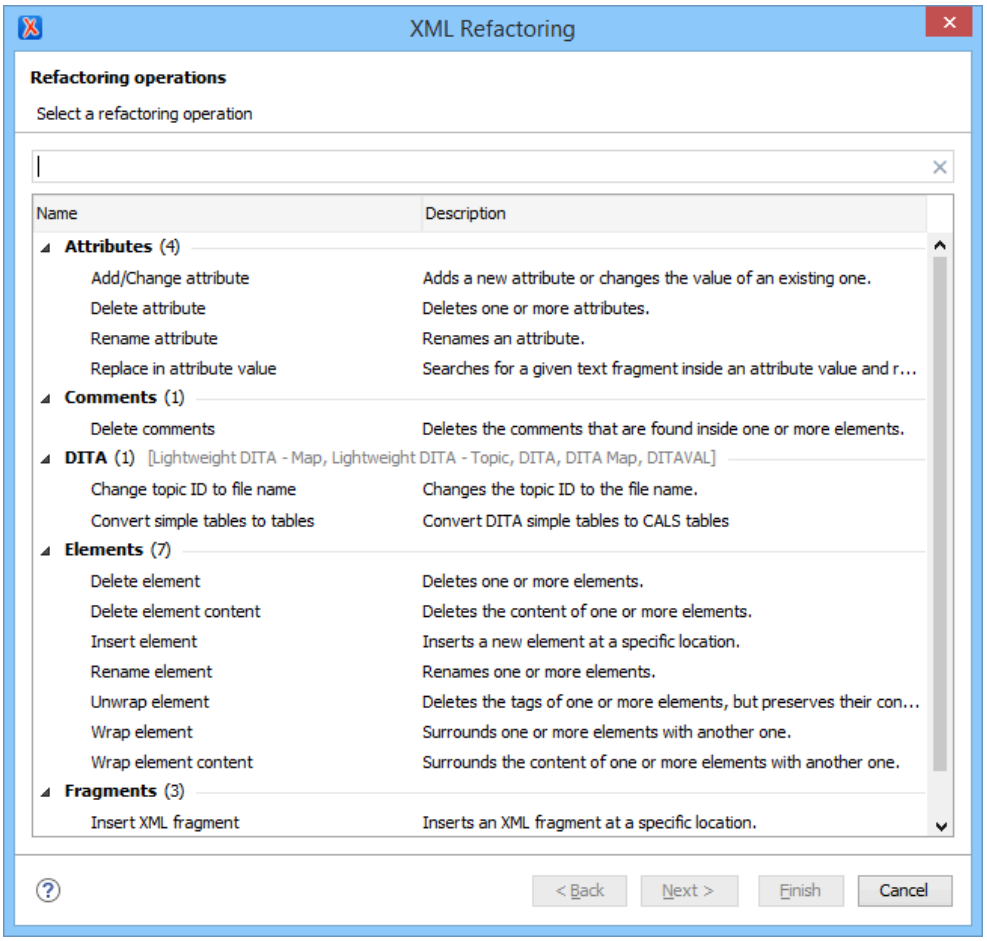
XML Refactoring Wizard

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.

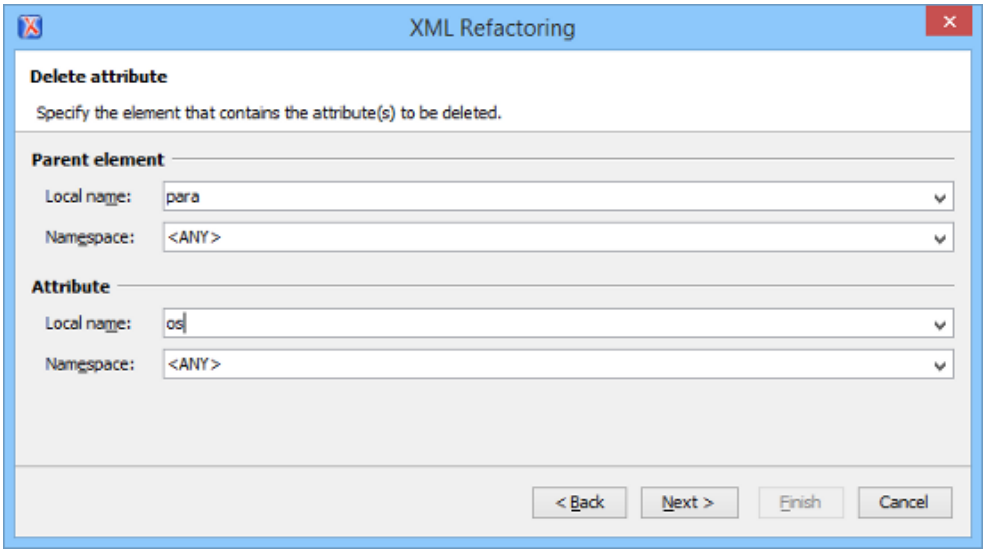
Figure 471. XML Refactoring Wizard



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.

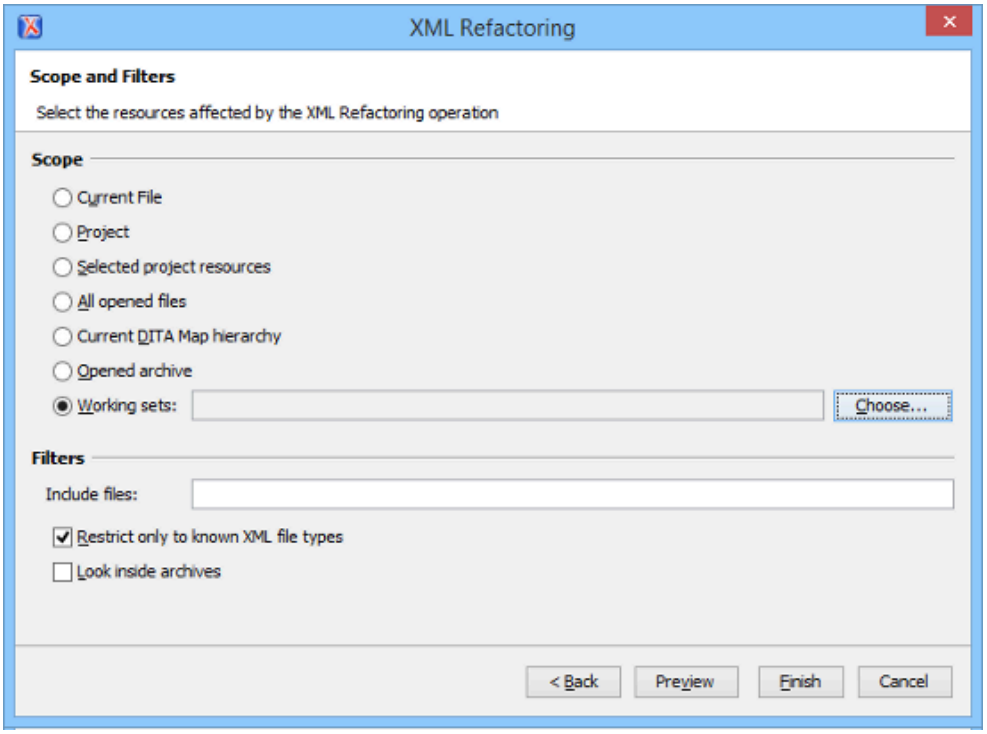
Figure 472. XML Refactoring 2nd Wizard Page (Delete Attribute Operation)



Scope and Filters

The last wizard page allows you to select the set of files that represent the input of the operation.

Figure 473. XML Refactoring - Scope and Filters Wizard Page



Scope section

You can specify the scope for the operation by selecting from predefined resource sets or you can define your own set of resources by creating a *working set (on page 2309)* (select **Working sets** and click the **Choose** button to the right). If you select **Project**, all files attached to the current project will be used for the scope of the operation. If you select **Current DITA Map hierarchy**, the current DITA map

that is open in the DITA Maps Manager along with all of its referenced topics and submaps (and topics referenced in those submaps) are used for the scope.

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 Developer 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 **Text** mode. The operations are also

grouped by category in this submenu. When selecting the operations from the contextual menu, Oxygen XML Developer 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.

Convert Nested Topics to New Topics

Use this operation on topics that contain nested `<topic>` elements to convert each nested topic to a new topic.

Convert Sections to New Topics

Use this operation on topics that contain multiple sections to convert each section to a new 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).

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).

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

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**, **Entire 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. The new value is modified and reflected in DITA maps, DITA topics, and DITAVAL files.

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**, **Entire 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

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 1287)* that are included in a custom Publishing Template that was created in Oxygen XML Developer version 20.0 or 20.1 so that they will be compatible with Oxygen XML Developer version 21.0.

Migrate HTML Page Layout Files to v22

Use this operation to convert custom HTML page layout files *(on page 1287)* that are included in a custom Publishing Template that was created in Oxygen XML Developer versions 20.0 - 21.1 so that they will be compatible with Oxygen XML Developer version 22.0.

Update HTML Pages



Attention:
This operation is only used by Oxygen XML Developer 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 190)* 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.

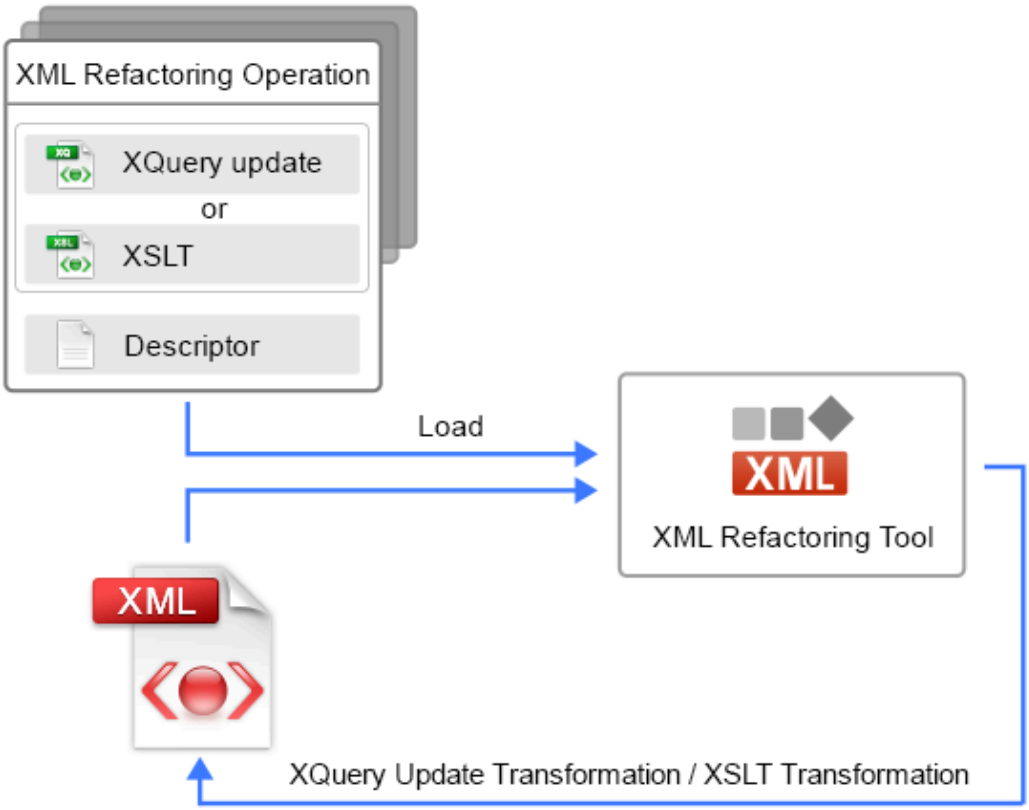
Custom Refactoring Operations

While Oxygen XML Developer 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 Developer 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 474. 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 579)*, along with the built-in operations.

After the user chooses an operation and specifies its parameters, Oxygen XML Developer 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 611)*.

  - 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 611)* 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 601)* or XSLT stylesheet file *(on page 606)*.
2. Create an XML refactoring operation descriptor file, that references the above script, as explained in these sections: Example descriptor file for an XQuery Update script *(on page 604)* or Example descriptor file for an XSLT stylesheet *(on page 609)*.
3. Store both files in one of the locations that Oxygen XML Developer *(on page 613)* 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 579)*.

Related information
Storing and Sharing Refactoring Operations *(on page 613)*

Custom Refactoring Script

The first step in creating a custom refactoring operation is to create an XQuery Update script *(on page 601)* or XSLT stylesheet *(on page 606)* 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 601)* or XSLT method example *(on page 606)* to help you with the content.

There are cases when it is necessary to add parameters in the XQuery script *(on page 601)* or XSLT stylesheet *(on page 606)*. 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 Developer. 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 1190)* 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 1122)*.

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 190)*.

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 604)* or XSLT stylesheet *(on page 609)*.

Related Information:
XQuery Update Script for Creating a Custom Operation *(on page 601)*
XSLT Stylesheet for Creating a Custom Operation *(on page 606)*

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 descriptor file root element specifies required attributes to define the operation `@name`, `@description`, and `@id` which are necessarily when loading an XML Refactoring operation. It also contains the path to the XQuery Update script *(on page 601)* or XSLT stylesheet *(on page 606)* that is associated with the particular operation through the `<script>` element.

The optional `@filesFilter` attribute can be specified to filter the resources by using a file pattern or list of file patterns separated by a comma (for example: `filesFilter="*.dita, *.xml"`). When set, its value is automatically populated in the **Include files** field within the **Scope and Filters** wizard page *(on page 581)* as a default value.

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 global 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 579)* 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 579)*.

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 Developer 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 190)* 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 Developer 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 2308)* for an element or attribute.
- **BOOLEAN** - Used to edit boolean parameters. For the **BOOLEAN** type, the possible values are "true" and "false". The equivalent parameter defined in the XSLT stylesheet or XQuery script must have the `xs:boolean` type.
- **TEXT_CHOICE** - It is useful for parameters whose value should be from a list of possible values. Oxygen XML Developer renders each possible value as a radio button option.

- **optional** - Specifies whether the parameter is optional or required. For optional parameters, the end user is not required to fill in a value in the XML refactoring wizard. By default if this attribute is missing a parameter is required.
- **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:

```
<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>
```

On a `<value>`, you can specify the `@default` attribute with the value `true` to mark it as the default presented value in the XML refactoring wizard. The text specified inside the `<value>` element is displayed as placeholder default text in the text entry box. If the dialog box is accepted with the placeholder text in place, the `@name` attribute value is passed to the refactoring script. Example:

```
<value name="my-actual-default-value" default="true">[default displayed]</value>
```

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 Developer 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>`:

```
<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>`:

```
<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>
```

Grouping Parameters in the Descriptor File

You can use `<section>` elements to group related parameters in the descriptor file:

```
<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>
```

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 609)*
Example of an Operation Descriptor File with an XQuery Update script *(on page 604)*

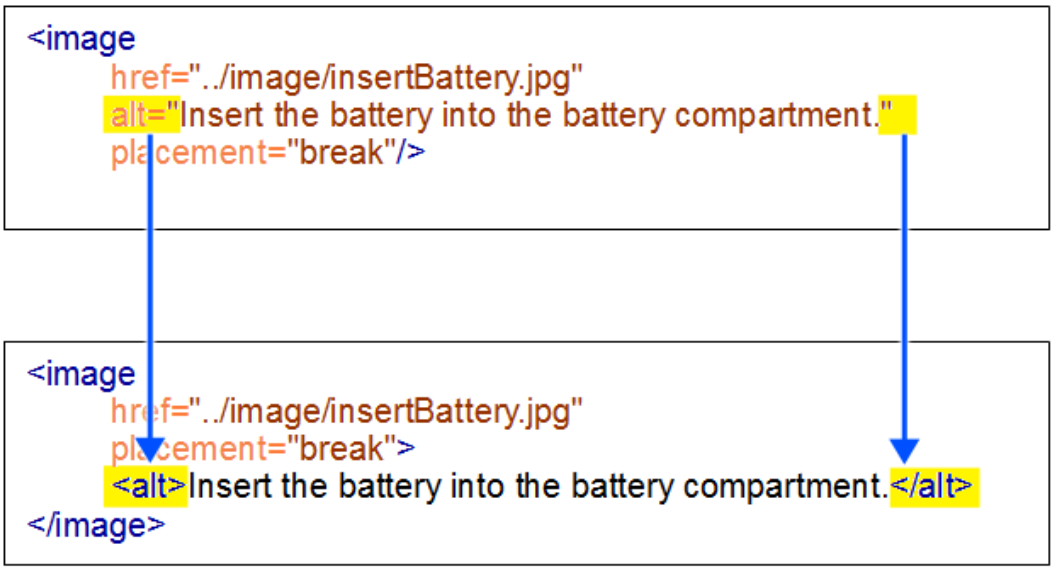
XSLT Stylesheet for Creating a Custom Operation

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 475. 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 1948)* 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 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:import
href="http://www.oxygenxml.com/ns/xmlRefactoring/resources/commons.xsl"/>
```

```
<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>
                <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: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 2309)* set in the XML Refactoring *framework (on page 2305)*.

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 (for example, named `convert-attribute-to-element.xsl`), you have to create an XML Refactoring operation descriptor (for example, named `convert-attribute-to-element.xml`) that references the stylesheet and provides descriptions and possible values for its parameters. 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
 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>
 <script type="XSLT" href="convert-attribute-to-element.xsl"/>
  <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>
  </parameters>
</refactoringOperationDescriptor>
```

Note:

If you are using an XSLT file, the line with the `<script>` element would look like this:

```
<script type="XSLT" href="convert-attribute-to-element.xsl"/>
```

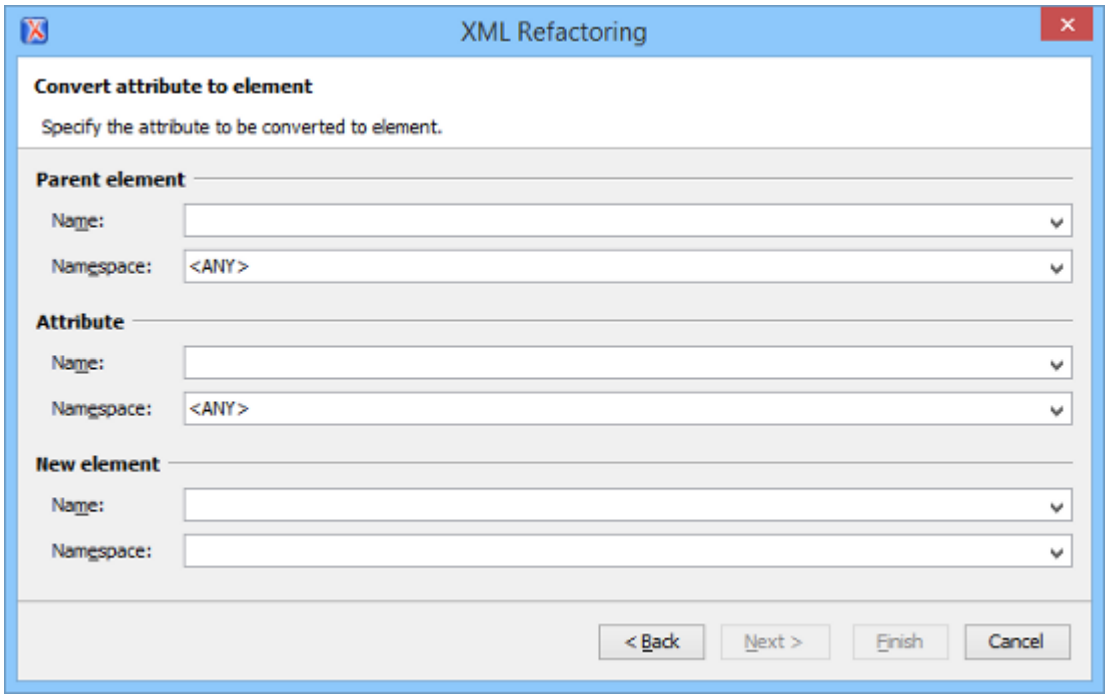
The code exemplified above and other refactoring examples can be found on the DITA Refactoring GitHub sample project.

Results

After you have created these files, copy them into a folder scanned by Oxygen XML Developer when it loads the custom operation *(on page 613)*. 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 476. Example: XML Refactoring Wizard for a Custom Operation



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 601)* 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 version="1.0" encoding="UTF-8"?>
<!-- comment before root -->
<?pi before root ?>
<root>
    <child></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:

  ```
  <!-- 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 version="1.0" encoding="UTF-8"?>
  <!-- comment before root -->
  <?pi before root ?>
  <root>
      <child></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 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:

  ```
  <!-- Inserted comment --><root>
      <child></child>
  </root>
  <!-- comment after root -->
  <?pi after root ?>
  ```

XSLT Example:

To process content after the root node, the XSLT would look like this:

```
<?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()"/>

        <xsl:variable name="processedContent"
                      select="concat($after-root-content, $commentAsText)"/>

        <!-- Update the content after the root element -->
        <xsl:value-of select="xrf:set-content-after-root($processedContent)"/>

        <xsl:apply-templates/>
    </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 Developer scans the following locations when looking for XML Refactoring operations to provide flexibility:

- A folder named `refactoring`, created inside the folder of the *framework* you are customizing. In the **Classpath** tab of the **Document type** configuration dialog box *(on page 99)*, you need to add a reference to the `refactoring` folder specific for the framework.
- A folder that you specify in the **Load additional refactoring operations from** text box *(on page 200)* in the **XML Refactoring** preferences page *(on page 200)*.

> **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 2308)*:
>   a. Open the **Preferences** dialog box **(Options > Preferences)** *(on page 79)* and go to **XML > XML Refactoring** *(on page 200)*.
>   b. Select **Project Options** *(on page 2308)* at the bottom of the dialog box.
> 3. In the **Load additional refactoring operations from** text box *(on page 200)*, use the *${pd}* editor variable *(on page 259)* so that the folder path is declared relative to the project.

- A folder specified by the XML Refactoring Operations Plugin Extension *(on page 1879)*.
- The `refactoring` folder from the Oxygen XML Developer installation directory (*`[OXYGEN_INSTALL_DIR]/refactoring/`*).

Sharing Custom Refactoring Operations

The purpose of Oxygen XML Developer 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 2305)* or projects.

After storing custom operations, you can share them with other users by sharing the resources.

Localizing XML Refactoring Operations

Oxygen XML Developer 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 `<description>` of the `<parameters>` element.
- The `label`, `description`, and `possibleValues` for each `parameter`.

Translated refactoring information uses the following form:

```
${i18n(translation_key)}
```

Oxygen XML Developer 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 79)*, 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 1879)*.
- The `refactoring/i18n` folder from the Oxygen XML Developer installation directory (*`[OXYGEN_INSTALL_DIR]`*`/refactoring/i18n`).

Example: Refactoring Operation Descriptor File with *i18n* Support

```
<?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>
```

Generating Sample XML Files

Oxygen XML Developer 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 172)*.

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 694)*.

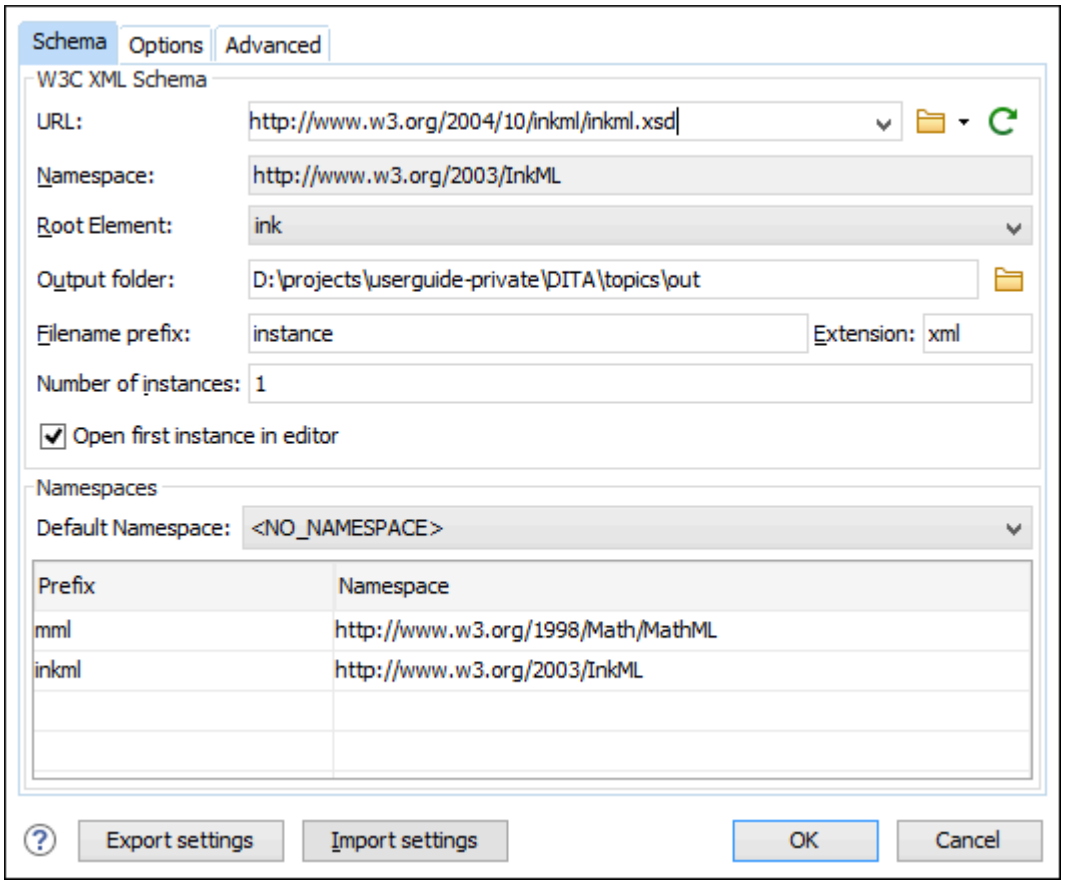
The action opens the **Generate Sample XML Files** dialog box that allows you to configure a variety of options for generating the files.

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 176)*. 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 477. 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

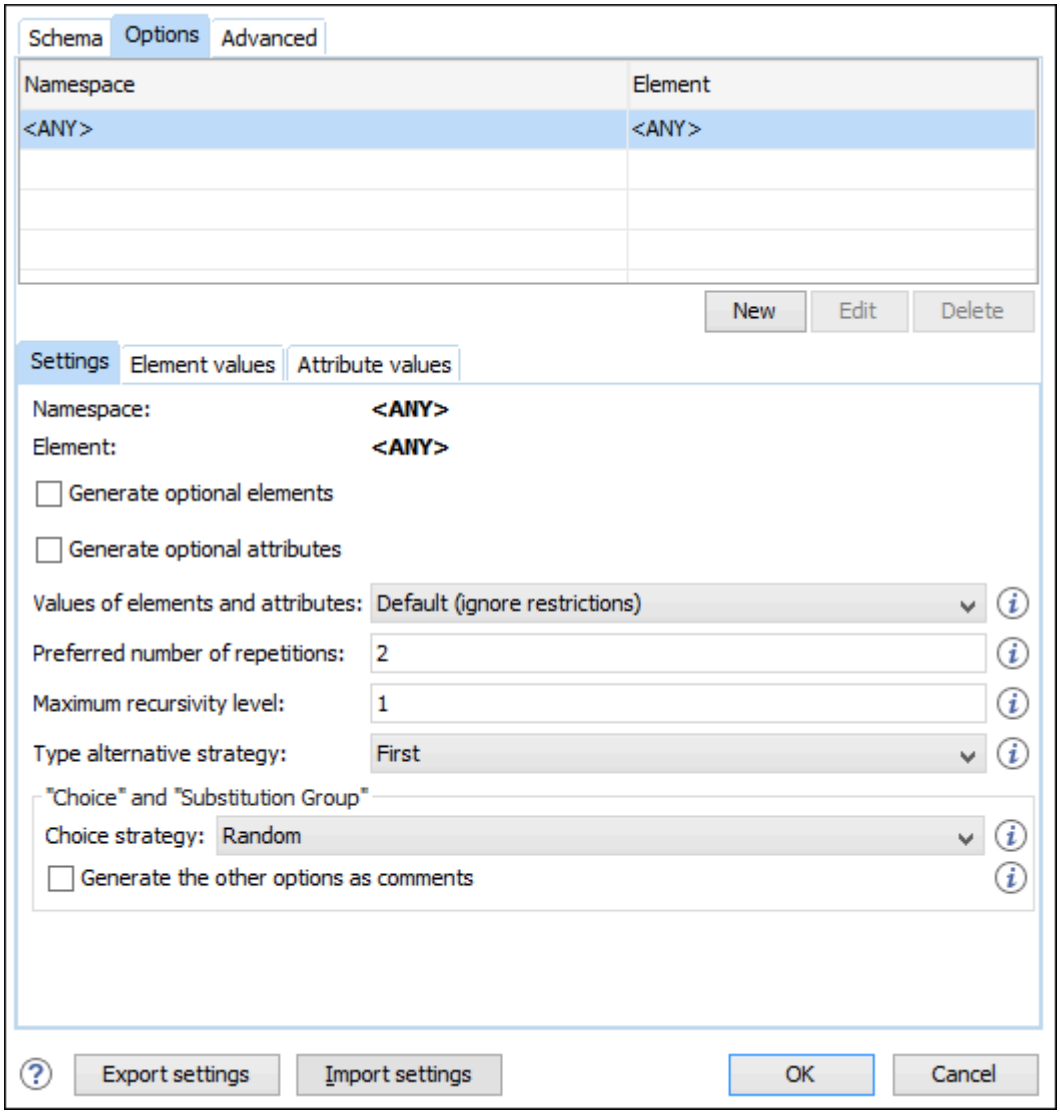
Use this button to load previously exported settings.

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 478. 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:
  - 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

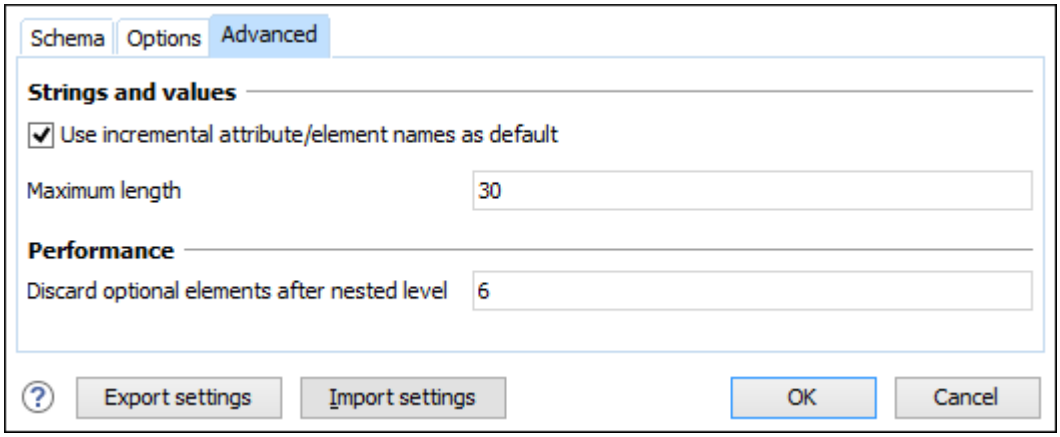
Use this button to load previously exported settings.

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 479. 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 2263)*.

Applying All Default Quick Fix Proposals

The **Apply all default quick fix proposals** tool can be used to apply quick fix proposals for all reported validation errors in one or more documents when one or more quick fix proposals have been detected for reported validation errors. If multiple quick fixes are available for the same validation error, the default quick fix defined in the Schematron validation schema using the `sqf:default-fix` attribute is automatically selected to be applied. If no default quick fix is explicitly specified, the first quick fix proposal in the list is considered. All quick fix proposals are then automatically executed in bulk, one after the other.

Important Notes to Consider:

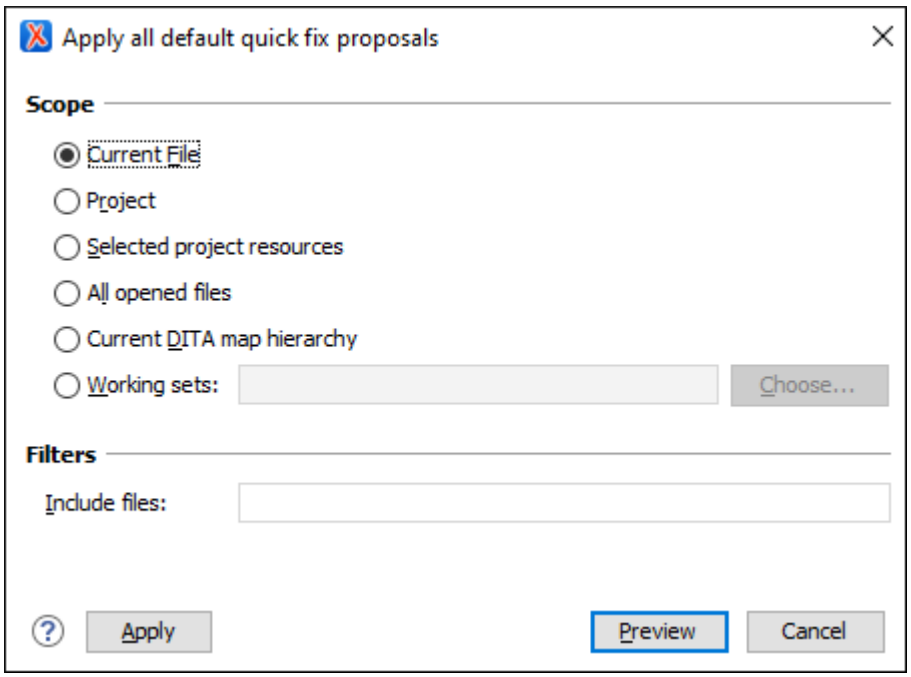
- To maintain the accuracy of the initially calculated error validation ranges, the quick fix proposals are applied in the reverse order of their selection.
- If two or more quick fixes act on the same "area" within the document, only one is applied (no changes can be made to changes already made).
- Quick fixes that involve "user-entered values" that normally present a dialog box to facilitate data entry will not be executed (the automatic process of applying all selected quick fixes cannot be interrupted by the presence of the respective dialog boxes).

The **Apply all default quick fix proposals** tool can be accessed from the following interface locations:

- **[Single Document]** The contextual menu when you right-click anywhere in **Text** mode *(on page 492)* when there is a detected quick fix proposal in the current document.
- **[One or More Documents]** The **Tools** menu.
- **[One or More Documents]** The **Refactoring** submenu when you right-click one or more files in the **Project** view *(on page 334)*.

Invoking the tool opens the **Apply all default quick fix proposals** dialog box where you can choose the scope for the operation and optionally specify a filter to narrow the scope.

Figure 480. Apply All Default Quick Fix Proposals Dialog Box



The **Scope** section allows you to choose from the following scopes:

- **Current File** - The quick fix proposals are applied on the current file only.
- **Project** - The quick fix proposals are applied on the current project.

- **Selected project resources** - The quick fix proposals are applied on the selected files from the current project.
- **All opened files** - The quick fix proposals are applied on all opened files.
- **Entire DITA map hierarchy** (for DITA documents) - The quick fix proposals are applied on current DITA map along with all of its referenced topics and submaps (and topics referenced in those submaps).
- **Working sets** - The quick fix proposals are applied on a user-defined set of resources, referred to as a *working set (on page 2309)*.

The **Include files** option in the **Filters** section can be used to narrow the scope of the operation by filtering the selected resources using a file pattern. For example, to restrict the operation to only parse and update XML documents from the selected files, you could use `{*}.xml` for the file pattern.

You can click the **Preview** button to see a comparison preview that provides an overview of the content changes that will be made, according to the quick fixes that will be applied. The comparison panel also informs you of any problems encountered. While previewing the changes, you can also choose to exclude certain files from having the quick fixes be applied if you do not agree with the changes for the particular files. If you agree with the changes presented, click **Apply** to trigger the quick fixes and update the content.

Troubleshooting:
If a quick fix did not apply, a notification with some information about the error will be displayed in the **Results** view at the bottom of the application.

Tip:
If the operation takes longer than expected, you can use the **Stop** button in the progress bar to cancel the operation.

Resources

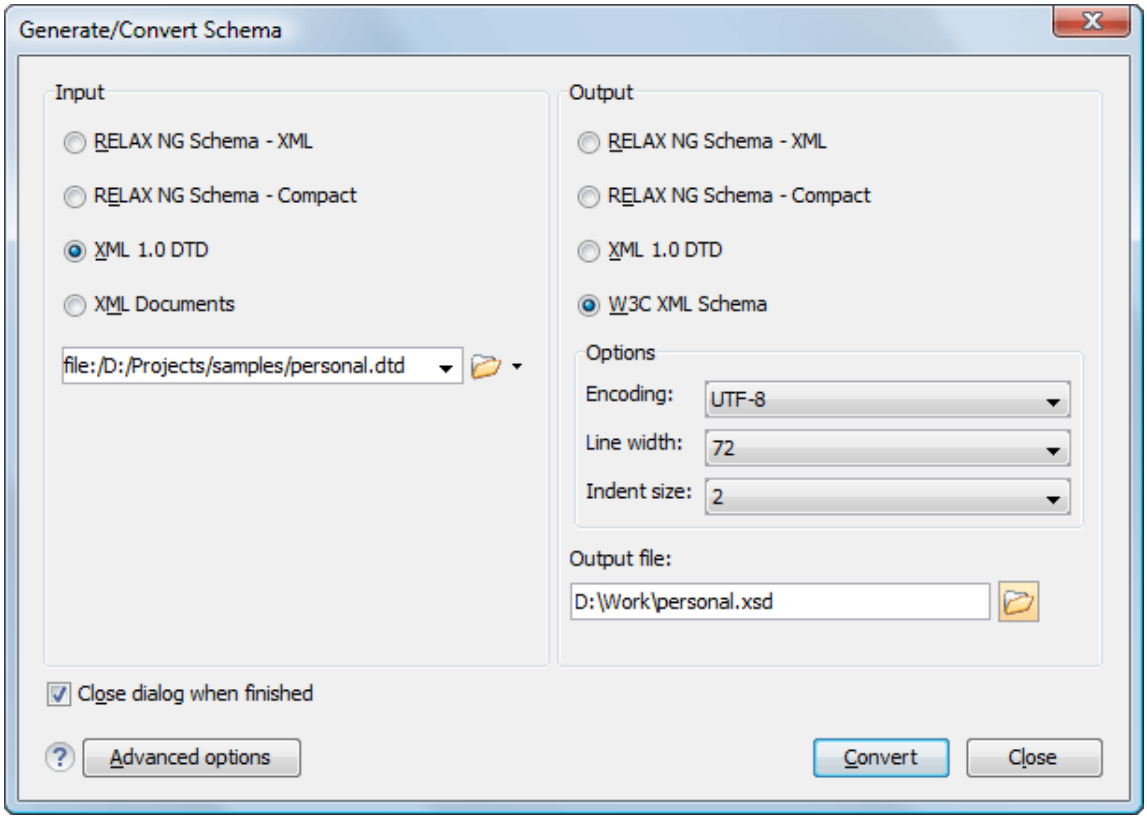
For more information about the **Apply all default quick fix proposals** tool, see our video: Introducing the Apply All Default Quick Fix Proposals Tool.

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 Developer generates an approximation of the source schema. Oxygen XML Developer 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 + Option + 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 334)*. This action opens the **Generate/Convert Schema** dialog box that allows you to configure various options for conversion.

Figure 481. 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 `<anyAttribute>` elements. The default is skip (corresponding to RELAX NG semantics).

Converting Database to XML Schema

Oxygen XML Developer 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 205)* 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**.

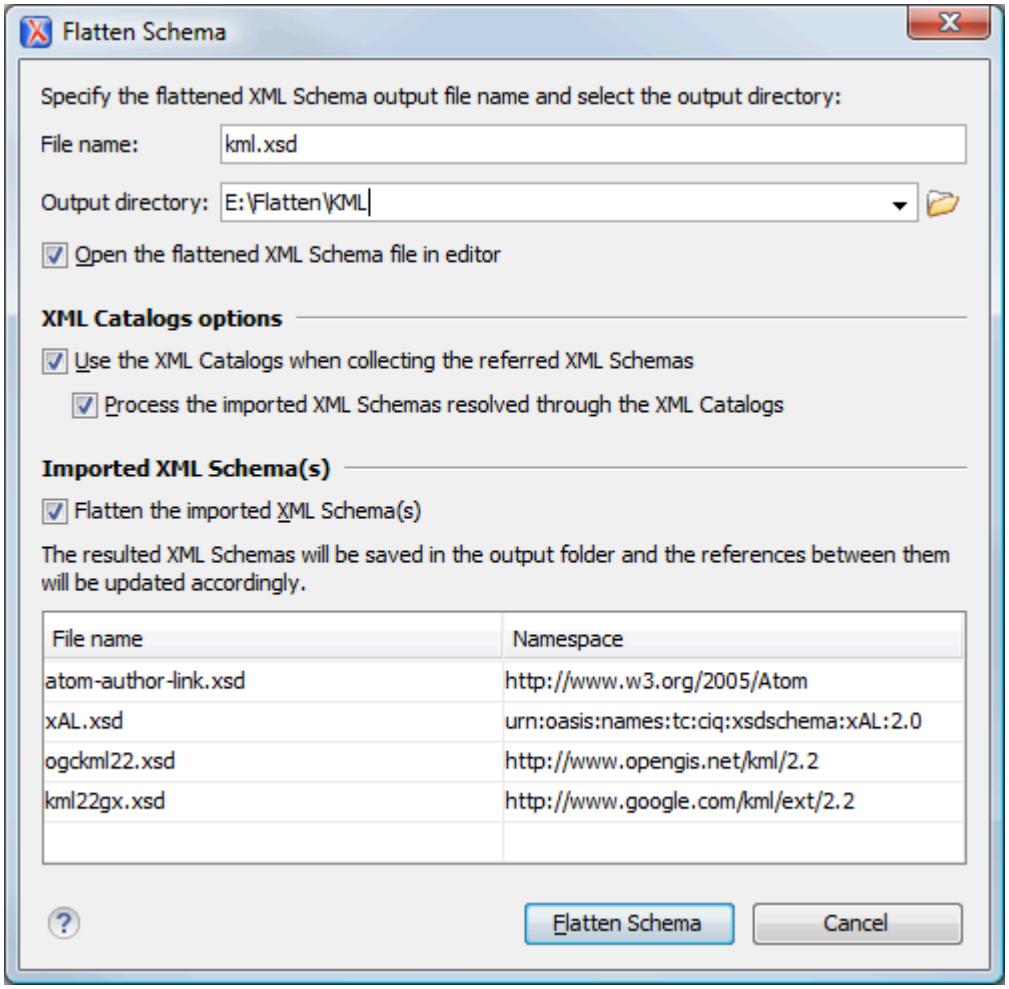
Result: The database structure is converted to an XML Schema and it is opened for viewing and editing.

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 Developer 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 482. 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 Developer 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 2309)*.

> **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 2309)* 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 2263)*.

Generating Java Classes from XML Schema

Oxygen XML Developer 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 Developer will present a dialog box asking if you want to install it. Once installed, you need to restart Oxygen XML Developer and the action will invoke the Java class generator tool.

Add-on documentation and installation details.

Compiling an XSL Stylesheet for Saxon

As of Saxon 12.9, 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 Developer 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 Developer** - You can also use a *stylesheet export file* (SEF) in Oxygen XML Developer 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 1969)*. When configuring the XSLT transformation, you will specify the SEF file in the **XSL URL** field *(on page 1103)*.

Compiling an SEF File

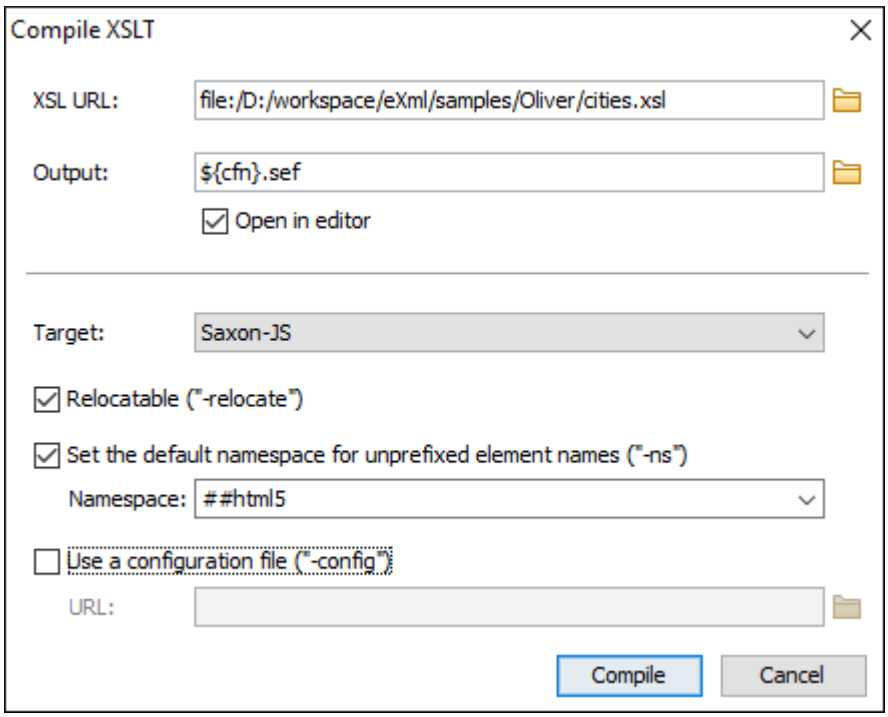
The **Compile XSL Stylesheet for Saxon** tool can be found in the **Tools** menu and it compiles a specified stylesheet as a JSON or an XML file (*stylesheet export file*).

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.

Figure 483. Compile XSLT Stylesheet for Saxon Dialog Box



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 Developer 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 12.9 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.

Attention:
Oxygen XML Developer does not support the `ALLOWED_PROTOCOLS` Saxon property (*http://saxon.sf.net/feature/allowedProtocols*). This feature is specific to Saxon and controls access by Saxon (but not by underlying software, such as the XML parser). For more information, see https://www.saxonica.com/documentation12/index.html#!configuration/config-features.

Compile

Use this button to generate the *stylesheet export file* according the options selected in this dialog box.

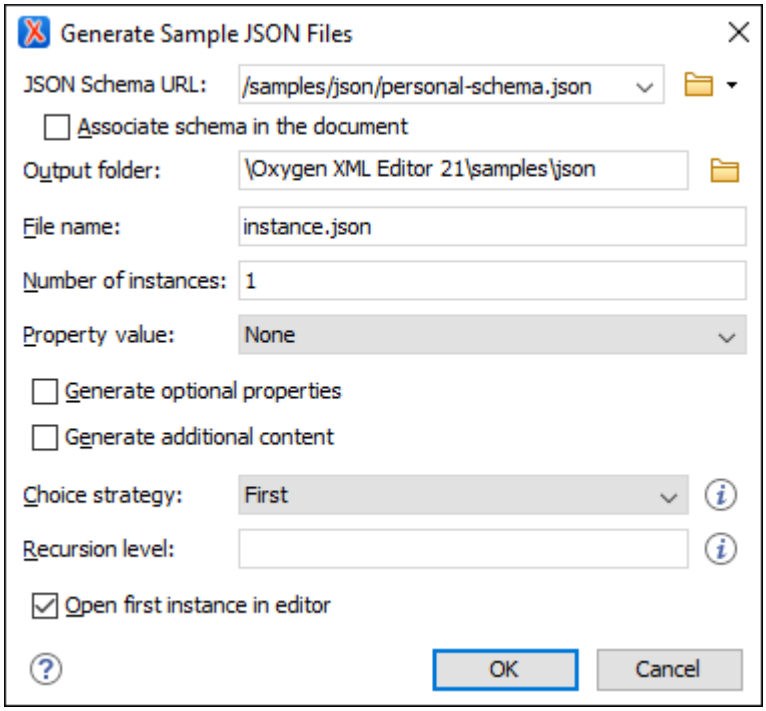
JSON Tools

Oxygen XML Developer includes some useful tools for converting between JSON, XML and YAML, converting XSD to JSON Schema, generating JSON instances or a JSON Schema, and OpenAPI (JSON and YAML) testing.

Generating Sample JSON Files from a JSON Schema

Oxygen XML Developer 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.

Figure 484. Generate Sample JSON Files Dialog Box



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. The tool supports schemas with versions *Draft 04*, *06*, *07*, *2019-09*, and *2020-12*.

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

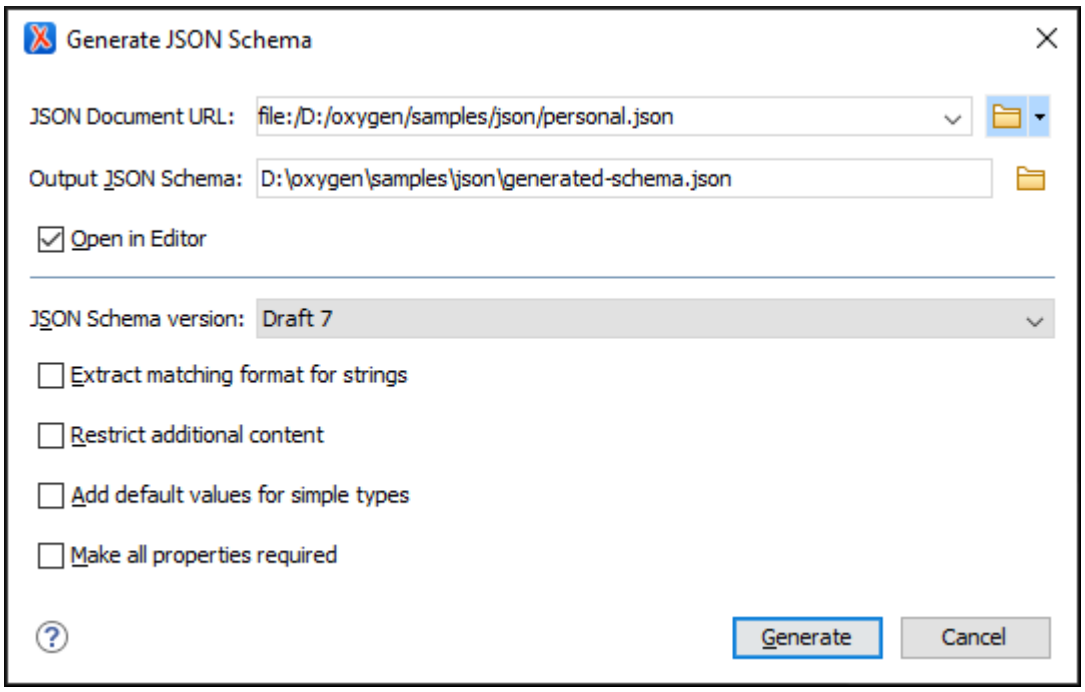
If selected, the first generated instance is opened in the editor.

You can click **OK** at any point to generate the sample JSON files.

Generating JSON Schema from a JSON File

Oxygen XML Developer 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 485. 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**, **2019-09**, and **2020-12**.

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

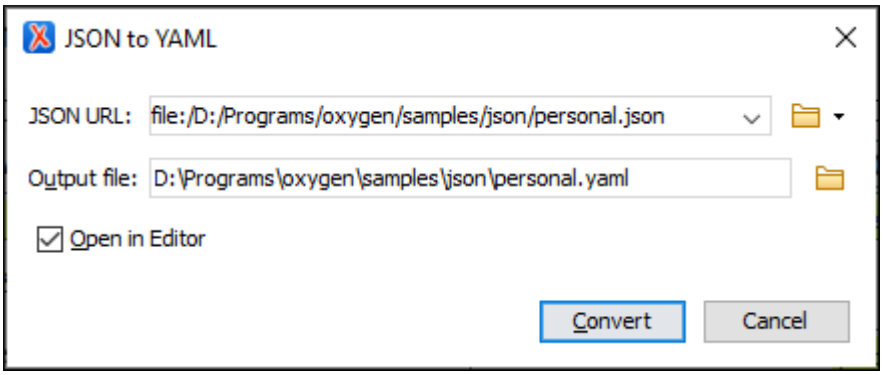
Oxygen XML Developer includes a useful and simple tool for converting JSON files to YAML. The **JSON to YAML** action for invoking the tool can be found in the **Tools > JSON Tools** menu.

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:

   Figure 486. 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 879)*

YAML to JSON Converter

Converting YAML to JSON in Oxygen

Oxygen XML Developer 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.

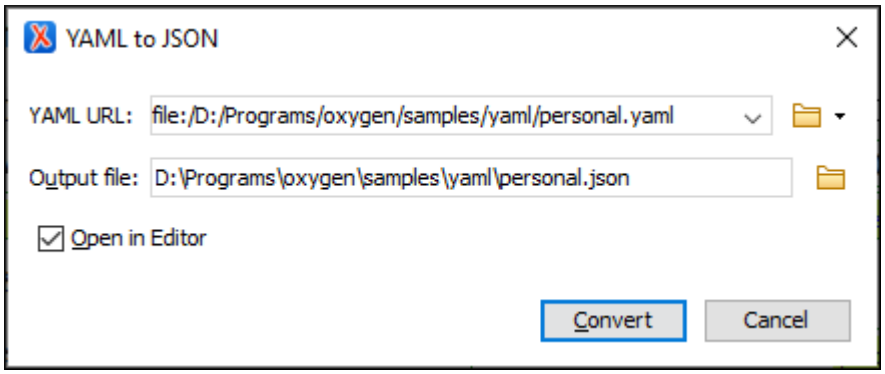
The **YAML to JSON** action for invoking the tool can be found in the **Tools > JSON Tools** menu.

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:

   Figure 487. 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 879)*

JSON to XML Converter

Discover the Oxygen JSON Editor!
Tailored for Working with JSON and JSON Schema Documents

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

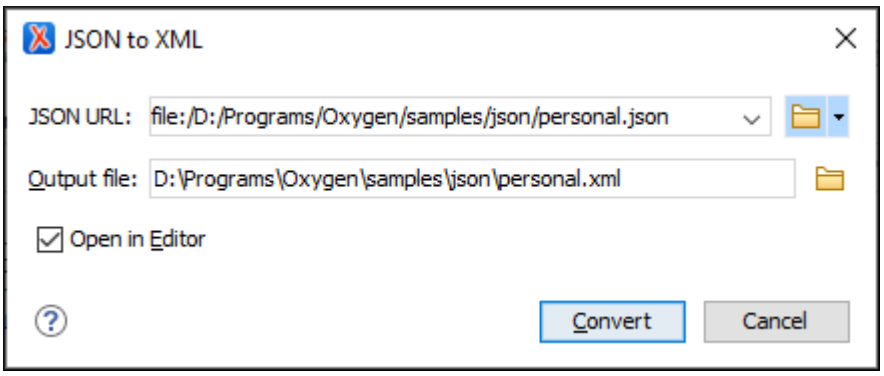
Oxygen XML Developer includes a useful and simple tool for converting JSON files to XML. The **JSON to XML** action for invoking the tool can be found in the **Tools > JSON Tools** menu.

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:

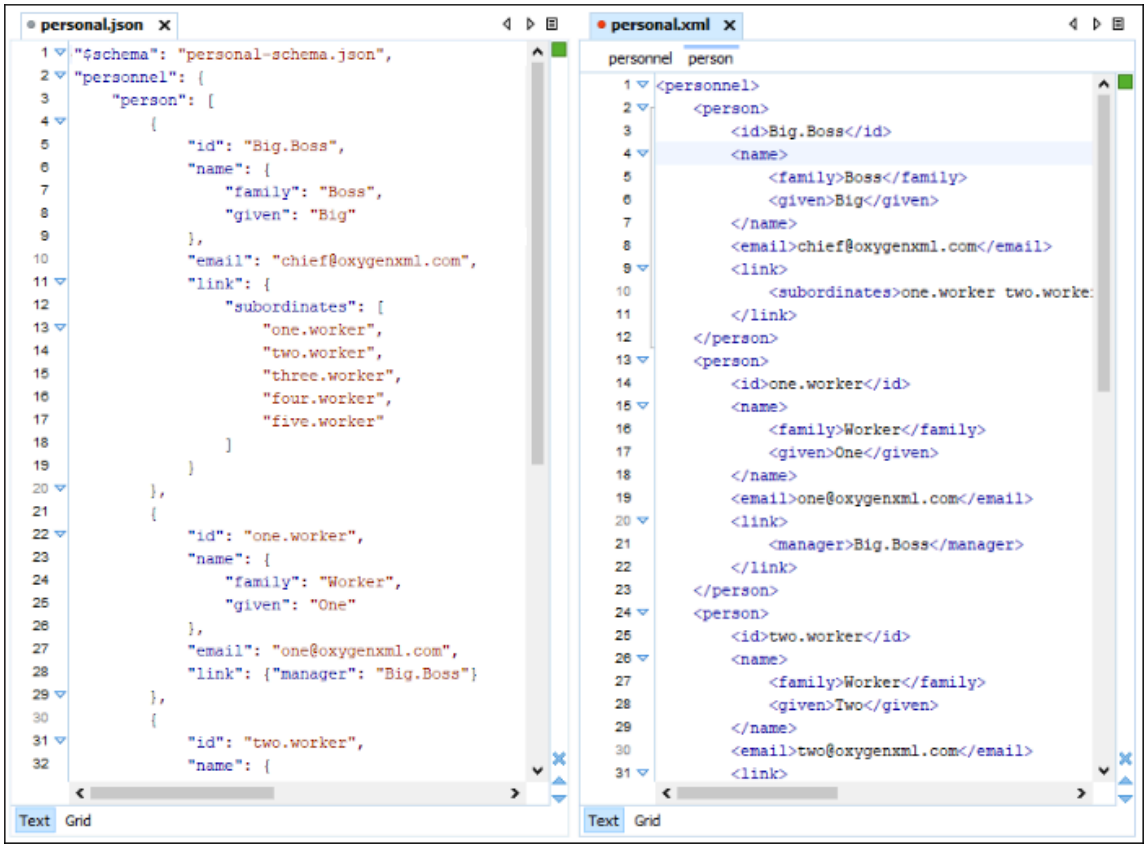
   Figure 488. JSON to XML Dialog Box

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.

Figure 489. Example: XML to JSON Operation Result



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:

```
{
   "personnel": {
    "person": [
      {"name": "Boss"},
      {"name": "Worker"}
    ]
  },
  "id":"personnel-id"
}
```

  it is converted to:

```
<?xml version="1.0" encoding="UTF-8"?>
<JSON>
  <personnel>
    <person>
      <name>Boss</name>
```

```
    </person>
    <person>
      <name>Worker</name>
    </person>
  </personnel>
  <id>personnel-id</id>
</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.

```
[
  {"name": "Boss"},
  {"name": "Worker"}
]
```

it is converted to:

```
<?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.

```
{"$id": "personnel-id"}
```

is converted to:

```
<_X24_id>personnel-id</_X24_id>
```

Related Information:

XML to JSON Converter *(on page 876)*

XML to JSON Converter

Discover the Oxygen JSON Editor!
Tailored for Working with JSON and JSON Schema Documents

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

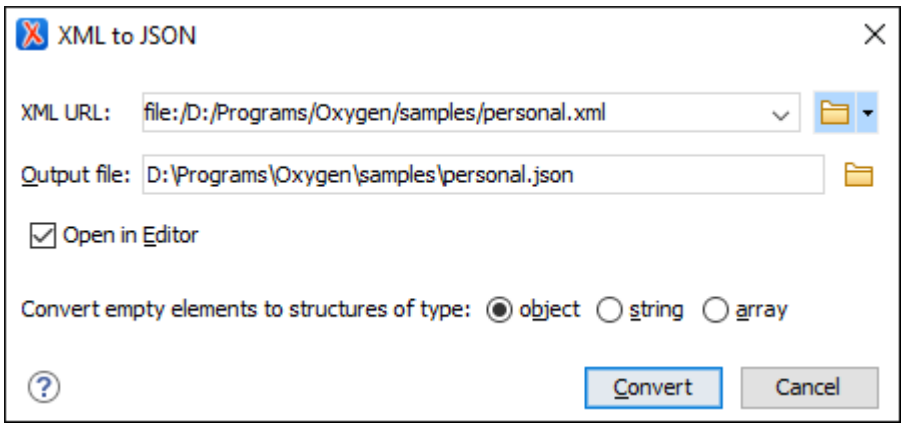
Oxygen XML Developer includes a useful and simple tool for converting XML files to JSON. The **XML to JSON** action for invoking the tool can be found in the **Tools > JSON Tools** menu.

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:

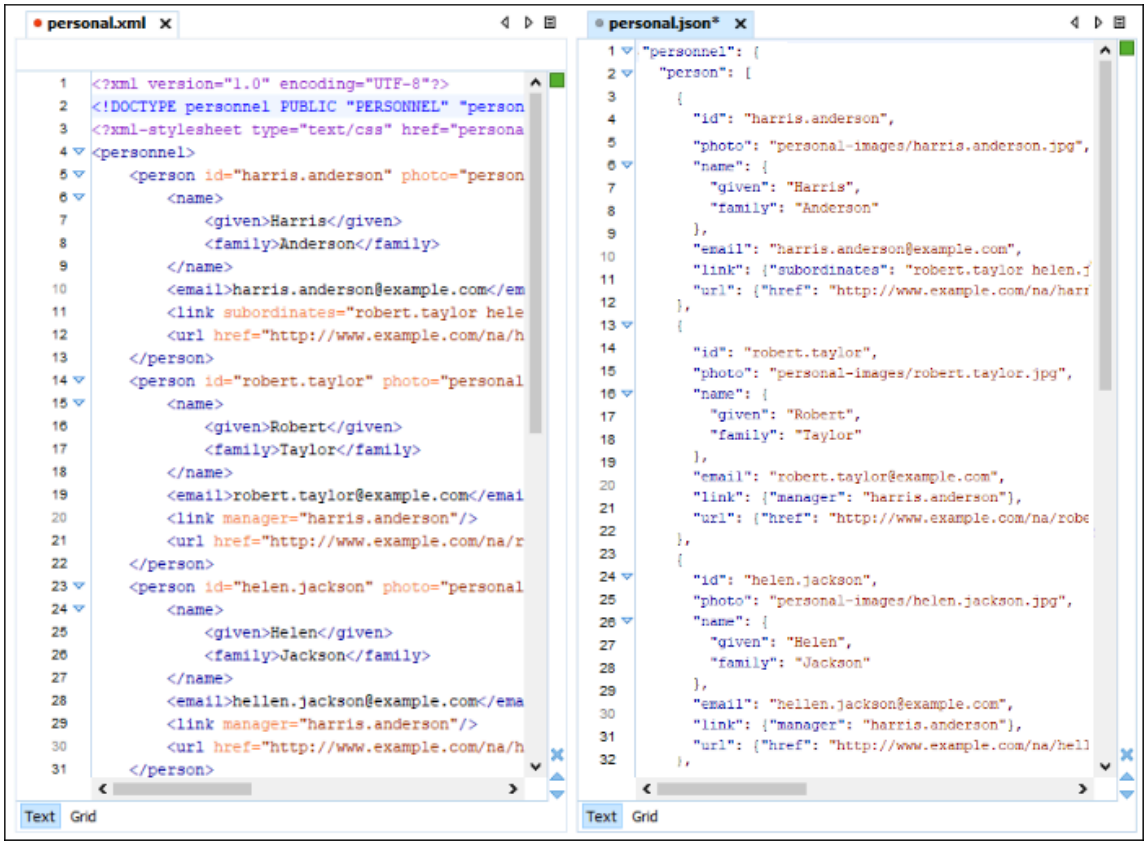
   Figure 490. XML to JSON Dialog Box

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.

Figure 491. Example: XML to JSON Operation Result



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:

```
<personnel>
  <person id="person.one">
    <name>Boss</name>
  </person>
  <person id="person.two">
    <name>Worker</name>
  </person>
</personnel>
```

  it is converted to:

```
{
  "personnel": {
    "person": [
      {
```

```
        "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).

  ```
  <p>This <b>is</b> an <b>example</b>!</p>
  ```

  is converted to:

  ```
  {
    "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 873)*), it will be converted to the normal character value in the converted JSON document.

  ```
  <_X24_id>personnel-id</_X24_id>
  ```

  is converted to:

  ```
  {"$id": "personnel-id"}
  ```

Related Information:

JSON to XML Converter *(on page 873)*

XSD to JSON Schema Converter

Oxygen XML Developer 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 Developer will present a dialog box asking if you want to install it. Once installed, you need to restart Oxygen XML Developer and the **XSD to JSON Schema** action will invoke the tool.

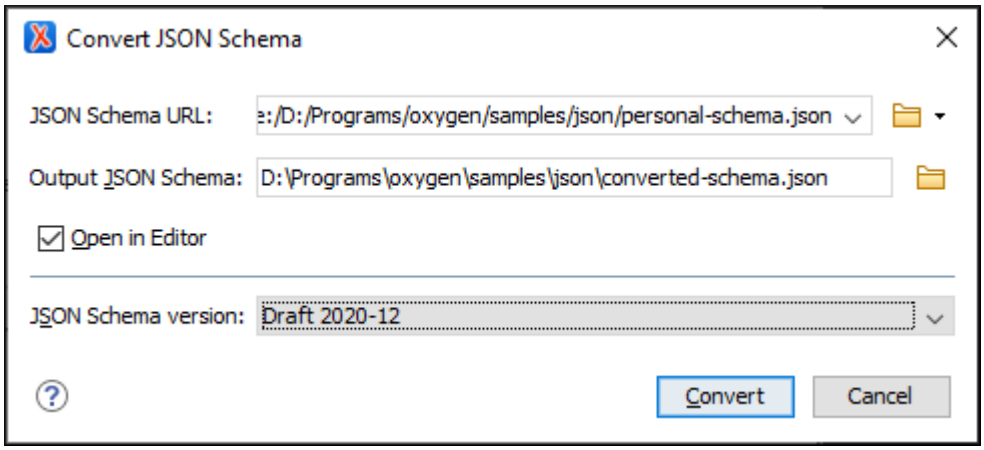
Addon documentation and installation details.

JSON Schema Converter

Oxygen XML Developer includes a tool for converting an older version of a JSON schema (Draft 4, 6, or 7) to the latest versions (2019-09 or 2020-12).

To convert a JSON schema, select **Convert JSON Schema** from the **Tools > JSON Tools** menu. The action opens a dialog box where you can configure some options for converting the JSON Schema.

Figure 492. Convert JSON Schema Dialog Box



The **Convert JSON Schema** 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 JSON Schema

The path to the folder where the converted JSON schema will be saved.

Open in Editor

If selected, the converted JSON schema is opened in the editor.

JSON Schema version

The version of the resulting JSON schema. The possible choices are: **Draft 2019-09** or **2020-12**.

You can click **Convert** at any point to generate the JSON Schema.

Conversion Notes

- The `$schema` declaration is changed according to the selected JSON schema version.
- The `definitions` keyword is converted to `$defs` and all the references are updated.
- The `dependencies` keyword is split into `dependentRequired` and `dependentSchemas`.
- The `items` keyword (tuple array) is converted to `prefixItems` (2020-12).
- The `additionalItems` keyword is converted to `items` (2020-12, only if `prefixItems` is present).
- The `exclusiveMinimum` and `exclusiveMaximum` keywords with boolean values (Draft 4) are removed.
- The `id` keyword (Draft 4) is converted to `$id`.
- The `$ref` keyword wrapped into 1-item `allOf` is unwrapped because the latest versions allow processing `$ref` along with other keywords.

OpenAPI Tester

Oxygen XML Developer 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 Developer presents a dialog box asking if you want to install it. Once installed, you need to restart Oxygen XML Developer and the **OpenAPI Tester** action will invoke the tool.

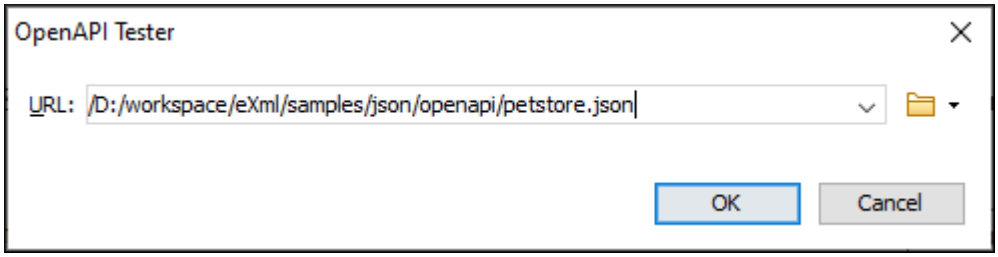
Run OpenAPI Test Scenarios

Oxygen XML Developer includes a testing tool for running *OpenAPI* test scenarios. The **Run OpenAPI Test Scenario** tool provides the ability to run a test suite for an OpenAPI document in JSON format. It performs the requests based on the specified OpenAPI document and the data entered in the test file, and then checks if the server responses are as expected.

Attention:
This tool requires the **OpenAPI Tester** add-on *(on page 1983)* (version 1.2.0 or newer) to be installed before it becomes available in the **JSON Tools** menu.

To use the tool, select **Run OpenAPI Test Scenario** from the **Tools > JSON Tools** menu. This opens a dialog box where you can specify the location of the test scenario file that you want to run.

Figure 493. Run OpenAPI Test Scenario Dialog Box



The scenario file must be valid according to the schema from here: `frameworks/json/schemas/openapi/scenario/openAPIScenario.jschema`. There is a default scenario file template available when creating new documents from templates *(on page 299)* and it can be found in the **Framework Templates > OpenAPI Test Scenario**. The template will automatically be validated against the schema.

For the scenario file, you have to specify the path of the OpenAPI document and the server where the requests are made. Then, for each test, you need to enter valid data for the required fields "**path**", "**operation**", "**expectedResponse**", and the optional fields "**description**", "**parameters**", "**authorization**", or "**body**".

After successfully running the test scenario, the results are displayed in a new JSON file.

Tip:
Oxygen XML Developer includes a specialized framework for editing and working with OpenAPI test scenario files *(on page 1066)*.

Resources

For more information about OpenAPI editing, testing, and documenting, watch our webinar:

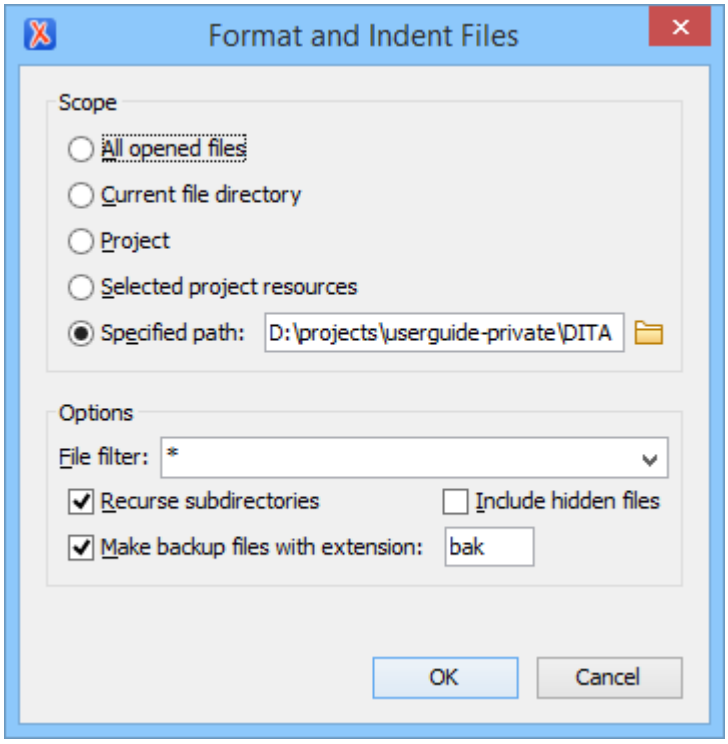
https://www.youtube.com/embed/gKdabeh49Qk

Format and Indent (Pretty-Print) Multiple Files

Oxygen XML Developer provides support for formatting and indenting (*pretty-print (on page 2307)*) 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 334)* or from the **Tools** menu. This opens the **Format and Indent Files** dialog box that allows you to configure options for the action.

Figure 494. Format and Indent Files Dialog Box



The **Scope** section allows you to choose from the following scopes:

- **All opened files** - The *pretty-print (on page 2307)* 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 2307)* 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 2307)* 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 2307)* is also performed in the hidden files.
- **Make backup files with extension** - When selected, Oxygen XML Developer 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 Developer includes a tool for generating documentation for XSLT, XML Schema, XQuery, WSDL, JSON schema, and OpenAPI documents.

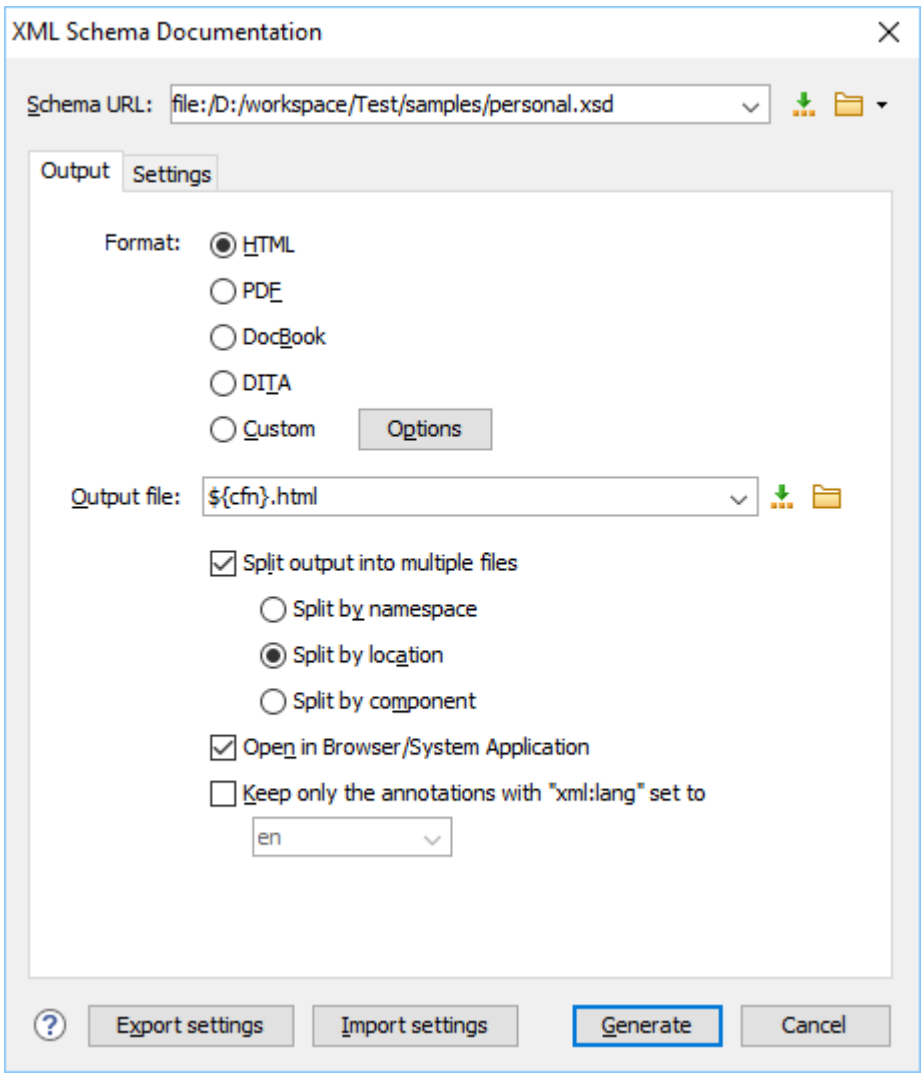
Generating Documentation for an XML Schema

Oxygen XML Developer 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 334)*. You can also open the tool by using the **Generate Documentation** toolbar button.

Figure 495. XML Schema Documentation Dialog Box



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 758)*.
  - **PDF** - The documentation is generated in PDF output format *(on page 761)*.
  - **DocBook** - The documentation is generated in DocBook output format *(on page 761)*.
  - **DITA** - The documentation is generated in DITA output format *(on page 761)*.
  - **Custom** - The documentation is generated in a custom output format *(on page 761)*, 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 Developer (in the current editor).

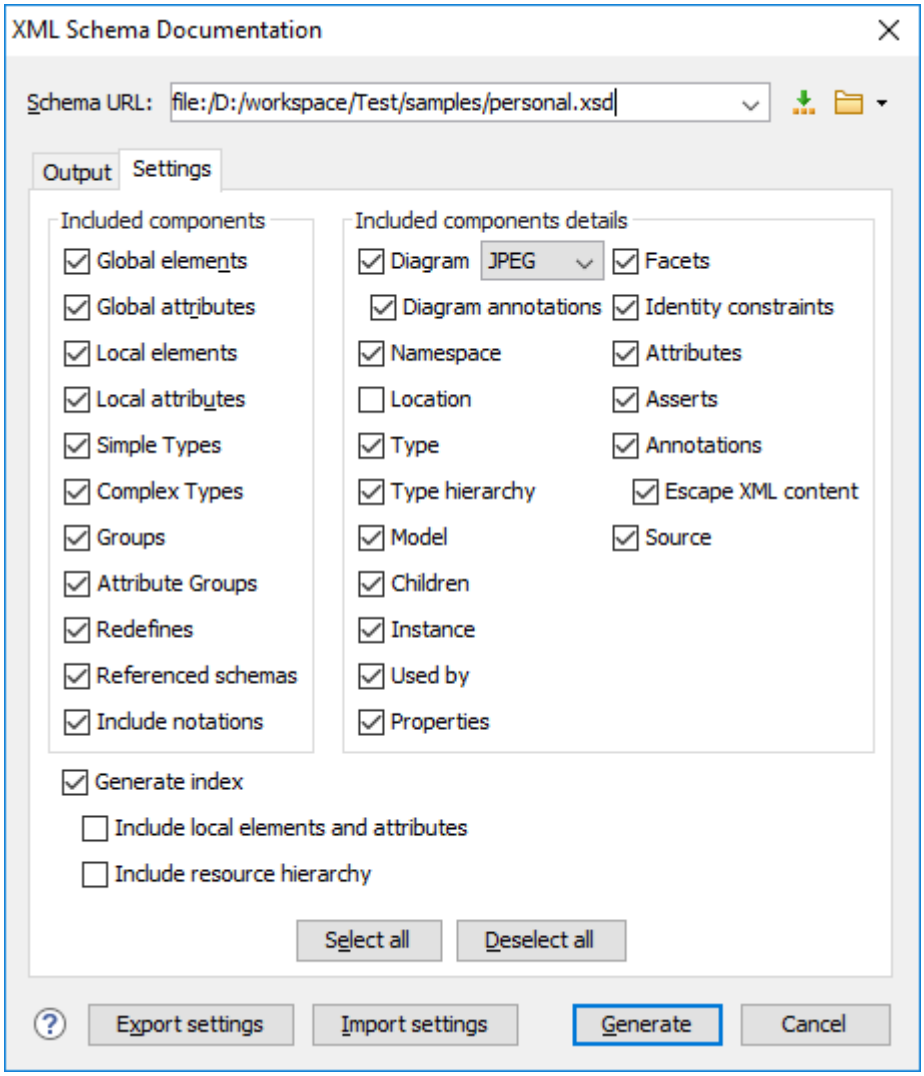
  > **Note:**
  > To set the browser or system application that will be used, open the **Preferences** dialog box **(Options > Preferences)** *(on page 79)*, go to **Global**, and set it in the **Default Internet browser** field. This will take precedence over the default system application settings.

- **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.

Figure 496. Settings Tab of the XML Schema Documentation Dialog Box



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 132)* 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 2263)*.

Related Information:
Customizing PDF or DocBook Output of Generated XML Schema Documentation *(on page 762)*

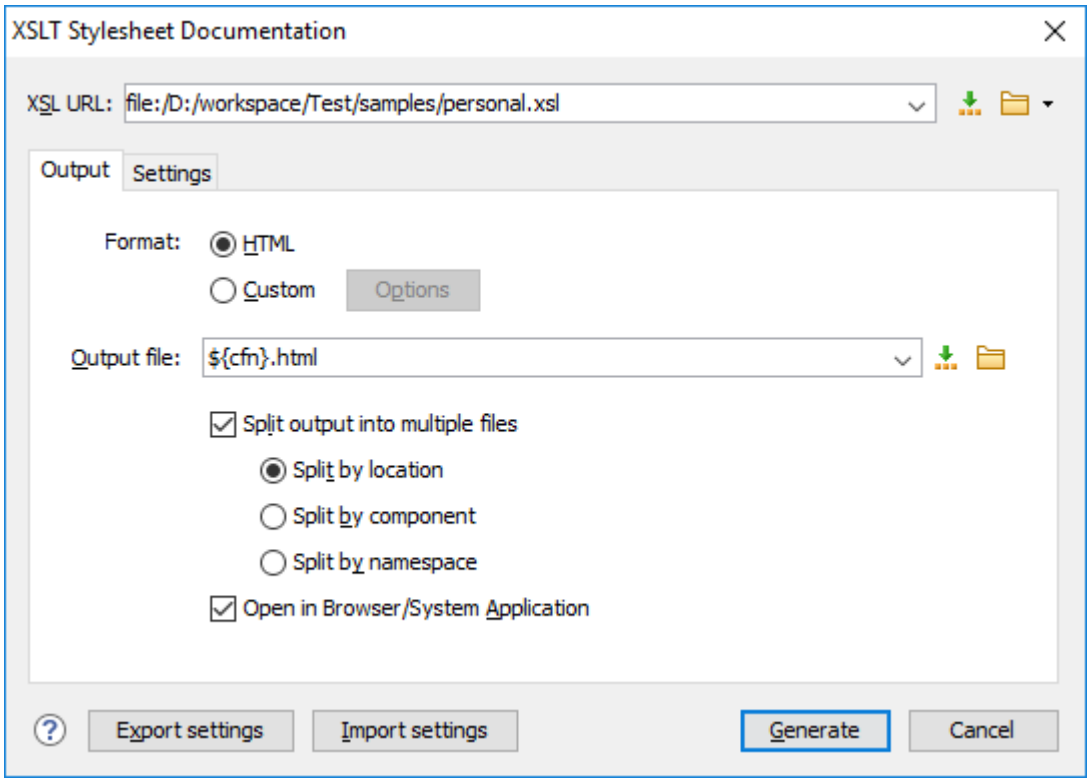
Generating Documentation for an XSLT Stylesheet

You can use Oxygen XML Developer 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 671)*, 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 334)*. You can also open the tool by using the **Generate Documentation** toolbar button.

Figure 497. 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 668)*.
  - **Custom** - The documentation is generated in a custom output format *(on page 671)*, allowing you to control the output. Click the **Options** button to open a **Custom format options** dialog box *(on page 672)* 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.

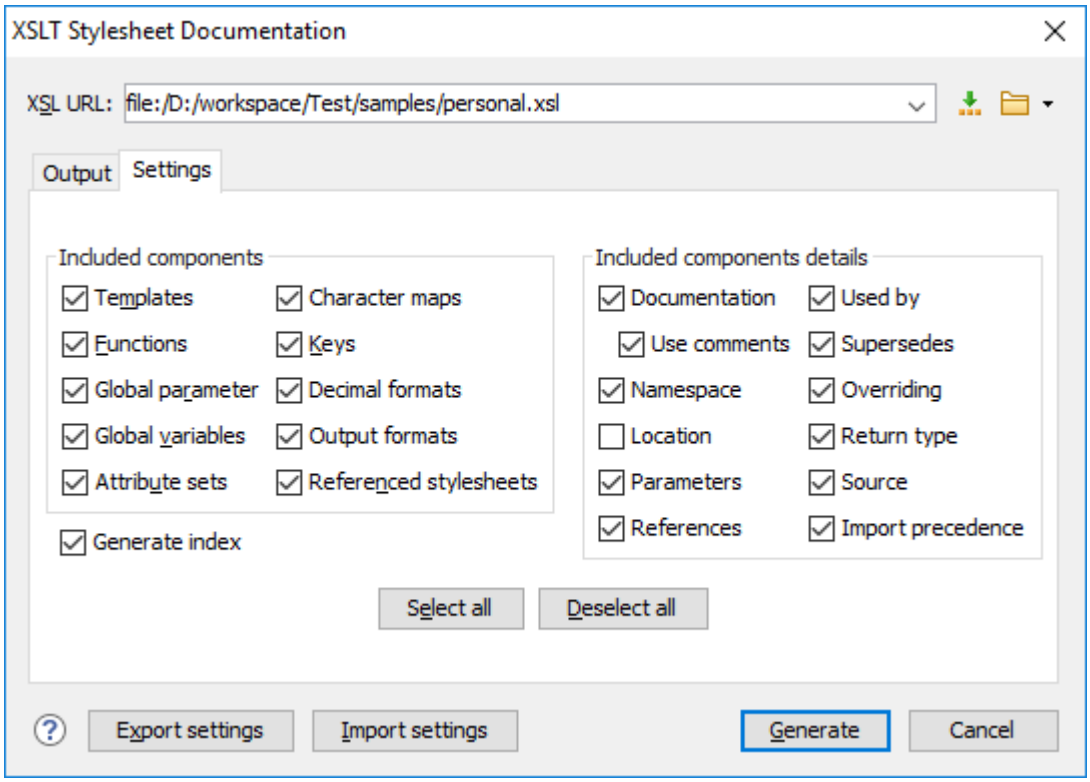
- **Output file** - You can specify the path of the output file by entering it in the text field, or by using the **Insert Editor Variables** button or the options in the **Browse** drop-down menu.
- **Split output into multiple files** - Instructs the application to split the output into multiple files. 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 79)*, 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 498. 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 Developer 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 2305)*. 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 671)* instead of the built-in HTML format *(on page 668)* 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 2263)*.

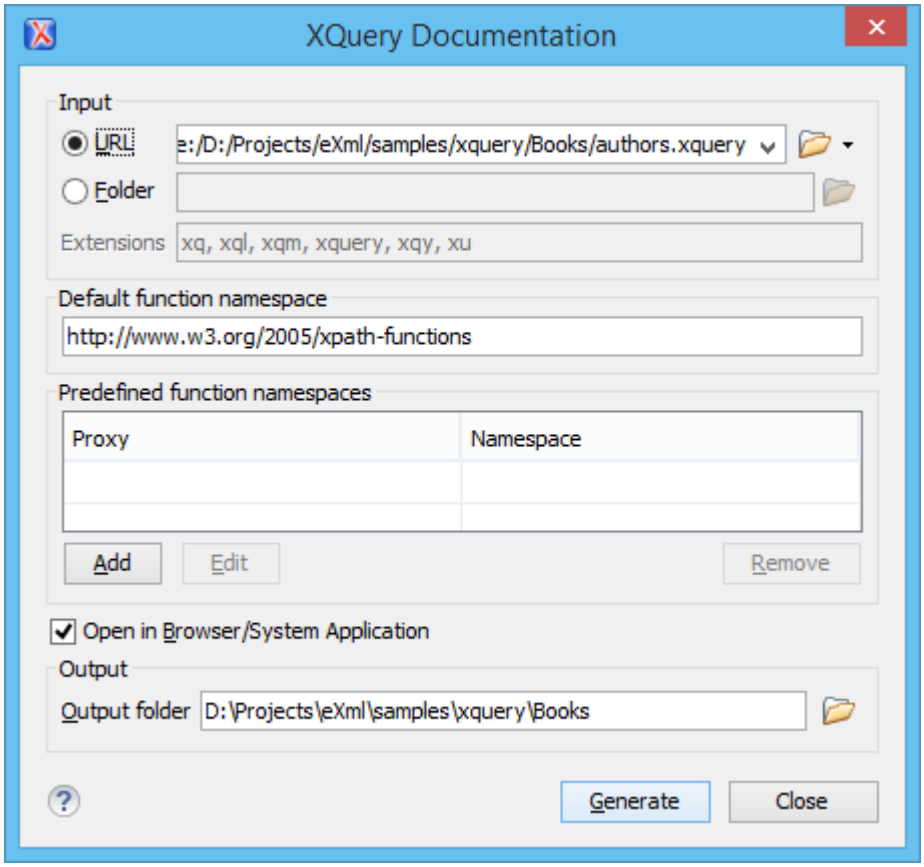
Related Information:
XSLT Stylesheet Component Documentation Support *(on page 652)*

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 334)*. 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 499. 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:
  - **URLFile** - 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 79)*, 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.

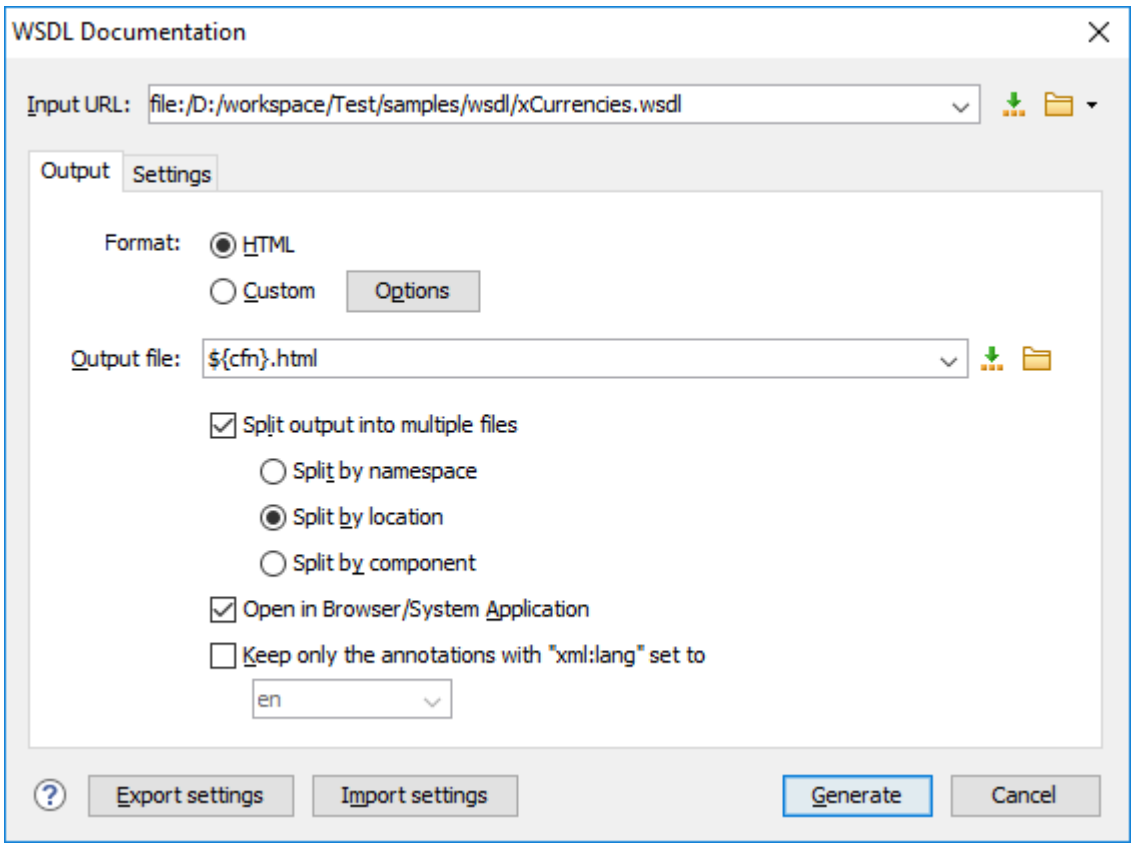
Generating Documentation for WSDL Documents (Deprecated)

You can use Oxygen XML Developer 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 813)* 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 334)*. You can also open the tool by using the **Generate Documentation** toolbar button.

Figure 500. 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 812)*.
  - **Custom** - The documentation is generated in a custom output format *(on page 813)*, 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.

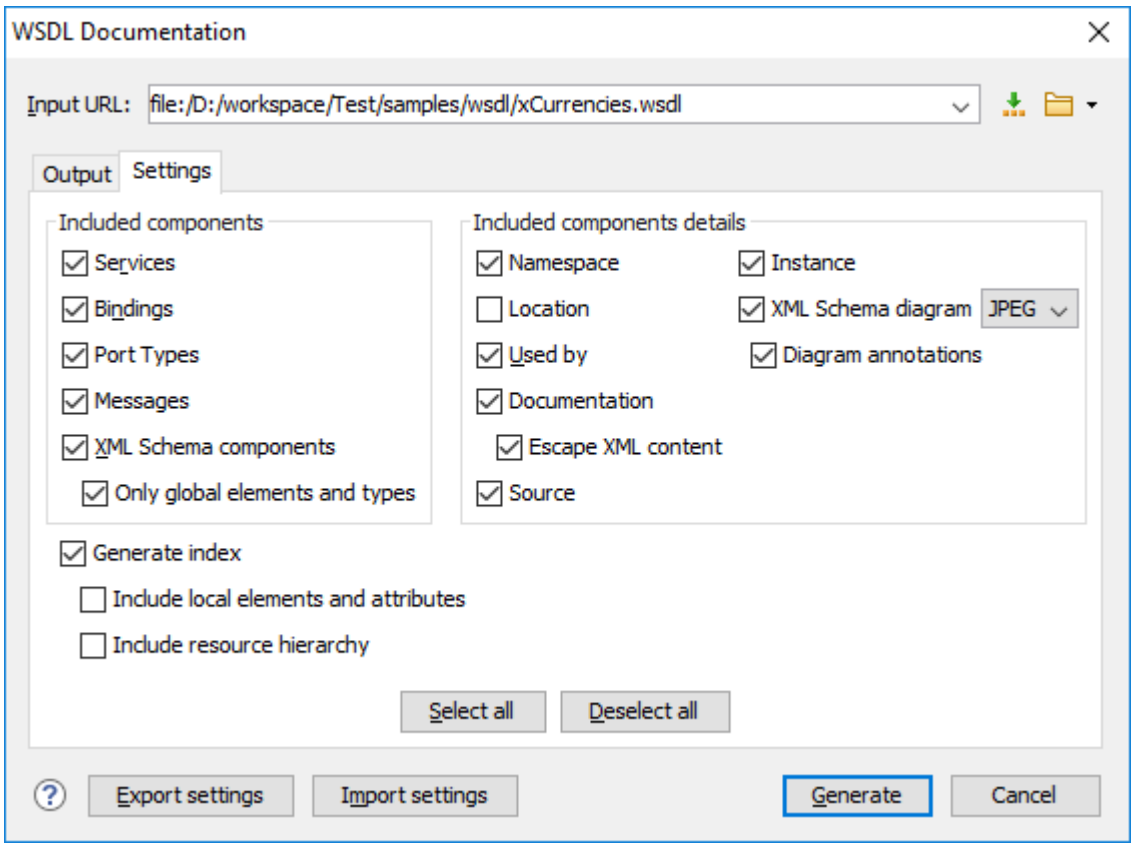
  > **Note:**
  > To set the browser or system application that will be used, open the **Preferences** dialog box **(Options > Preferences)** *(on page 79)*, go to **Global**, and set it in the **Default Internet browser** field. This will take precedence over the default system application settings.

- **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 501. 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 2263)*.

Generating JSON Schema Documentation

Oxygen XML Developer 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 Developer presents a dialog box asking if you want to install it. Once installed, you need to restart Oxygen XML Developer and the **JSON Schema Documentation** action will invoke the tool.

Add-on documentation and installation details.

Generating OpenAPI Documentation

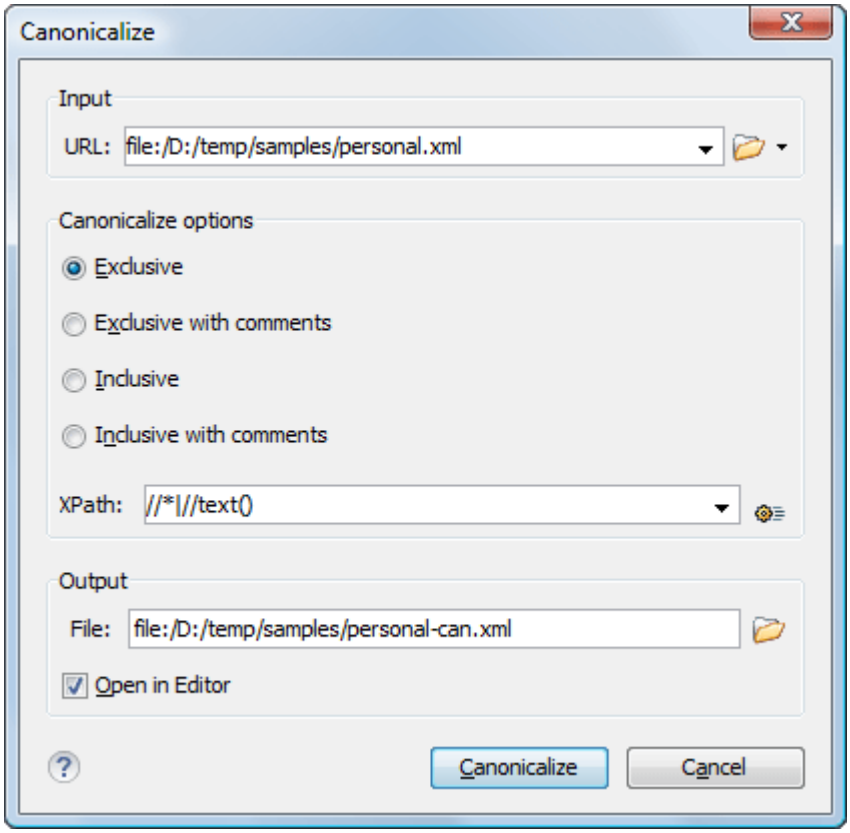
Oxygen XML Developer includes a tool for generating documentation for *OpenAPI* 3.0, or 3.1 documents in either JSON or YAML format, including annotations and cross references. The documentation displays information about the servers, paths, components and tags defined in the OpenAPI documents and you can choose whether the output is presented in HTML (with various sections, hyperlinks, and filtering options), DITA, or PDF.

Addon documentation and installation details.

Canonicalizing Files

You can select the *canonicalization (on page 2303)* 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.

Figure 502. Canonicalization Settings Dialog Box



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 2303)* 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 2303)* method is used.
- **Inclusive** - If selected, the inclusive (uncommented) *canonicalization (on page 2303)* 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.

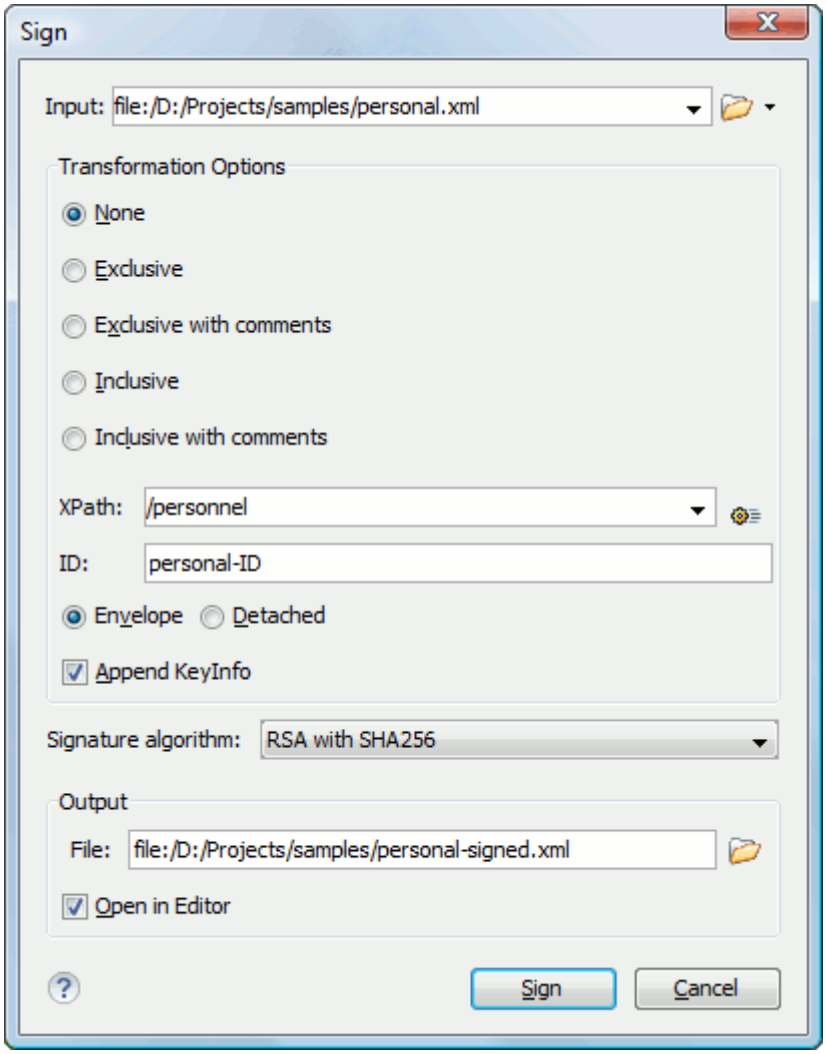
- **Inclusive with comments** - If selected, the inclusive with comments *canonicalization (on page 2303)* method is used.
- **XPath** - The XPath expression provides the fragments of the XML document to be signed.
- **Output** - Available if the **Canonicalize** action was selected from the **Tools** 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 617)*

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 503. Signature Settings Dialog Box



The following options are available:

Note:
If Oxygen XML Developer 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 199)* 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 617)* section for more information about these options.

- **None** - If selected, no *canonicalization (on page 2303)* algorithm is used.
- **Exclusive** - If selected, the exclusive (uncommented) *canonicalization (on page 2303)* 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 2303)* method is used.
- **Inclusive** - If selected, the inclusive (uncommented) *canonicalization (on page 2303)* 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 2303)* 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 617)* for more information.
- **Detached** - If selected, the *detached* signature is used. See the Digital Signature Overview *(on page 617)* 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**, **RSA with SHA512**, **ECDSA with SHA1**, **ECDSA with SHA256**, **ECDSA with SHA384**, and **ECDSA 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 Developer.

Related Information:
Digital Signatures Overview *(on page 617)*
Verifying Signature *(on page 625)*
Example of How to Digitally Sign XML Files or Content *(on page 625)*

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 617)*
Signing Files *(on page 622)*
Example of How to Digitally Sign XML Files or Content *(on page 625)*

WSDL SOAP Analyzer Tool (Deprecated)

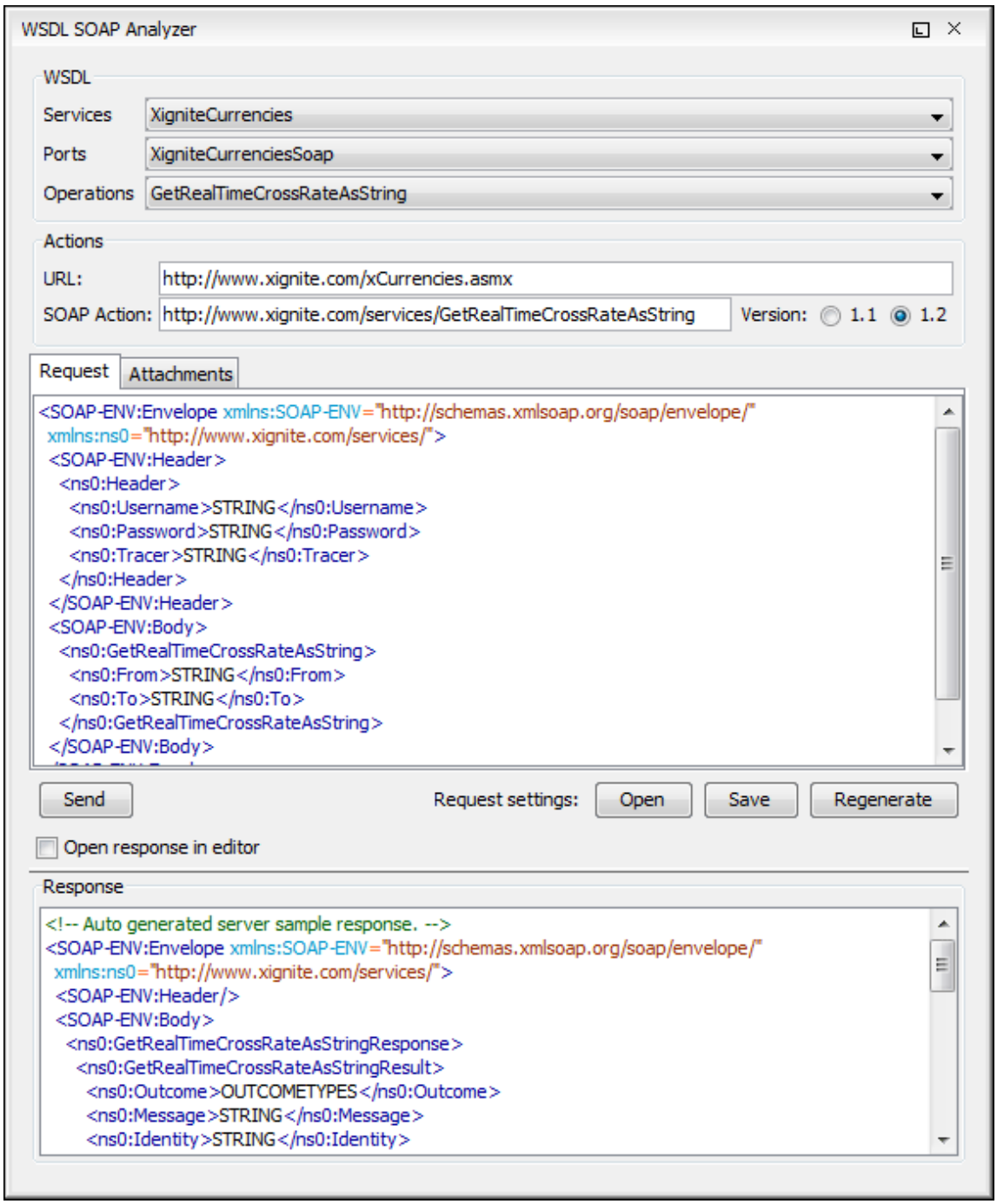
WSDL SOAP Analyzer is a tool that helps you test if the messages defined in a Web Service Descriptor (WSDL) are accepted by a Web Services server.

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 Developer 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 334)* view.

Figure 504. WSDL SOAP Analyzer Dialog Box



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 Developer 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 2303)* 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 Developer 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 Developer 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 Developer 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 816)* 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:

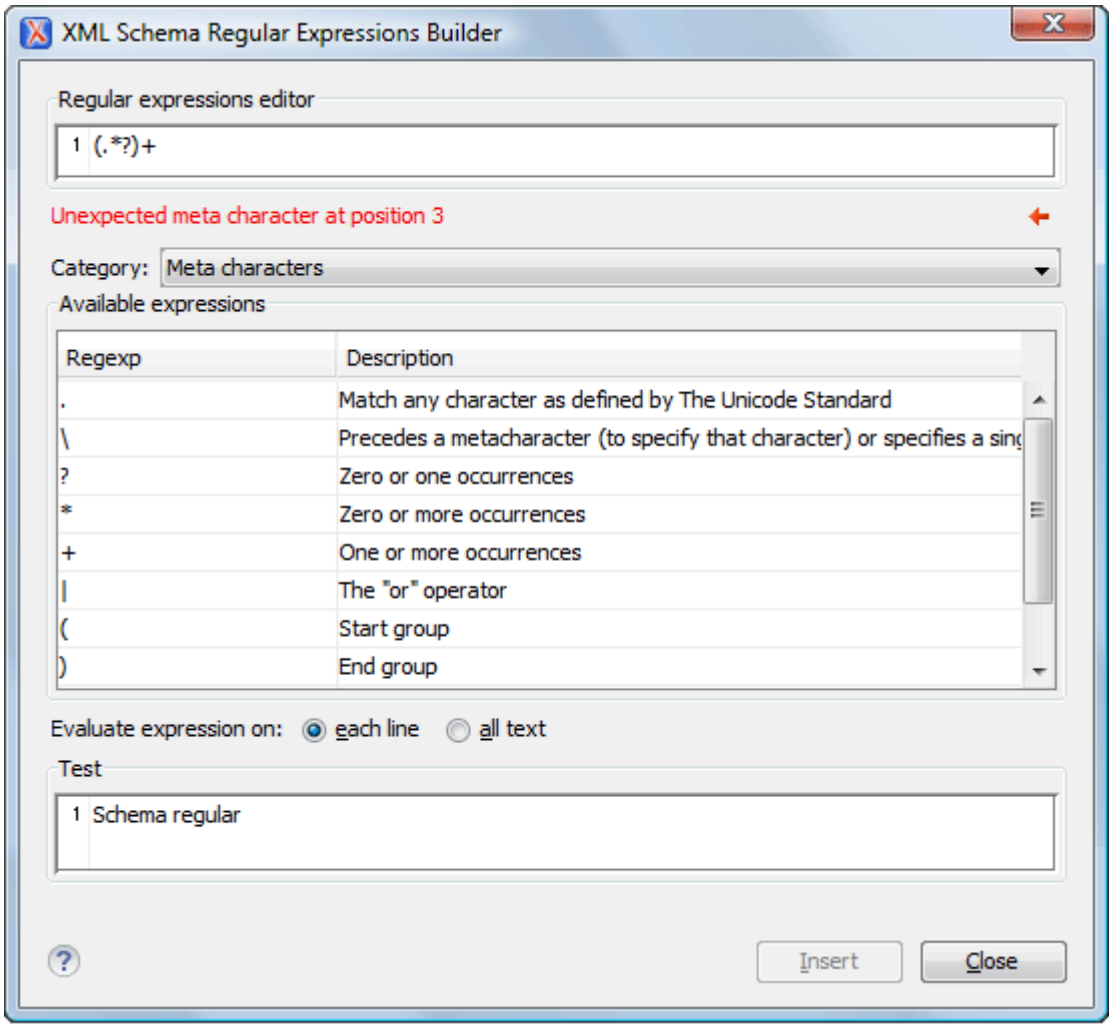
1. Go to **Tools >** **WSDL SOAP Analyzer** .
2. On the **WSDL File** tab enter the URL of the remote WSDL file.

3. Click the **OK** button.
   This will open the **WSDL SOAP Analyzer** tool *(on page 813)*. 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 505. 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**.

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 279)* or schema **Design** mode *(on page 280)*. 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

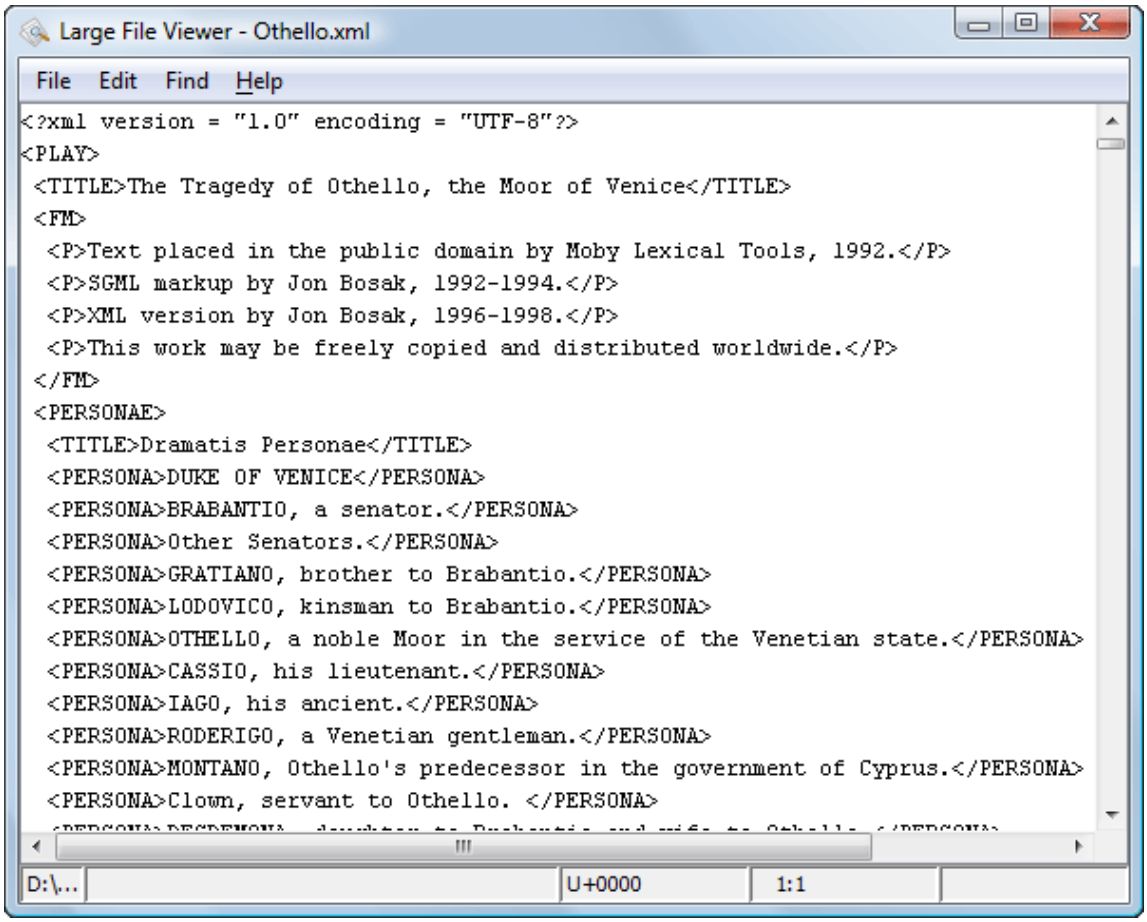
XML files tend to become larger and larger mostly because they are frequently used as a format for database export or for porting between multiple database formats. Traditional XML text editors simply cannot handle opening these huge export files, some having sizes exceeding one gigabyte, because all 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 Developer options. See Encoding for non-XML files *(on page 123)*.

Figure 506. 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 379)*.
- **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.

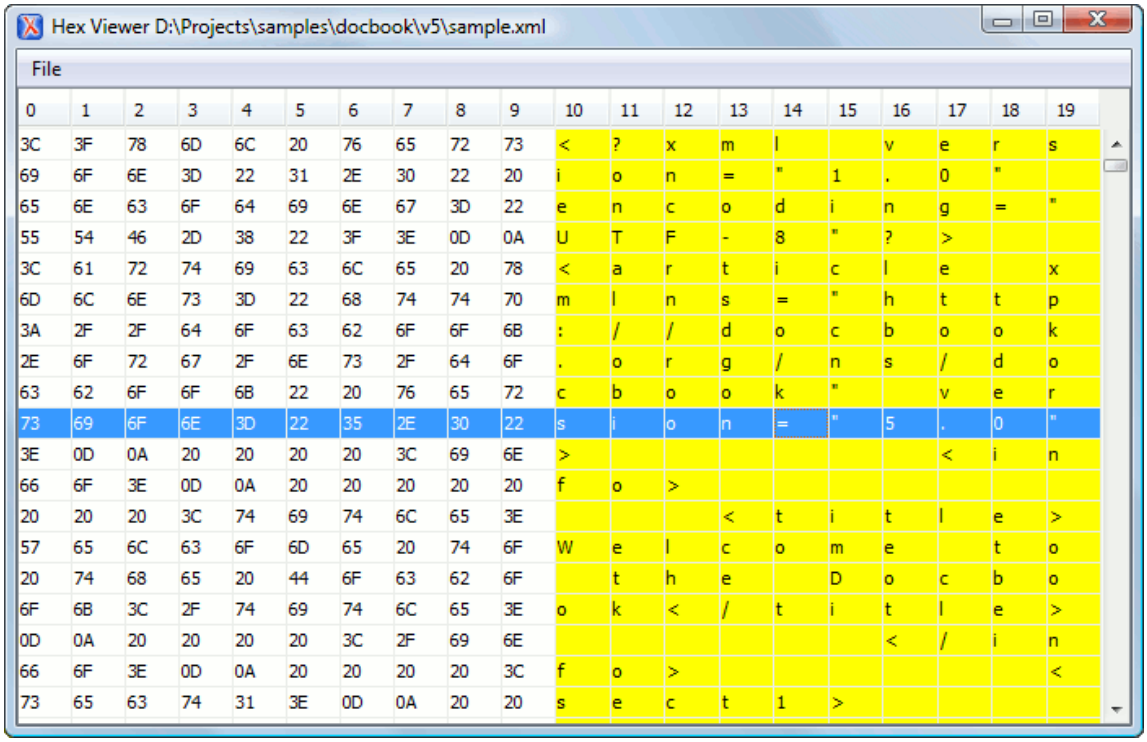
Attention:
For faster computation the **Large File Viewer** uses a fixed font (plain, monospace font of size 12) to display characters. The font is *not* configurable from the **Preferences** page *(on page 79)*.

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 507. Hex Viewer



0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
3C	3F	78	6D	6C	20	76	65	72	73	<	?	x	m	l		v	e	r	s
69	6F	6E	3D	22	31	2E	30	22	20	i	o	n	=	"	1	.	0	"	
65	6E	63	6F	64	69	6E	67	3D	22	e	n	c	o	d	i	n	g	=	"
55	54	46	2D	38	22	3F	3E	0D	0A	U	T	F	-	8	"	?	>		
3C	61	72	74	69	63	6C	65	20	78	<	a	r	t	i	c	l	e		x
6D	6C	6E	73	3D	22	68	74	74	70	m	l	n	s	=	"	h	t	t	p
3A	2F	2F	64	6F	63	62	6F	6F	6B	:	/	/	d	o	c	b	o	o	k
2E	6F	72	67	2F	6E	73	2F	64	6F	.	o	r	g	/	n	s	/	d	o
63	62	6F	6F	6B	22	20	76	65	72	c	b	o	o	k	"		v	e	r
73	69	6F	6E	3D	22	35	2E	30	22	s	i	o	n	=	"	5	.	0	"
3E	0D	0A	20	20	20	20	3C	69	6E	>							<	i	n
66	6F	3E	0D	0A	20	20	20	20	20	f	o	>							
20	20	20	3C	74	69	74	6C	65	3E				<	t	i	t	l	e	>
57	65	6C	63	6F	6D	65	20	74	6F	W	e	l	c	o	m	e		t	o
20	74	68	65	20	44	6F	63	62	6F		t	h	e		D	o	c	b	o
6F	6B	3C	2F	74	69	74	6C	65	3E	o	k	<	/	t	i	t	l	e	>
0D	0A	20	20	20	20	3C	2F	69	6E							<	/	i	n
66	6F	3E	0D	0A	20	20	20	20	3C	f	o	>							<
73	65	63	74	31	3E	0D	0A	20	20	s	e	c	t	1	>				

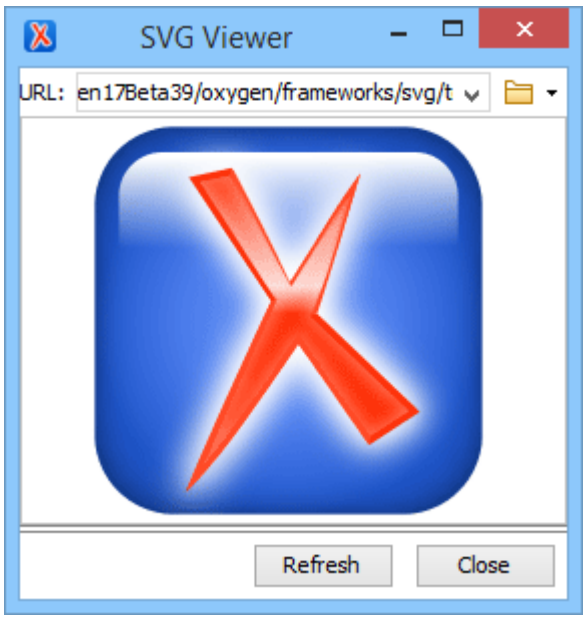
To open a file in **Hex Viewer** use the **File > Open** action. Alternatively, you can drag a file and drop it in the **Hex Viewer** panel.

Standalone SVG Viewer

Oxygen XML Developer includes a simple **SVG Viewer** that allows you to work with SVG images.

To open the viewer, select **SVG Viewer** from the **Tools** menu.

Figure 508. SVG Viewer



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 334)* 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 in 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 1006)*

Tree Editor (Deprecated)

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 Developer **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.

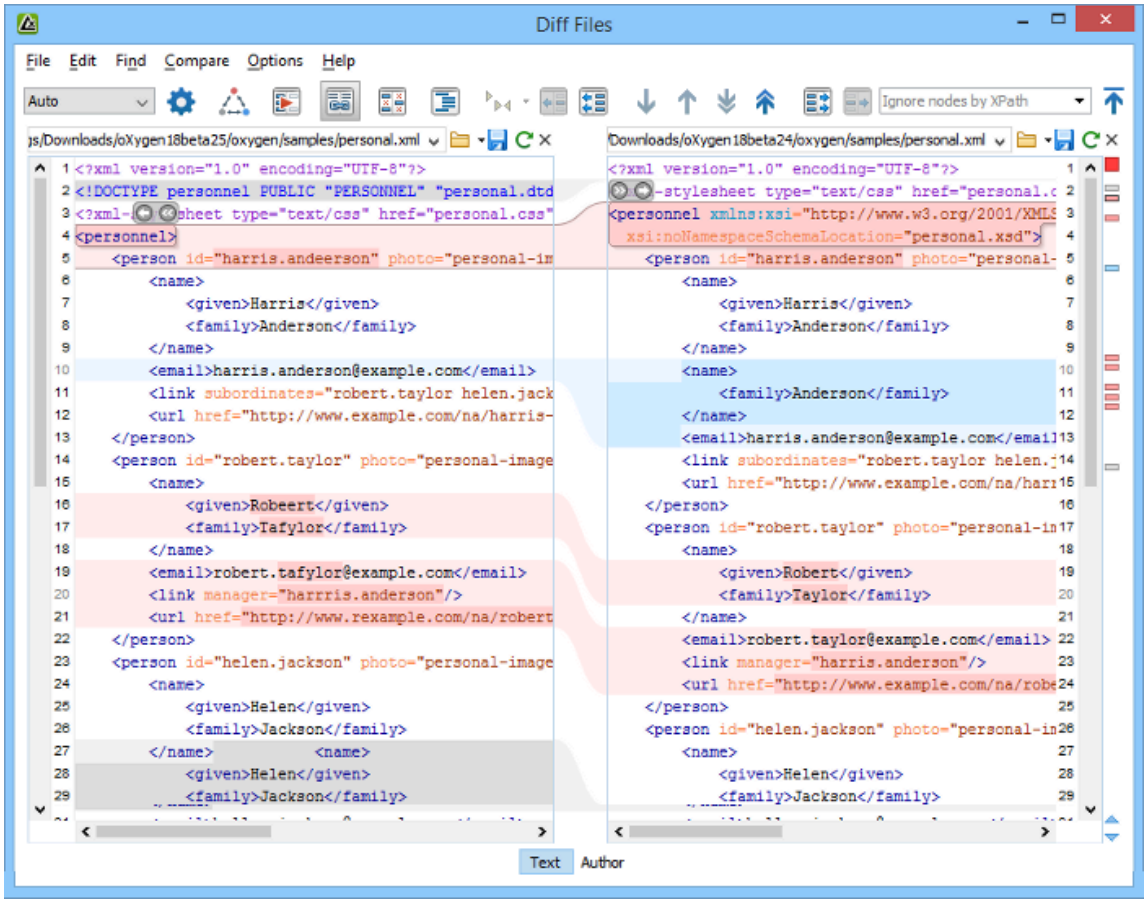
Comparison Tools

Oxygen XML Developer includes some useful tools for comparing files and directories. These tools are found in the **Comparison Tools** submenu within the **Tools** menu.

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 > Comparison Tools** menu or can be opened as a stand-alone application from the Oxygen XML Developer installation folder (`diffFiles.exe`).

Figure 509. 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.

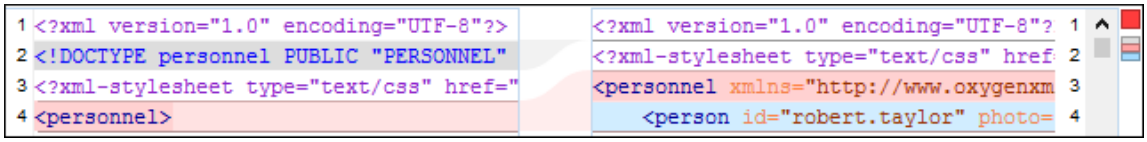
   Step Result: The selected files are opened in the two side-by-side editors. A text editing mode is used to offer a better view of the differences.

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 2015)* 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 510. 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 217)*, 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 413)* and in the various menus *(on page 416)* (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 Developer 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.
- **Words** - Computes the differences at word level, meaning that it compares two files or fragments looking for identical words.
- **Lines** - Computes the differences at line level, meaning that it compares two files or fragments looking for identical lines of text.
- **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.

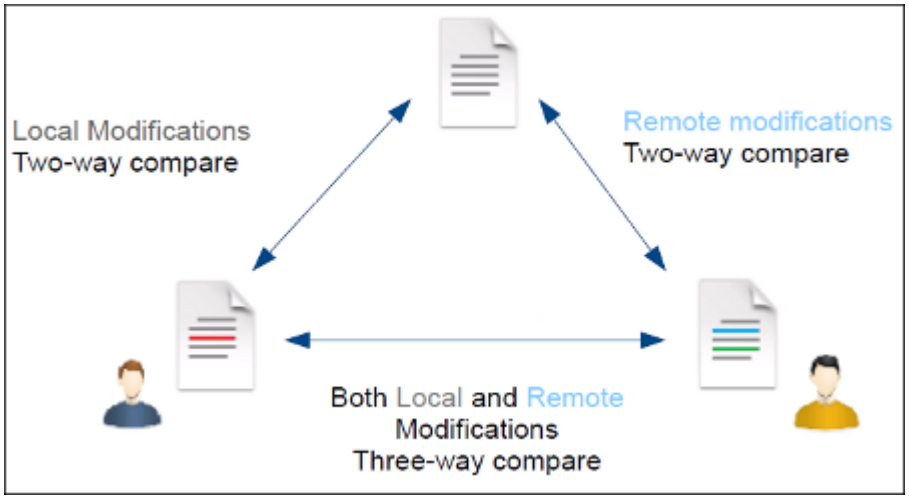
- **XML Fast** - Comparison that works well on large files or fragments, but it is less precise than **XML Accurate**.
- **XML Accurate** - Comparison that is more precise than **XML Fast**, at the expense of speed. It compares two XML files or fragments looking for identical XML nodes.

Three-Way Comparisons

Oxygen XML Developer 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 511. Three-Way Comparison



Compare Files

To perform a three-way comparison, follow these steps:

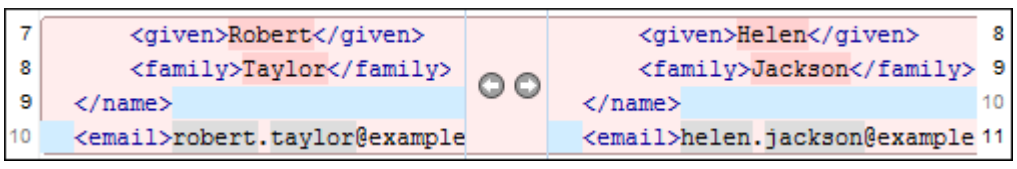
1. Open a file in the left panel and the file you want to compare it to in the right panel. You can specify the path by using the text field, the history drop-down, or the browsing actions in the **Browse** drop-down menu.

   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 2015)* 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 512. 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 217)*, 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 413)* and in the various menus *(on page 416)* (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

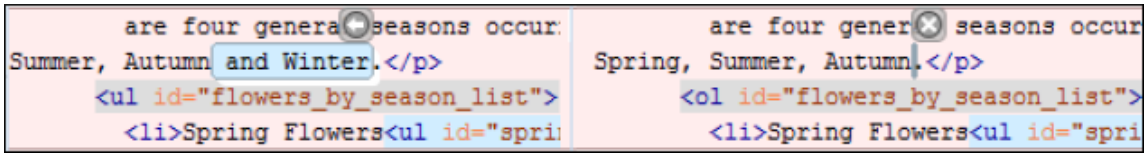
Oxygen XML Developer offers the following three-way diff algorithms to compare files:

- **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.
- **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 Developer 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 2015)*, 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.

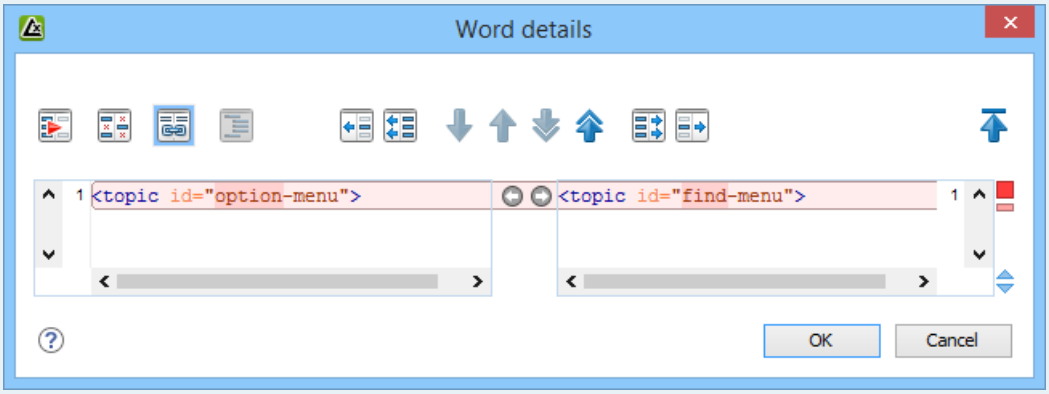
Figure 513. Second-Level Diff Comparison



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.

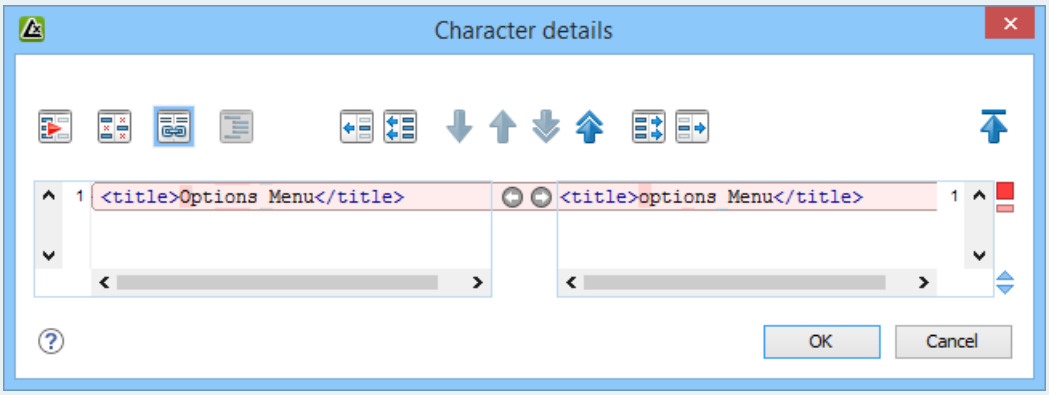
To do a word level comparison, select **Show word level details** from the contextual menu or **Compare** menu.

Figure 514. Word Level Comparison



To do a character level comparison, select **Show Character Level details** from the contextual menu or **Compare** menu.

Figure 515. Character Level Comparison



Related information

Files Comparison Preferences Page *(on page 215)*

Compare Directories Tool *(on page 421)*

Toolbar and Contextual Menu Actions of the Compare Files Tool *(on page 413)*

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 405)*. If only two files are specified, the tool will start in the 2-way comparison mode *(on page 402)*. 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
-ext $LOCAL $REMOTE $BASE $MERGED'
git config --global mergetool.oxygendiff.trustExitCode true
```

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.

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 *Oxygen* **Compare Files** tool provides the following advantages when using it with Sourcetree:

- The files are presented side-by-side to make it much easier to determine real changes.
- XML comparison algorithms are available.

- Various options can be used to configure the comparison.
- You can navigate through changes, one by one.

To integrate the **Compare Files** tool with Sourcetree, use the following 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* `diffFiles.exe` file (for example: `c:\Programs\Oxygen XML Editor\diffFiles.exe`).
   - **Arguments** - Enter **-ext $REMOTE $LOCAL $LOCAL**.
   - **Merge Tool** - Select **Custom**.
   - **Diff Command** - Enter the path of the *Oxygen* `diffFiles.exe` file (for example: `c:\Programs\Oxygen XML Editor\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

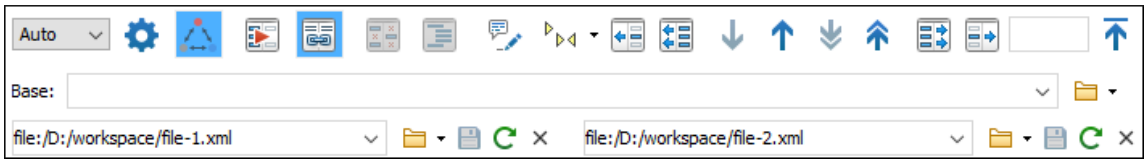
1. In Sourcetree, go to **Sourcetree > Preferences**.
2. Go to the **Diff** tab.
3. In the **External Diff/Merge** section, configure the settings as follows:
   - **External Diff Tool** - Select **Custom**.
   - **Diff Command** - Enter a command-line argument to launch the *Oxygen* `diffFiles.sh` file (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 516. 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.
- **Words** - Computes the differences at word level, meaning that it compares two files or fragments looking for identical words.
- **Lines** - Computes the differences at line level, meaning that it compares two files or fragments looking for identical lines of text.
- **Syntax Aware** - Computes differences for the file types or fragments known by Oxygen XML Developer, taking the syntax (the specific types of tokens) into consideration.
- **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 215)* 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.

 **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. If content is selected in the left panel, only that selection is affected, copying only the corresponding changes from the right to the selection on the left. The action name is also automatically updated to **Copy Changes in Selection from Right to Left** in this case.

 **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.

 **Copy All Changes from Left to Right**

Copies all changes from the file in the left panel to the file in the right panel. If content is selected in the right panel, only that selection is affected, copying only the corresponding changes from the left to the selection on the right. The action name is also automatically updated to **Copy Changes in Selection from Left to Right** in this case.

 **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 1726)* 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 217)* 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 405)*. 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 Developer 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 405)*.

Base > Open URL

Browses for a remote file that will be compared with both files in a three-way comparison *(on page 405)*.

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 405)*.

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 436)*.

Save Comparison as Document with Tracked Changes (Available for two-way comparison in Author mode only)

Allows you to merge two compared documents based on the differences detected and save the results as a specified file that includes the special change tracking marks. You can load the resulting file in **Oxygen's Author** mode to review the changes that resulted from the merge process and you can accept or reject them. Note that if the documents to be compared already contain tracked changes, they will be automatically accepted before generating the output file.

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. If content is selected in the left panel, only that selection is affected, copying only the corresponding changes from the right to the selection on the left. The action name is also automatically updated to **Copy Changes in Selection from Right to Left** in this case.

Copy All Changes from Left to Right

Copies all changes from the file in the left panel to the file in the right panel. If content is selected in the right panel, only that selection is affected, copying only the corresponding changes from the left to the selection on the right. The action name is also automatically updated to **Copy Changes in Selection from Left to Right** in this case.

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 Developer 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 Developer Support Center web page in a browser.

How to Compare and Merge Documents with Change Tracking Highlights

Oxygen XML Developer includes two actions (**Merge Documents with Change Tracking Highlights** and a **Merge Directories with Change Tracking Highlights**) within the **Tools > Comparison Tools** menu that can be used to compare and merge content and the comparison results are saved as documents with highlighted tracked changes that can be later reviewed and accepted or rejected.

This topic provides instructions and several different use cases for how to get the most out of comparing and merging documents with change tracking highlights present.

Use Case 1: Review Changes Made to an XML Document

The **Merge Documents with Change Tracking Highlights** tool *(on page 2039)* can be used to merge two XML documents, based on a 2-way mode comparison. The files involved in the process are merged by saving the comparison results as documents with highlighted tracked changes.

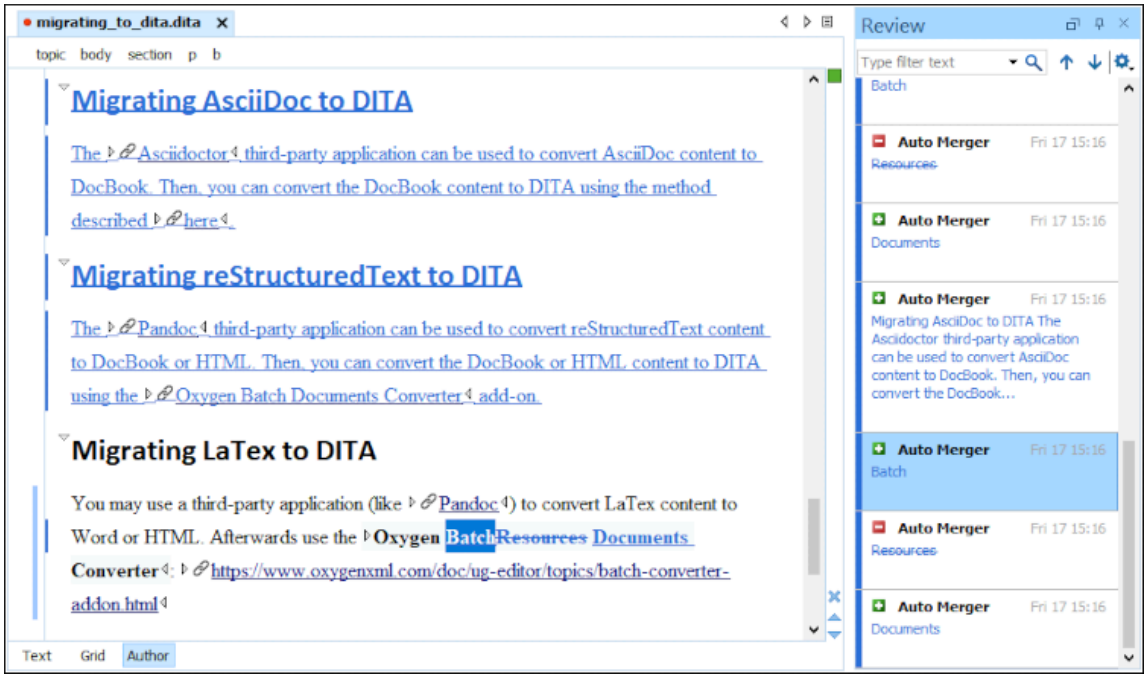
Here is the basic procedure:

1. Select the **Merge Documents with Change Tracking Highlights** action that is found in the **Tools > Comparison Tools** menu to invoke the tool. It opens a dialog box where you can specify the documents to merge.
2. In the resulting dialog box, you need to provide the URLs of the base version of the document (**Base document**) and the modified version (**Document to merge with**), and then click the **Merge** button.

   Step Result: The resulting document is automatically loaded into a new editor tab. It incorporates special markings for the tracked changes/differences between the 2 compared and analyzed XML files.
3. Switch to the **Author** mode and then open the **Review** view (click the  **Manage reviews** button on the toolbar or select it from the **Window > Show View** menu).

   Step Result: All detected changes are listed in the **Review** view and they are marked accordingly in the resulting document as well.
4. You can analyze each detected change one by one and accept or reject them individually or in bulk.
5. Save the changes you made during the reviewing process.

Figure 517. Revised Merged Document

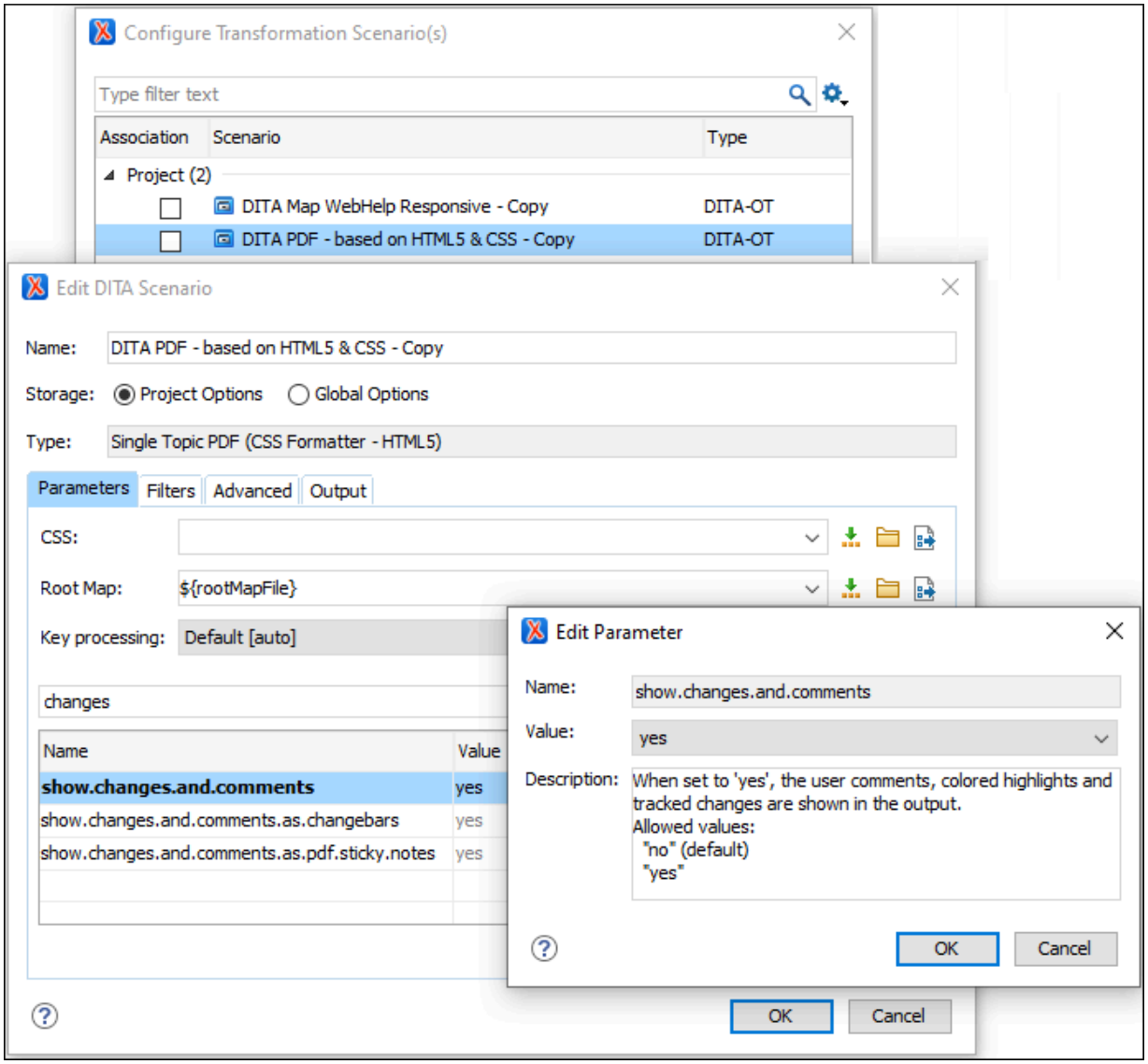


Use Case 2: Compare Differences Between Two Versions

Another interesting way of using the resulting merged document that incorporates the change tracking marks between 2 versions is to apply a transformation of the resulting document into a PDF file that highlights the changes between the compared versions.

Here is the basic procedure (for this particular example, suppose the resulting merged document is a DITA file):

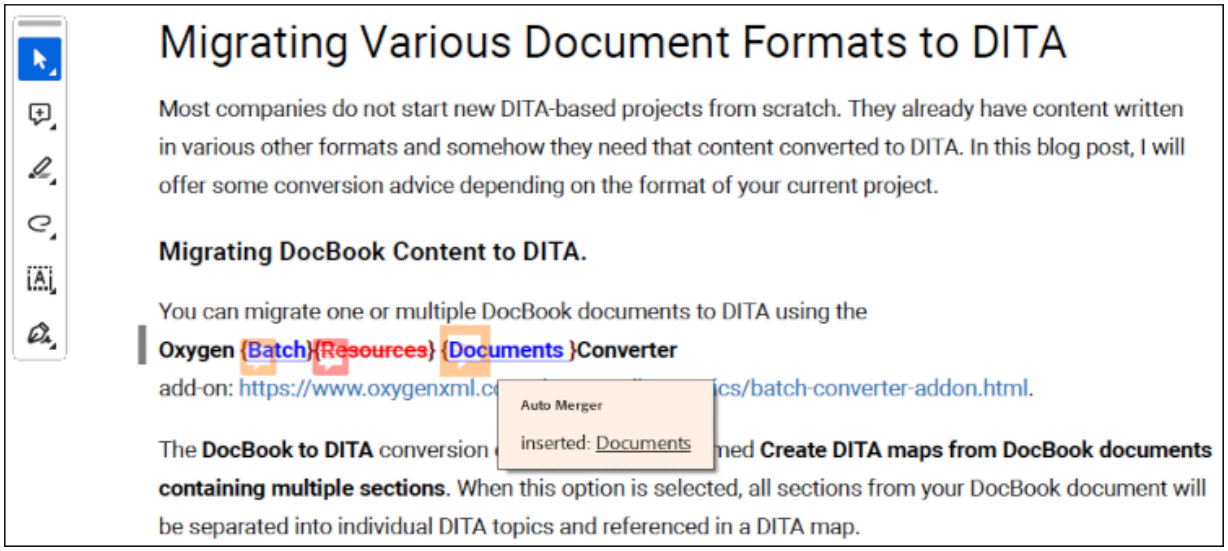
1. Select the **Merge Documents with Change Tracking Highlights** action that is found in the **Tools > Comparison Tools** menu to invoke the tool. It opens a dialog box where you can specify the documents to merge.
2. In the resulting dialog box, you need to provide the URLs of the base version of the document (**Base document**) and the modified version (**Document to merge with**), and then click the **Merge** button.
3. With the resulting merged document open in the editor, click the **Configure Transformation Scenario(s)** button from the toolbar.
4. Select the default read-only scenario **DITA PDF - based on HTML5 & CSS** and click the **Duplicate** button to create a copy so that you can make a few changes to the default settings.
5. In the resulting **Edit DITA Scenario** dialog box, go to the **Parameters** tab and type "changes" in text filter field above the table of parameters. This will display the three parameters that are responsible for displaying the tracked changes in the resulting PDF and they must all be set to the value of *yes*:
    - `show.changes.and.comments`
    - `show.changes.and.comments.as.changebars`
    - `show.changes.and.comments.as.pdf.sticky.notes`



6. Click **OK**, and with the newly created copy of the scenario selected, click the **Apply associated** button to run the transformation.

   Result: Once the transformation is finished, the resulting PDF is automatically opened in the system's default PDF reader (e.g. Adobe). If you hover the mouse over the presented colored markers, you will notice that the tooltip shows that some are of the type "inserted", while others are of the type "deleted". Text areas and fragments with changes are also marked with gray "change" bars on the left. The

changed content is placed between curly brackets and is colored blue or red depending on the type of change (insertion or deletion).



Use Case 3: Compare and Merge Entire Directories

Another use case is to use the **Merge Directories with Change Tracking Highlights** action to merge entire directories instead of specified pairs of XML files.

Here is the basic procedure (for this particular example, suppose you want to review the changes between two versions of an entire DITA project):

1. Select the **Merge Directories with Change Tracking Highlights** action that is found in the **Tools > Comparison Tools** menu to invoke the tool. It opens a dialog box where you can specify the documents to merge.
2. In the resulting dialog box, you need to provide the URLs of the **Base directory**, the **Directory to merge with**, and the **Output directory**.
3. You can optionally select the **Backup the base directory** option. Not that this option is automatically activated and selected in the following cases:
   - The output directory is not specified (the field is left blank).
   - The specified output directory is the base directory itself.

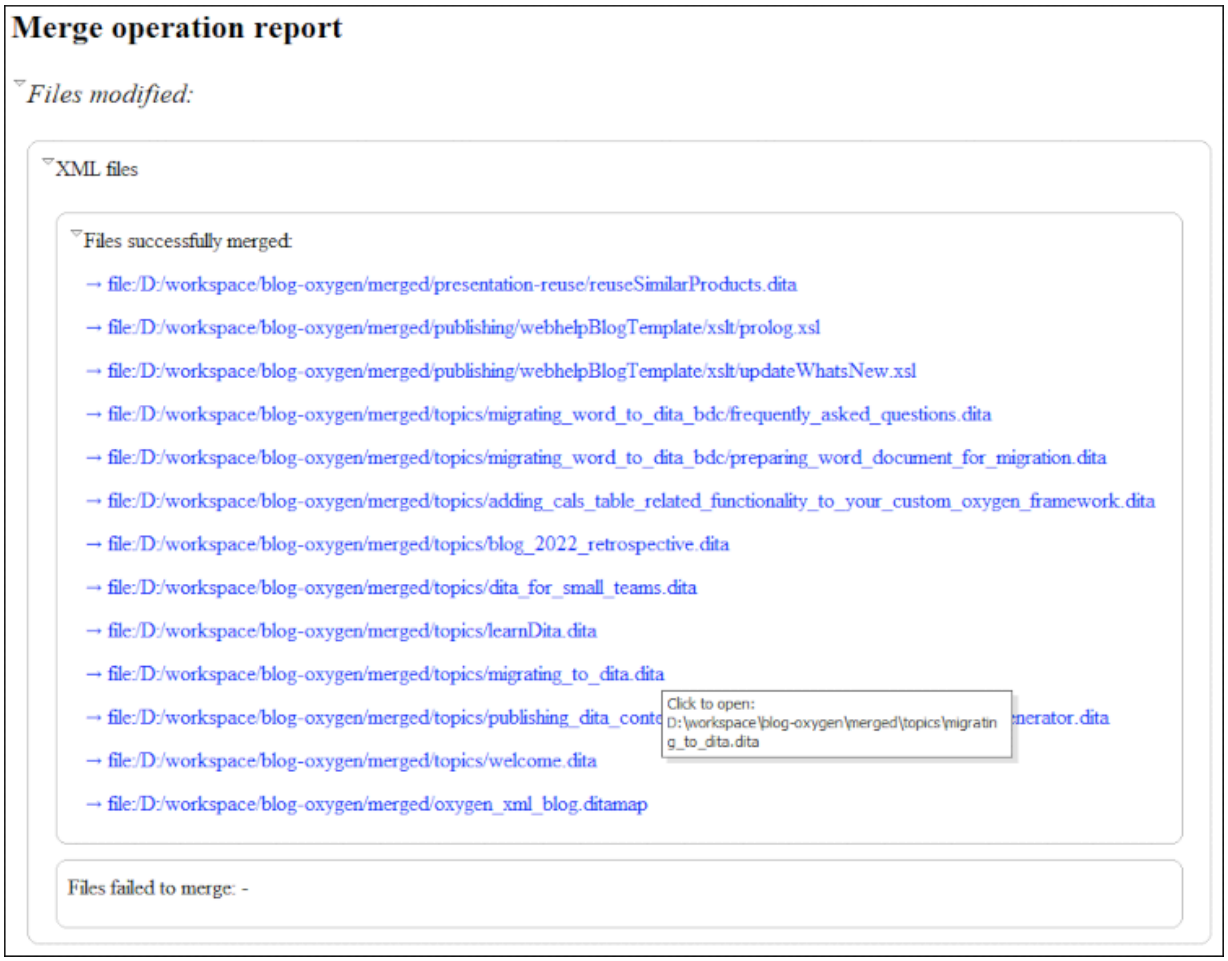
   In these situations, a backup of the base directory will be performed automatically and the backup operation must succeed to proceed with the merge. Otherwise, the merge process is aborted.
4. You can adjust the settings in the **Merge options** section. You can find details for each option in Merge Directories with Change Tracking Highlights: Merge Options *(on page 2058)*. Note that all of these options are enabled by default except for **Create change tracking markers for the XML files to be added**.

   To better understand the merge options presented in this section, note that the merge process has a preliminary phase where the entire structure and the content of the base directory is copied to the output directory.

Since this example is for reviewing the changes between two versions of an entire DITA project, you would select all options in the **Merge options** section, including the **Create change tracking markers for XML files to be added** option.

5. Click the he **Merge** button to start the process of generating the results of the merge operation. Depending on the number of files in the compared directories and the total number of changes detected, the process may take some time.

   Step Result: The operation report should automatically open in **Author** mode. The listed URLs are actually links that will open the corresponding merged DITA file in the editor. You can inspect the changes and accept or reject them, individually or in bulk.



The first group of URLs listed represents the group of XML files detected as modified and merged with change tracking highlights.

The modified non-XML file group follows. Since the **Update the modified non-XML files** option was selected, these type of non-XML files from the **Base directory** were replaced by their corresponding files in the **Directory to merge with**.

Next are the XML files that were initially present only in the **Directory to merge with**. They were also added to the **Output directory**, since the **Add the files found only in the directory to merge with** option was selected.

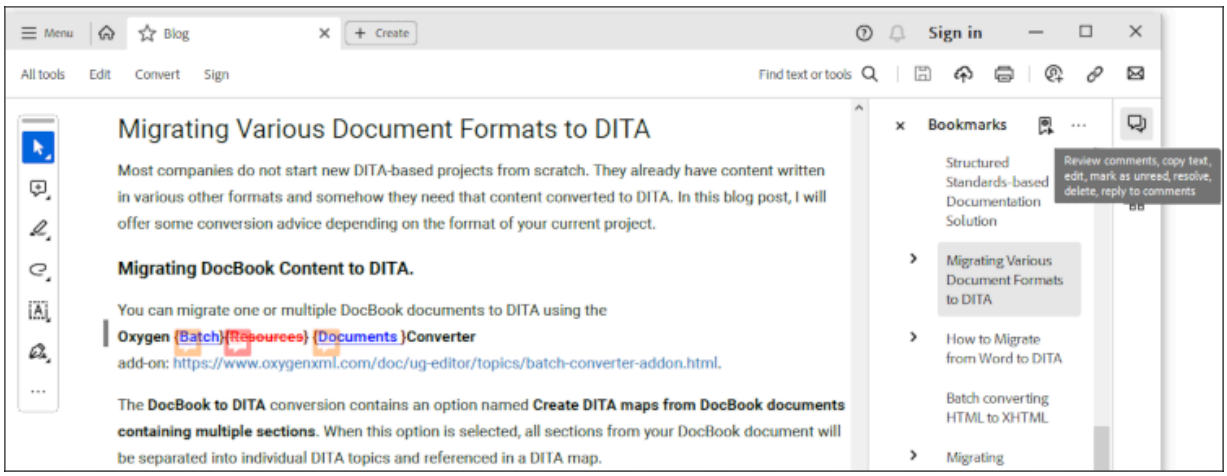
Since the **Create change tracking markers for the XML files to be added** was also selected, change tracking markers of type "entire content added/inserted" were created. Note that it does not make sense to reject that type of change. The **Create change tracking markers for the XML files to be added** option is not intended for the merge process itself, but it is useful if you want to apply various *Oxygen* transformation scenarios to the resulting output directory, as explained later.

Next comes the group of non-XML files initially present only in the **Directory to merge with** and they were also copied to the output directory. Basically, these are resources (usually image or video files) used by the file group listed above. Clicking these types of file links results in opening them in the appropriate application installed on your computer.

The last group listed in the report represents the group of files from the "base directory" that have no longer counterparts in the "directory to merge with". Since the **Delete the files found only in the base directory** option was selected, these files are eventually deleted, even if they were initially copied to the output directory. The links also work for them, opening the associated files from the **Base directory**.

6. Now that you have the directory resulting from the merge operation at your disposal, you can apply a transformation scenario to obtain a PDF document that highlights the changes between the 2 versions. Open the DITA map that resulted from the merge operation in the **DITA Maps Manager**, click the **Configure Transformation Scenario(s)** button in the toolbar of the view, and apply the **DITA Map PDF - based on HTML5 & CSS - Copy** transformation scenario using the same configuration as explained in Use Case 2: Compare Differences Between Two Versions *(on page 2033)*.

   Step Result: Once the transformation is finished, the resulting PDF is automatically opened in the system's default PDF reader (e.g. Adobe).
7. In Adobe, there is a side view identified as **Bookmarks** (on the right or left side, depending of the Adobe version). By selecting **Contents** in this view and then scrolling down, you can hover the mouse over the presented colored markers for the new topics and you will notice that tooltip shows that they are "inserted". You can click on any of these new topics to scroll to the respective topic and you will notice that it is marked with blue in the document (the color used for newly added content). If you continue to scroll down the PDF, you will notice various "insert" and "delete" changes.
8. Click the **Review comments** button on the side stripe of the **Bookmarks** view:



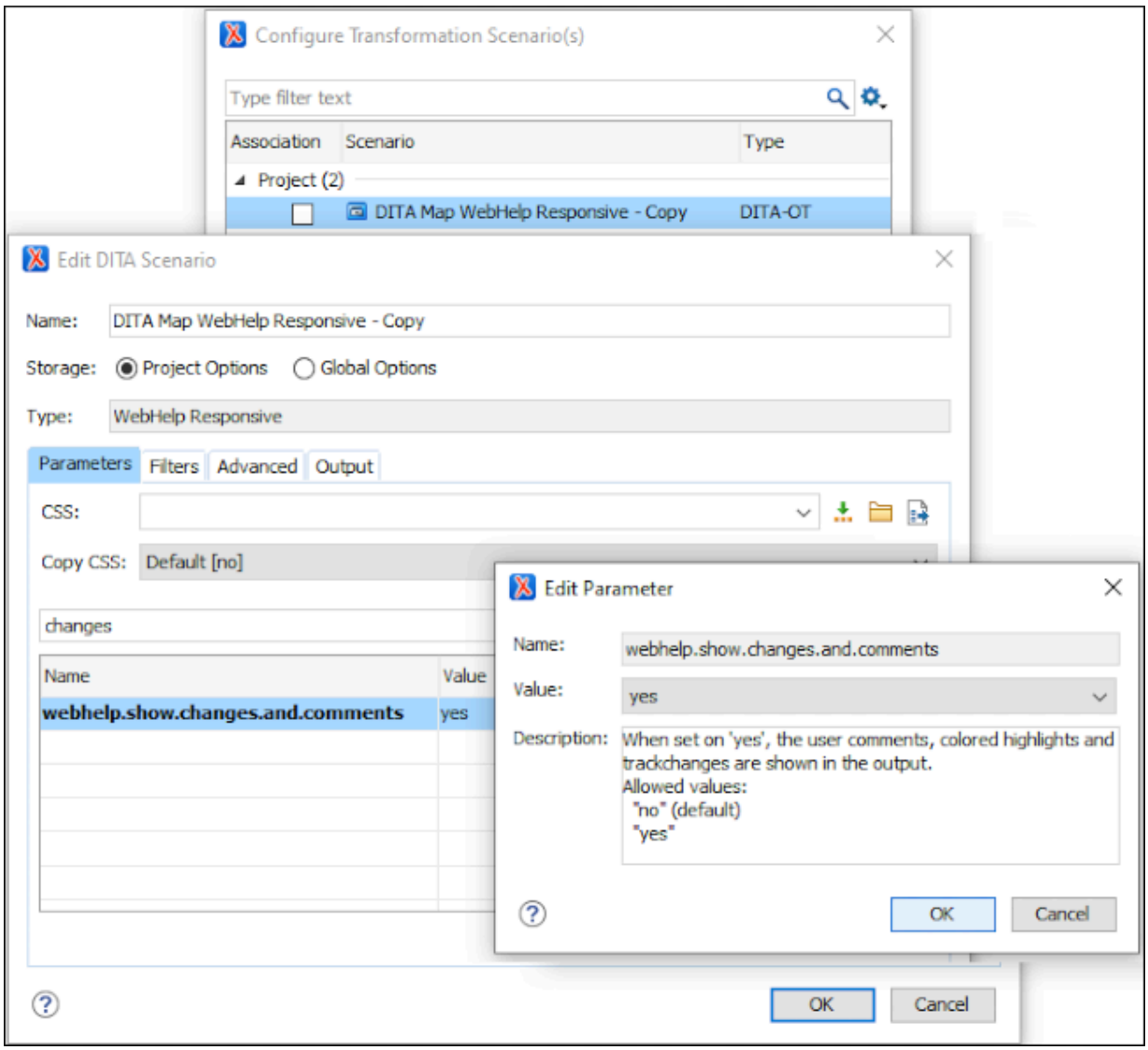
Step Result: A list appears that contains all highlighted changes in the PDF.

9. You can scroll through the list and click on the changes listed to conveniently analyze and review all tracked and highlighted changes between the 2 versions of the DITA project.

WebHelp Responsive Alternative

As an alternative to the **DITA Map PDF - based on HTML5 & CSS** transformation, there are other types of transformations that can be applied to the resulting DITA map, leading to a different kind of output. For example, you can use the **DITA Map WebHelp Responsive** transformation scenario:

1. Select the default read-only scenario **DITA Map WebHelp Responsive** and click the **Duplicate** button to create a copy so that you can make a few changes to the default settings.
2. In the resulting **Edit DITA Scenario** dialog box, go to the **Parameters** tab and type "changes" in text filter field above the table of parameters. Make sure the `webhelp.show.changes.and.comments` parameter is set to *yes*.



3. Click **OK**, and with the newly created copy of the scenario selected, click the **Apply associated** button to run the transformation.

Step Result: Once the transformation finishes, the resulting WebHelp output is opened in your default Internet browser. By clicking on any of the chapters listed, you can browse the respective content and observe how the changes are rendered. The color code is the same (blue for the changes of the type "added" or "inserted", and red for those of the type "deleted"). The yellow markings on the right side also help to navigate through and analyze/review the highlighted changes.

Resources

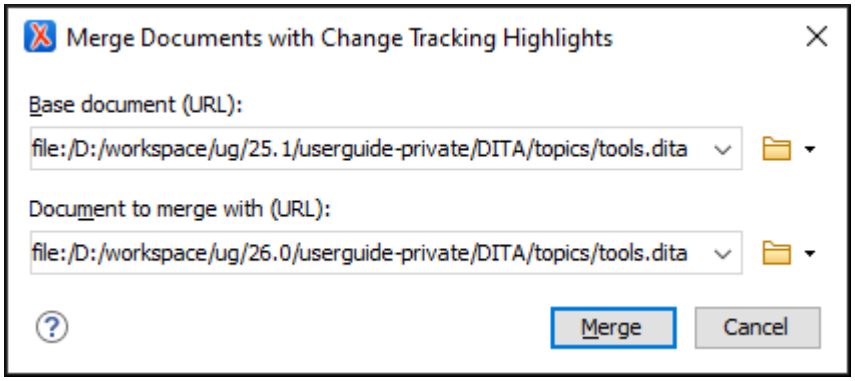
For more information about how to merge documents/directories with change tracking highlights, watch our video demonstration:

Merge Documents with Change Tracking Highlights

The **Merge Documents with Change Tracking Highlights** tool can be used to merge two XML documents (based on a 2-way mode comparison) and the files involved in the process are merged by saving the comparison results as documents with highlighted tracked changes that can later be reviewed and accepted or rejected.

To invoke the tool, select the **Merge Documents with Change Tracking Highlights** action that is found in the **Tools > Comparison Tools** menu. It opens a dialog box where you can specify the documents to merge.

Figure 518. Merge Documents with Change Tracking Highlights Dialog Box



The **Merge Documents with Change Tracking Highlights** dialog box contains the following options:

Base document (URL)

Specifies the URL of the base document.

Directory to merge with (URL)

Specifies the URL of the document to merge with.

After specifying the documents to be merged, click the **Merge** button to trigger the operation.

Once the merge operation is complete, the merged document is opened in a new editor tab. With the merged document open in **Author** mode, the highlighted tracked changes can easily be reviewed and accepted or rejected, thus finishing the entire merge process. Initially, the merged document is untitled and not saved. You

can choose the name when saving it in a location of your choosing (for example, a common use case is to overwrite the base document with the final resulting merged file).

Additional Notes and Limitations:

- When comparing documents in **Author** mode, changes made in the prolog of XML documents (where the XML version and encoding are declared) are not reported and therefore not considered in the merge process.
- Any "doctype" changes are not reported and therefore not considered in the merge process.
- If one or both of the compared XML documents contain tracked changes (represented internally by custom XML processing instructions), all these modifications are automatically accepted prior to the compare and merge procedures.

Resources

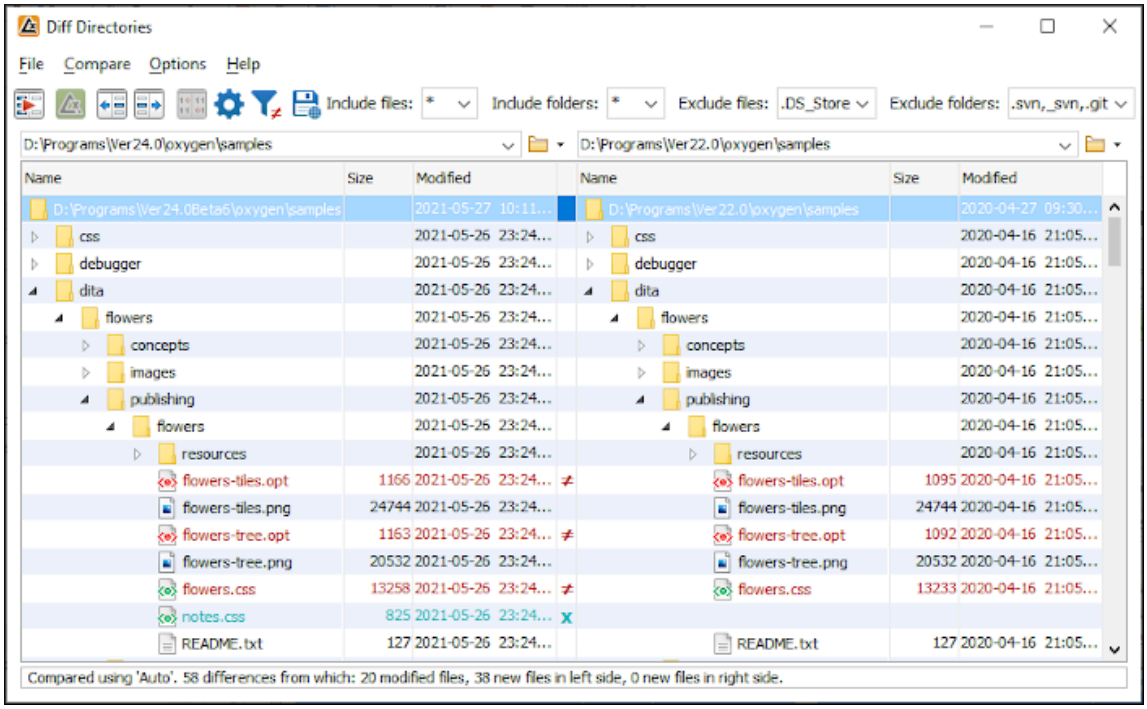
For more information about the merge with change tracking support, see the following resources:

- Video: **Mastering Document Comparisons: A Guide to Generating Tracked Changes Between Two Versions**
- Webinar: **Discover the Power of Diff with Change Tracking**

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 > Comparison Tools** menu or can be opened as a stand-alone application from the Oxygen XML Developer installation folder (`diffDirs.exe`).

Figure 519. Diff Directories Dialog Box



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 218)* 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 Developer treat supported archives as directories, select the **Look in archives** option *(on page 219)* 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 219)*. 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 Developer starts a file comparison *(on page 402)* between the two files, using the **Compare Files** tool.

Related information

Compare Files Tool *(on page 402)*

Compare Directories Script *(on page 2281)*

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 520. Compare toolbar



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 402)* 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 218)* 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 the following combo boxes: **Include files**, **Include folders**, **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. Each filter includes a drop-down menu with the latest 15 filters applied.

For example, to filter out all `JPEG` and `GIF` image files, edit the **Exclude files** filter box to read **`*.jpeg`**, **`*.png`**. To include only a specific sub-folder in the comparison (e.g. `topics/very-recent`), edit the **Include folders** filter to `topi*/ve*rece*`, or `topi*\ve*rece*`, choosing the appropriate **file separator** depending on your operating system.

> **Tip:**
> Instead of directly editing a filtering combo box, you can use **Ctrl+Shift+E (Meta+Shift+E on macOS)** keyboard shortcut when the focus is in the text field to display a specialized filter editor dialog box. You can edit any previously used filter (stored in the history) and create a new one. Furthermore, you can select multiple previous filters and simply combine them (as is) to create a new composite one, or you can edit the result of the multiple selection according to your needs. Also, previous filters can be removed from the history by selecting them and using **Delete (Fn+Backspace on macOS)** on your keyboard.

Contextual Menu Actions

Perform Files Differencing

Opens the **Compare Files** tool *(on page 402)* 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 Developer. 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 Developer.

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 Developer 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 436)*.

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 402)* 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 Developer 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 Developer 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.

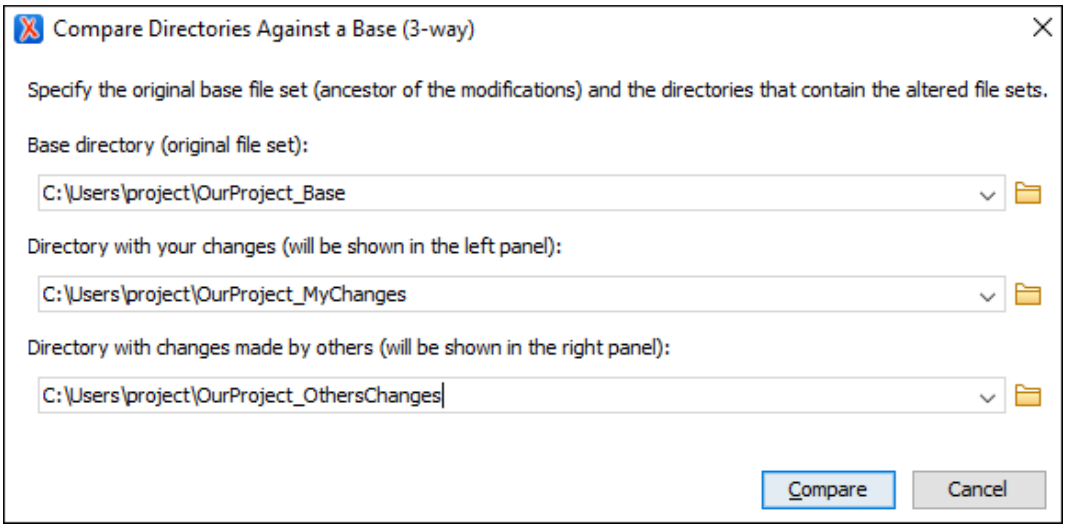
How to Perform 3-Way Directory Comparisons

To perform a 3-way directories comparison, follow these steps:

1. Select **Compare Directories Against a Base (3-way)** from the **Tools > Comparison 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.

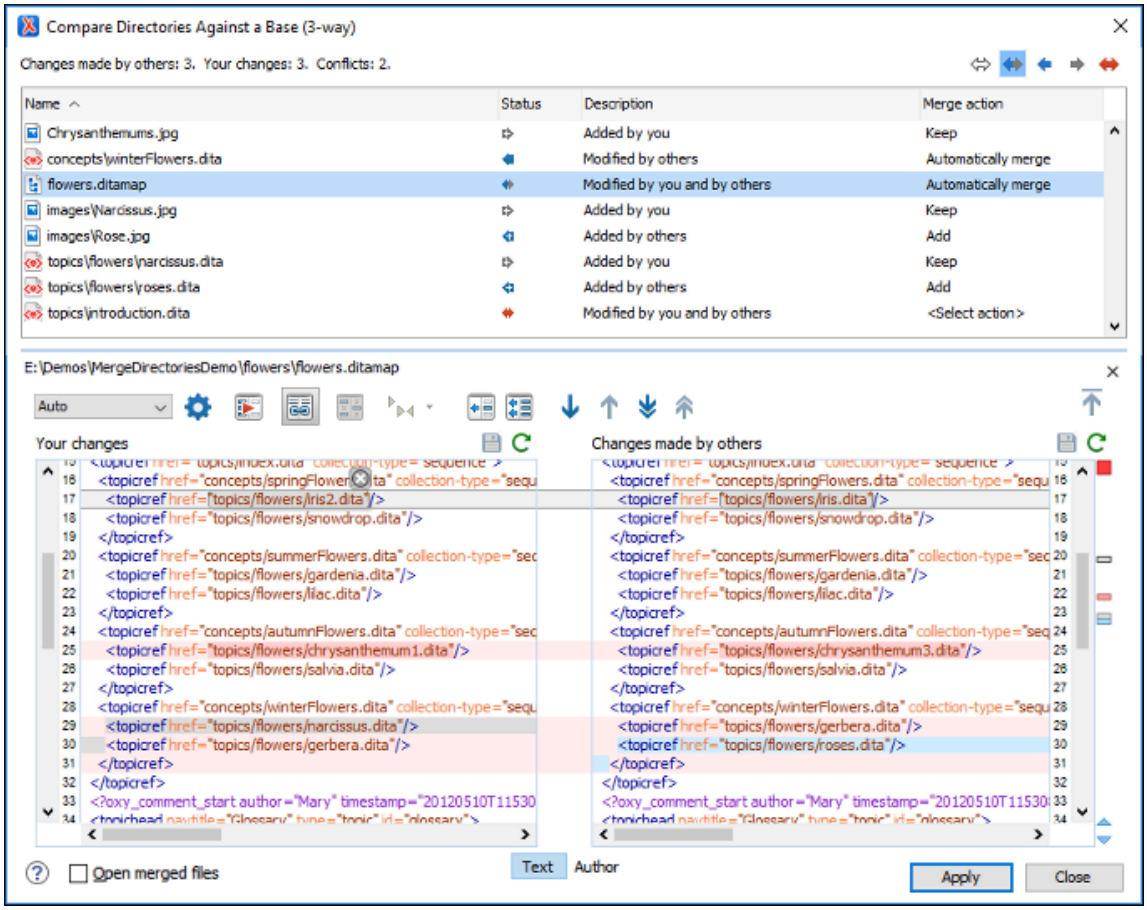
   Figure 521. Compare Directories Against a Base File Set Chooser

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 522. 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 Developer 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 Developer 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.
- **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 215)* 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.



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. If content is selected in the left panel, only that selection is affected, copying only the corresponding changes from the right to the selection on the left. The action name is also automatically updated to **Copy Changes in Selection from Right to Left** in this case.

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:

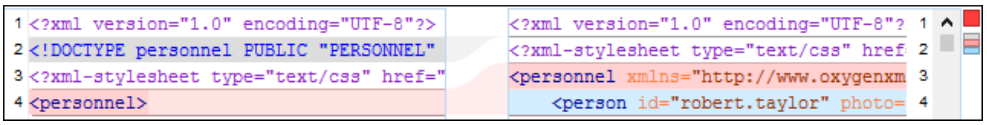
Reload

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 523. 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 217)*, 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.

Related information

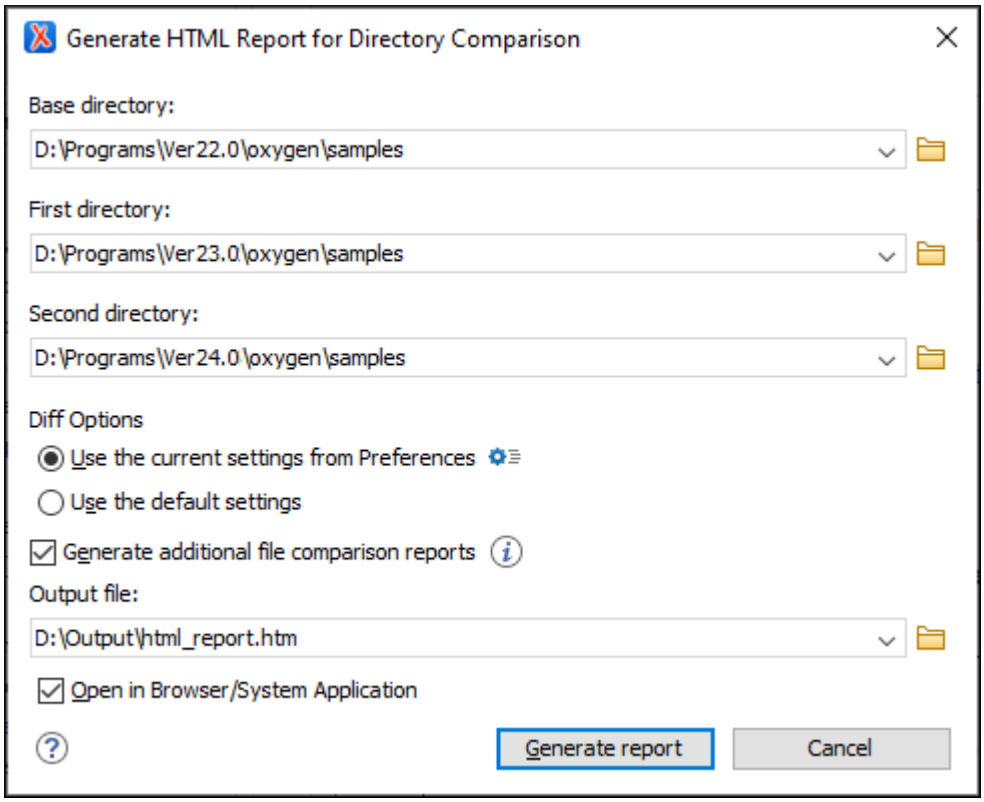
Compare Directories Tool *(on page 421)*
Compare Files Tool *(on page 402)*

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 >**

Comparison Tools menu. It opens a dialog box where you can specify the directories to compare as well as some other options.

Figure 524. Generate HTML Report for Directory Comparison Dialog Box



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 218)* and the include/exclude filter options in the **Compare Directories** tool *(on page 424)* 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 2056)*. 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.

Tip:
An XPath expression specified in the **Ignore nodes by XPath** text field within the **Files Comparison** preferences page *(on page 215)* is now taken into account if you enable the **Generate additional file comparison reports** option.

Output file

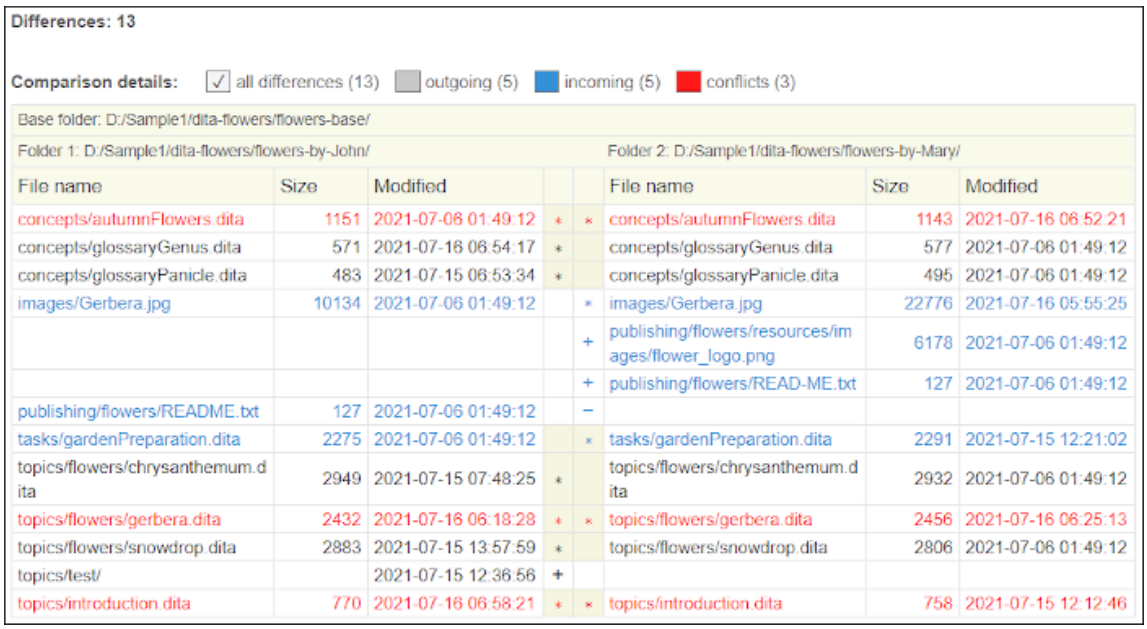
Specifies the path for an output file to save the comparison results file.

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 525. HTML Report for Directory Comparison



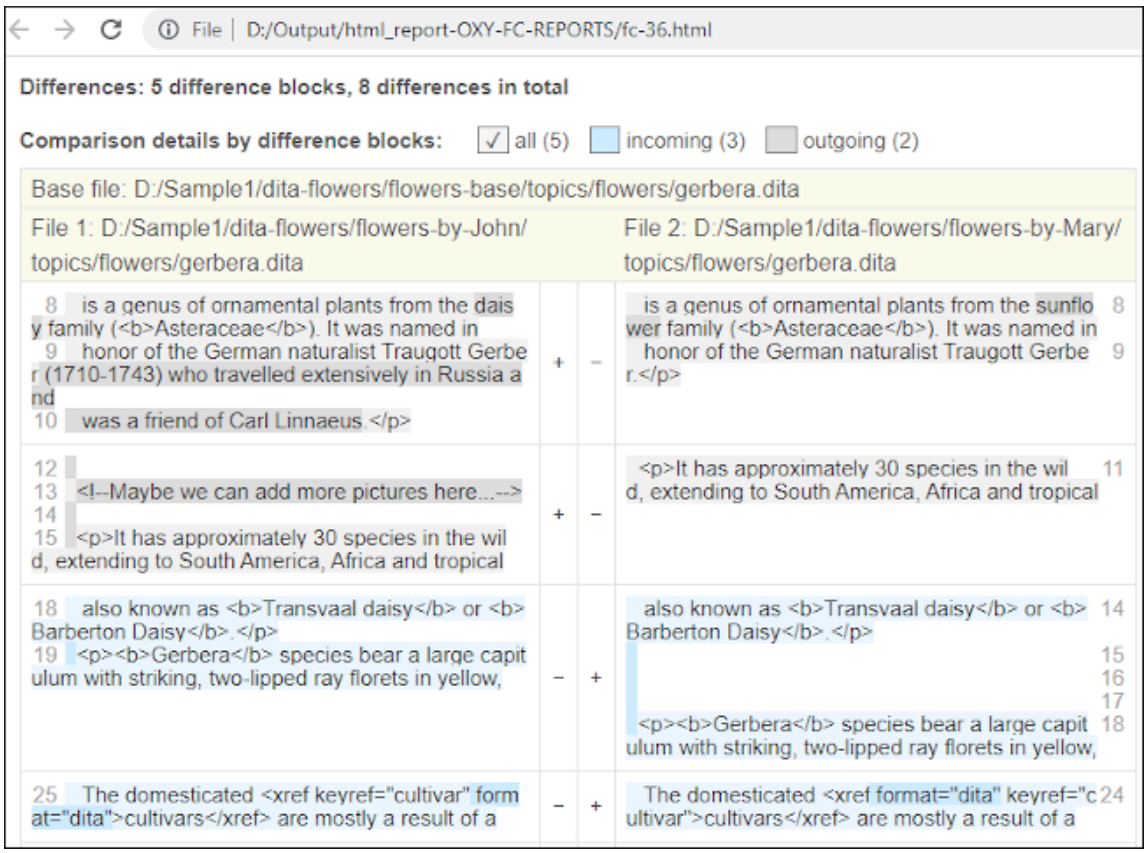
Differences: 13

Comparison details: ☑ all differences (13) outgoing (5) incoming (5) conflicts (3)

Base folder: D:/Sample1/dita-flowers/flowers-base/

Folder 1: D:/Sample1/dita-flowers/flowers-by-John/					Folder 2: D:/Sample1/dita-flowers/flowers-by-Mary/		
File name	Size	Modified			File name	Size	Modified
concepts/autumnFlowers.dita	1151	2021-07-06 01:49:12	*	*	concepts/autumnFlowers.dita	1143	2021-07-16 06:52:21
concepts/glossaryGenus.dita	571	2021-07-16 06:54:17	*		concepts/glossaryGenus.dita	577	2021-07-06 01:49:12
concepts/glossaryPanicle.dita	483	2021-07-15 06:53:34	*		concepts/glossaryPanicle.dita	495	2021-07-06 01:49:12
images/Gerbera.jpg	10134	2021-07-06 01:49:12		*	images/Gerbera.jpg	22776	2021-07-16 05:55:25
				+	publishing/flowers/resources/images/flower_logo.png	6178	2021-07-06 01:49:12
				+	publishing/flowers/READ-ME.txt	127	2021-07-06 01:49:12
publishing/flowers/README.txt	127	2021-07-06 01:49:12		−			
tasks/gardenPreparation.dita	2275	2021-07-06 01:49:12		*	tasks/gardenPreparation.dita	2291	2021-07-15 12:21:02
topics/flowers/chrysanthemum.dita	2949	2021-07-15 07:48:25	*		topics/flowers/chrysanthemum.dita	2932	2021-07-06 01:49:12
topics/flowers/gerbera.dita	2432	2021-07-16 06:18:28	*	*	topics/flowers/gerbera.dita	2456	2021-07-16 06:25:13
topics/flowers/snowdrop.dita	2883	2021-07-15 13:57:59	*		topics/flowers/snowdrop.dita	2806	2021-07-06 01:49:12
topics/test/		2021-07-15 12:36:56	+				
topics/introduction.dita	770	2021-07-16 06:58:21	*	*	topics/introduction.dita	758	2021-07-15 12:12:46

Figure 526. 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 421)*
Compare Directories Against a Base (3-Way) Tool *(on page 427)*
Compare Files Tool *(on page 402)*

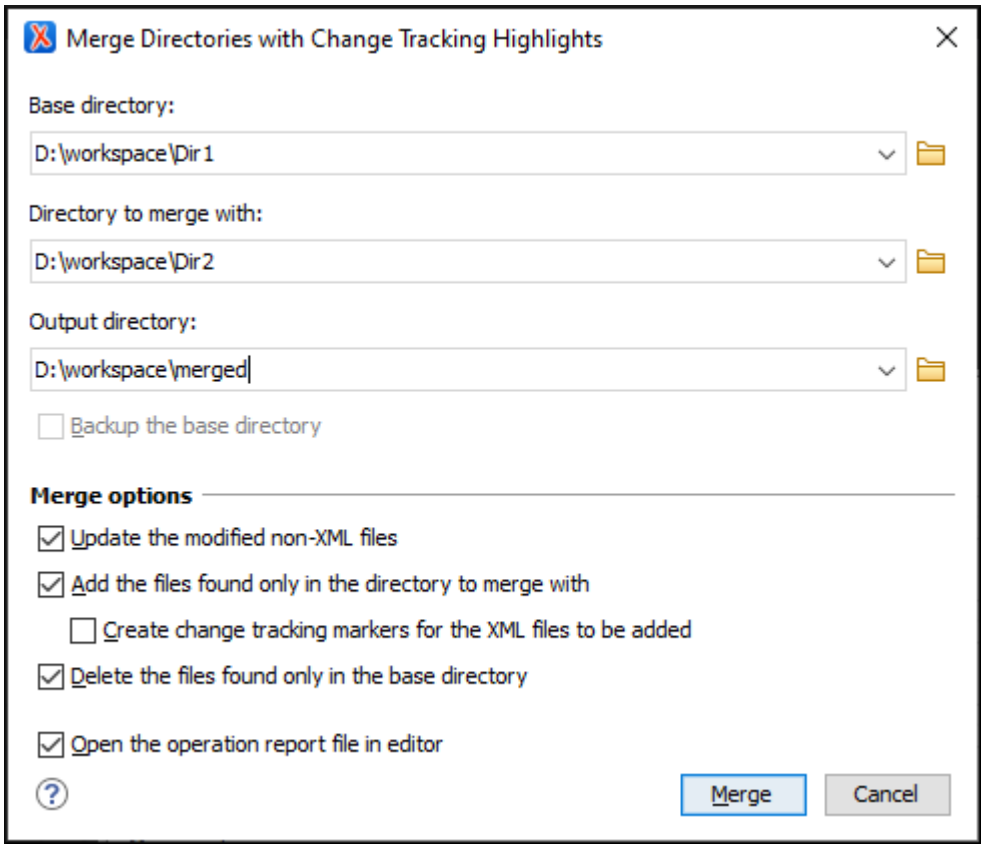
Merge Directories with Change Tracking Highlights

The **Merge Directories with Change Tracking Highlights** tool can be used to merge two directories (based on a 2-way mode comparison) and all pairs of modified XML files involved in the process are merged by saving the comparison results as documents with highlighted tracked changes that can be later reviewed and accepted or rejected.

All other detected situations are treated based on several options that are applicable to both XML and non-XML files, or another option that is applicable only to non-XML files.

To invoke the tool, select the **Merge Directories with Change Tracking Highlights** action that is found in the **Tools > Comparison Tools** menu. It opens a dialog box where you can specify the directories to merge and the output directory to save the results, as well as some other options.

Figure 527. Merge Directories with Change Tracking Highlights Dialog Box



The **Merge Directories with Change Tracking Highlights** dialog box contains the following options:

Base directory

Specifies the path of the base directory.

Directory to merge with

Specifies the path of the directory to merge with.

Output directory

Specifies the path of the directory where the merge operation results are saved to. You can leave this field blank and the merge results will be saved in the base directory by overwriting it, or you can choose a separate output directory.

> **Note:**
> You cannot choose the same directory specified as the directory to merge with as the output directory.

Backup the base directory

If this option is selected, a backup copy is saved on your hard disk when the operation completes. This option is automatically activated and selected in the following cases:

- The output directory is not specified (the field is left blank).
- The specified output directory is the base directory itself.
- In both of the above situations, a backup of the base directory will be performed automatically and the backup operation must succeed to proceed with the merge. Otherwise, the merge process being aborted.



Note:
The backup copy will have the same parent directory as the base directory and its name will be the name of the base directory suffixed by ".OXY.BAK".

Merge options

The merge process has a preliminary phase where the entire structure and the content of the base directory is copied to the output directory. The following merge options are available:

Update the modified non-XML files

If this option is selected, all files in the output directory that are copies of non-XML files in the base directory are replaced by their corresponding files in the directory to merge with. Otherwise, they remain unchanged.

Add the files found only in the directory to merge with

If this option is selected, all files that are only present in the directory to merge with are also added to the output directory.

Create change tracking markers for the XML files to be added

This option can only be used in conjunction with the **Add the files found only in the directory to merge with** option. Select this option if you want to create change tracking markers for the XML files that are only present in the directory to merge with and will be added to the output directory. Although these files have no counterparts in the base directory, change tracking markers of type "entire content added/inserted" will be created.

The option is not necessarily intended for the merge process itself, but it is useful if you want to apply various *Oxygen* transformation scenarios to the resulting output directory. For example, if you merge 2 versions of a DITA project and then want a PDF to highlight the changes between those versions, you can apply a transformation on the resulting `ditamap` file. This option results in presenting the new DITA files as "added content" in the resulting PDF.

Delete the files found only in the base directory

If this option is selected, all files that are only present in the base directory and initially copied to the output directory are deleted. Otherwise, they are preserved.

Open the operation report file in editor

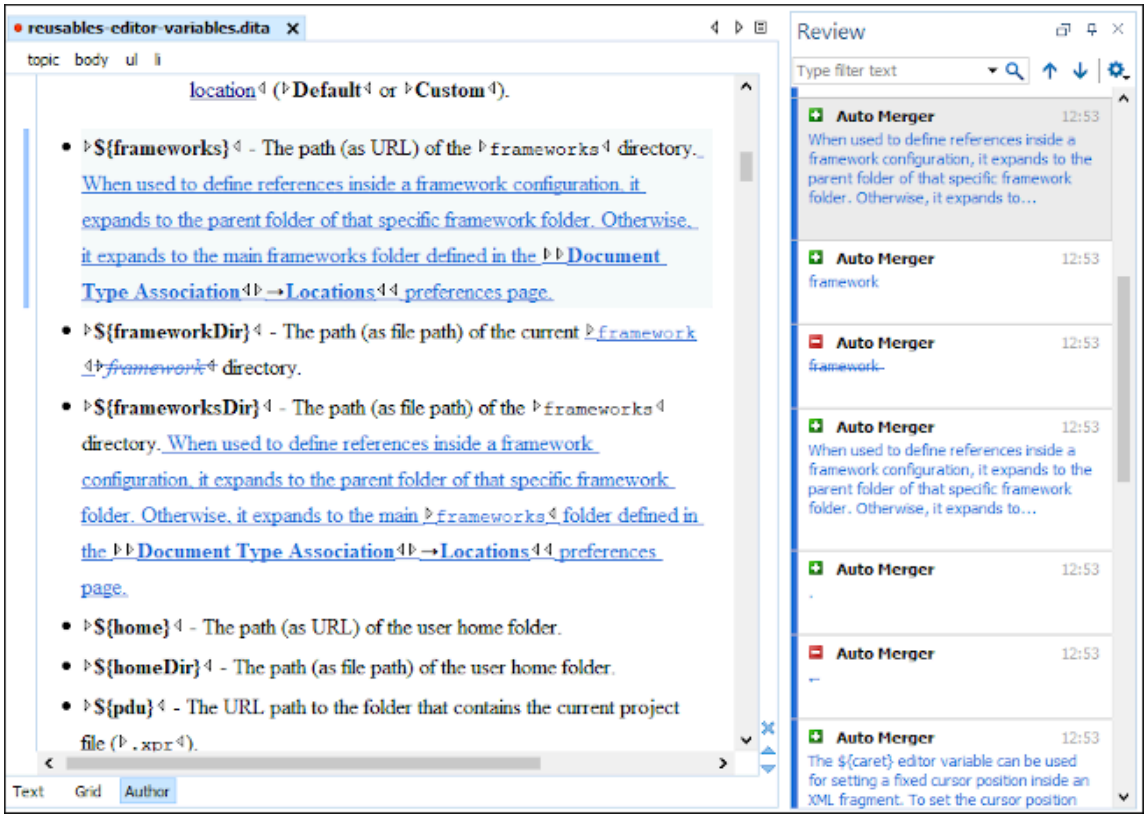
Opens the merge operation report (XML file) in the editor.

Once the merge operation is complete, loading the report file in Oxygen XML Developer provides additional functionality. Aside from the fact that the report provides an overview of the merge process, it also provides links to all the files in the resulting output directory. This is helpful for generating XML documents with tracked changes. For example, you can use the respective links to load these XML files in **Author** mode to review the tracked changes and accept or reject them. This phase of reviewing and manually merging the XML files is typically the final phase of the entire merge procedure.

Figure 528. Example of an Operation Overview Report File



Figure 529. Example of a Merged File Opened in Author Mode



Resources

For more information about the merge with change tracking support, see the following resources:

- Video: **Mastering Document Comparisons: A Guide to Generating Tracked Changes Between Two Versions**
- Webinar: **Discover the Power of Diff with Change Tracking**

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 Developer 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 2152)* - Allows you to define and manage Apache Subversion™ repository locations.
- **Working Copy** view *(on page 2157)* - Allows you to manage with ease the content of the working copy.
- **History** view *(on page 2172)* - 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 2178)* - Allows you to edit various types of text files, with full syntax-highlight.
- **Annotations** view *(on page 2179)* - Displays a list with information regarding the structure of a document (author and revision for each line of text).
- **Compare** view *(on page 2181)* - Displays the differences between two revisions of a text file from the working copy.
- **Image Preview** panel *(on page 2185)* - Allows you to preview standard image files supported by Syncro SVN Client: JPG, GIF and PNG.
- **Compare Images** view *(on page 2185)* - Displays two images side by side.
- **Properties** view *(on page 2185)* - Displays the SVN properties of a resource under version control.
- **Console** view *(on page 2187)* - Displays information about the currently running operation, similar with the output of the Subversion command-line client.
- **Dynamic Help** view *(on page 2187)* - 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 2152)* tree or the **Working Copy** view *(on page 2157)* tree, depending on the view that was last used. Note that for the **Working Copy** view *(on page 2157)*, 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 2152)* tree or the **Working Copy** view *(on page 2157)* tree, depending on the view that was last used. Note that for the **Working Copy** view *(on page 2157)*, 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 2193)*, 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 2195)* 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 2152)*, **Working Copy** view *(on page 2157)*, **History** view *(on page 2172)*, or **Directory Change Set** view *(on page 2177)*, 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 2157)*.

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 2152)*, **Working Copy** view *(on page 2157)*, **History** view *(on page 2172)*, or **Directory Change Set** view *(on page 2177)*, 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 + Option + 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 2092)*.
- **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 2093)*.
- **Unlock (Ctrl + Alt + K (Command + Option + 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 2185)* and displays the SVN properties for a selected resource from **Repositories** view *(on page 2152)* or **Working Copy** view *(on page 2157)*, 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 2109)*.

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)**

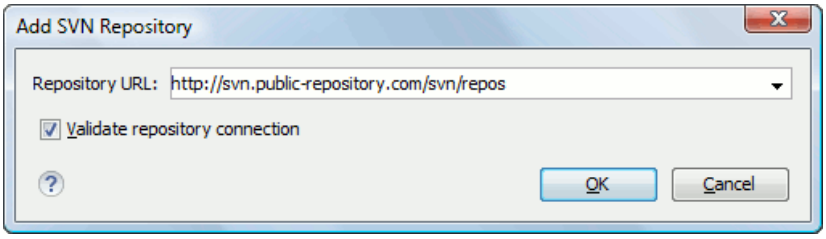
Go to the previous match using the same find options of the last find operation. This action runs in the editor panel and in any non-editable text area (for example, the **Console** view).

Repository Menu

 **New Repository Location (Ctrl + Alt + N (Command + Option + N on macOS))**

Displays the **Add SVN Repository** dialog box. This dialog box allows you to define a new repository location.

Figure 530. 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 + Option + 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 + Option + 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 + Option + 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 (Ctrl + Alt + O (Command + Option + 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 2082)*.

Export

Opens the **Export** dialog box *(on page 2147)* 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 2146)*.

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 files 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 212)* 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 2151)* 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 2100)* 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 2098)*.

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 2101)*.

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 2101)* 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 2100)* 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 2102)*.

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 + Option + 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 2088)* section.

 **Add to version control (Ctrl + Alt + V (Command + Option + 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 2086)* 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 + Option + 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 + Option + 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.

> **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. If content is selected in the left panel, only that selection is affected, copying only the corresponding changes from the right to the selection on the left. The action name

is also automatically updated to **Copy Changes in Selection from Right to Left** in this case.

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.

ormat 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 2179)*, 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 2188)*. 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 2080)* 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 2111)*.

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 2113)*.

Switch (Ctrl + Alt + W (Command + Option + 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 2130)* 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 2132)* section.

Create patch (Ctrl + Alt + P (Command + Option + 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 2135)*.

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 2164)* 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 223)*, 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 Developer 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.

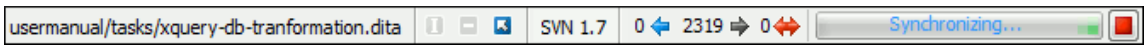
 **Enable/Disable flexible layout**

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 531. 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 2163)* 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 Developer 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 2152)*.

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 2193)* you want to connect to. You can also use peg revisions at the end of the URLs *(on page 2195)* (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 Developer tries to contact the server to get the content of the repository for displaying it in the **Repositories** view *(on page 2152)*. 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 2193)* you want to connect to. You can also use peg revisions at the end of the URLs *(on page 2195)* (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 2152)*. 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 2152)*.

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 Developer for the change to take effect.



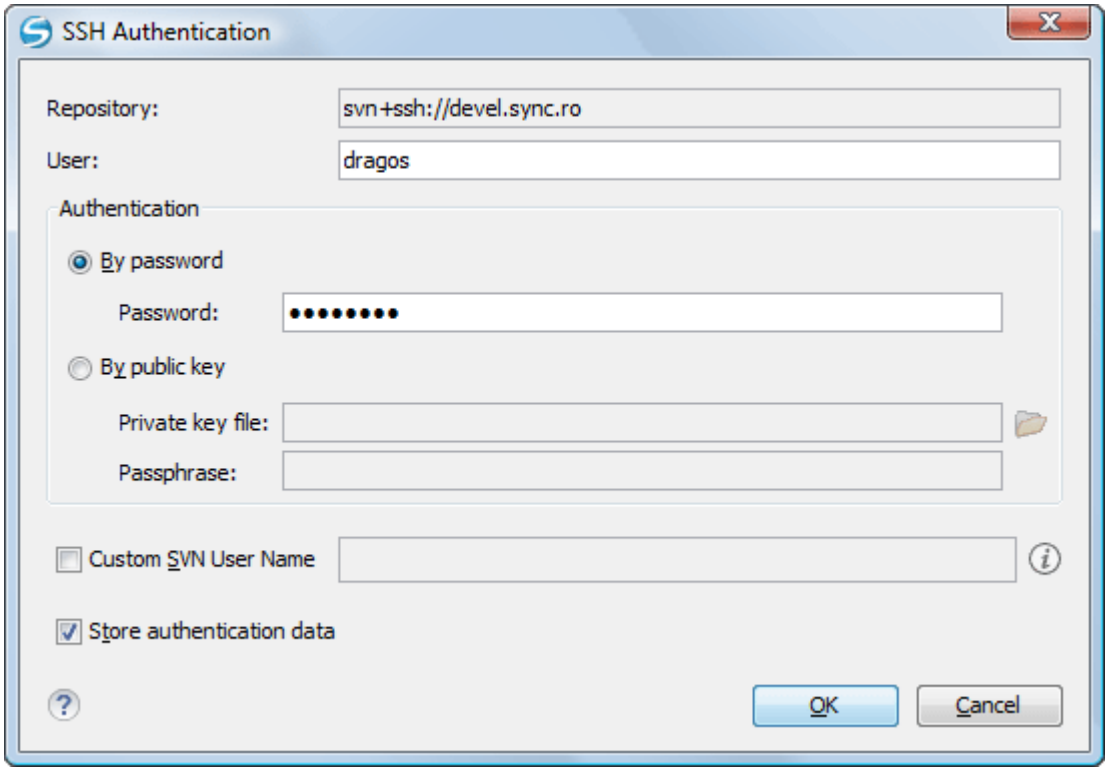
Tip:
The *FILE* protocol is recommended if the SVN repository and Oxygen XML Developer are located on the same computer as it ensures faster access to the SVN repository compared with other protocols.

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.

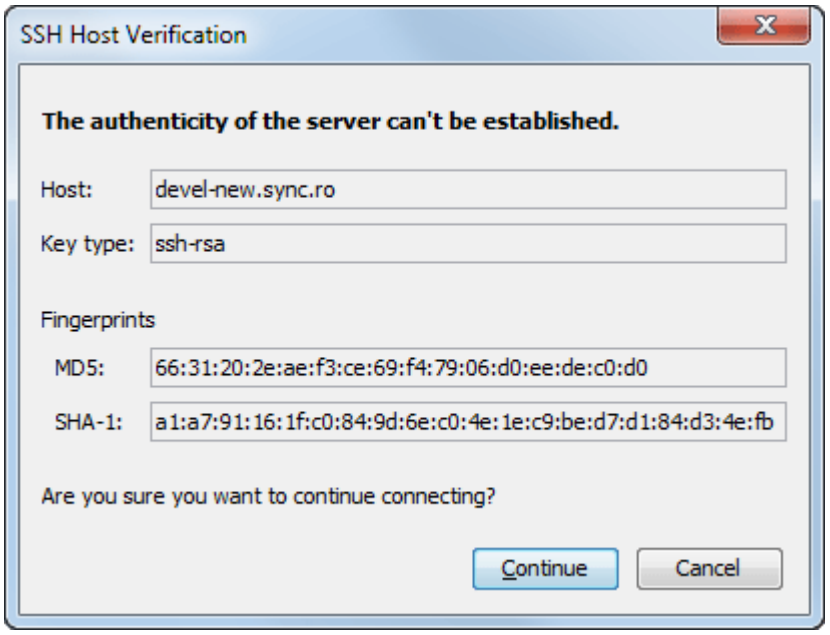
Figure 532. SSH Authentication Dialog Box



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.

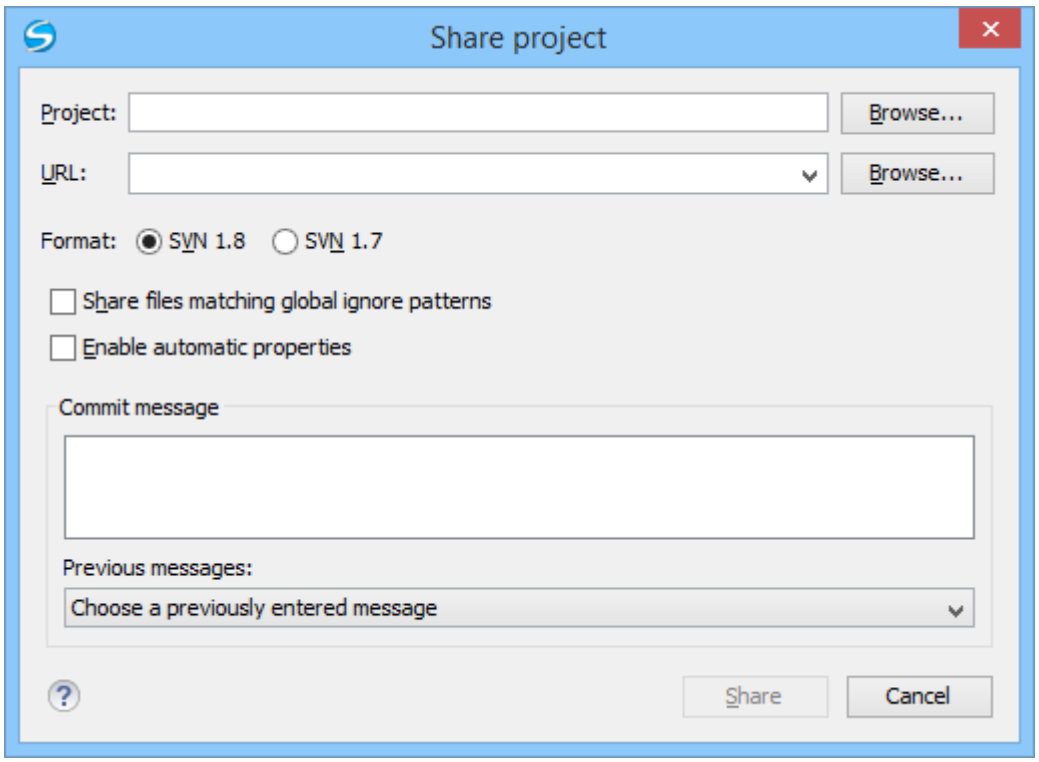
Figure 533. SSH server name and key fingerprint



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 534. 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 2192)* 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 2193)* (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 2192)*.
- The **File name ignore patterns** option *(on page 213)* in the **SVN > Working Copy** preferences page *(on page 212)*.

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 2192)*.

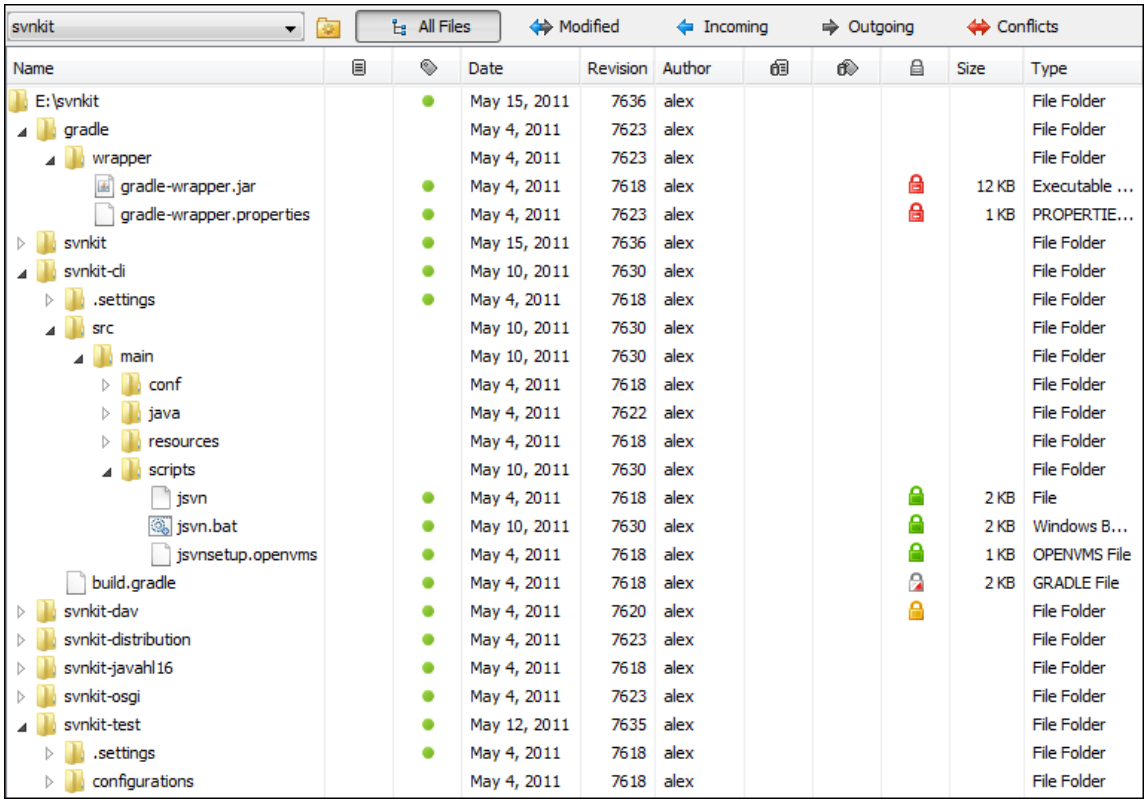


Note:
This option is available only when there are defined properties to be applied automatically for newly added items under version control. You can define these properties in the SVN `config` file (in the `auto-props` section). Based on the value of the `enable-auto-props` runtime configuration directive, 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.

Figure 535. Working Copy View



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.

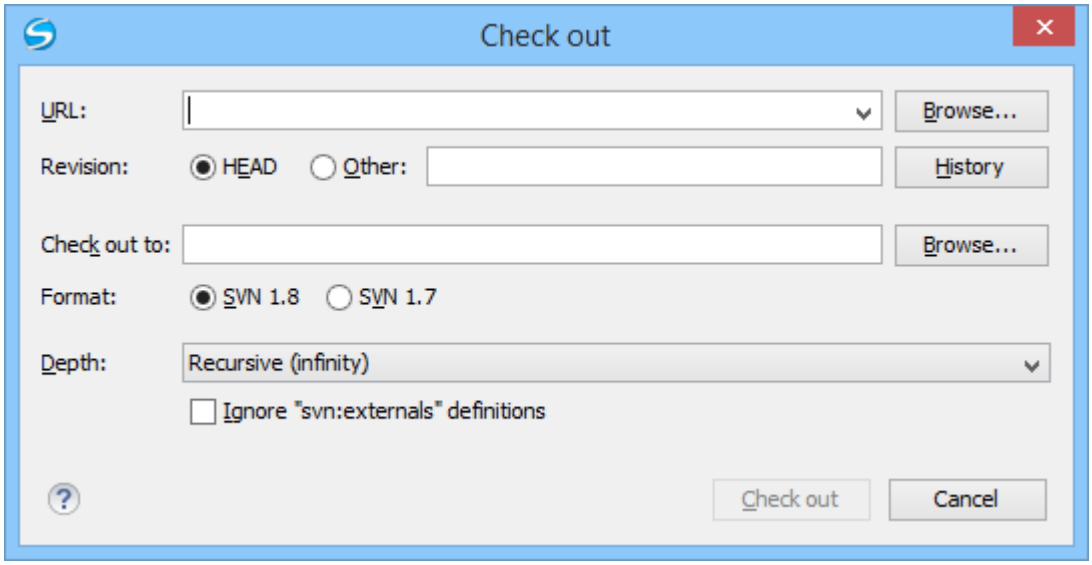
A typical Subversion repository often holds the files (or source code) for several projects. Usually each project is a subdirectory in the repository's file system tree. In this arrangement, a user's working copy usually corresponds to a particular sub-tree of the repository.

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.

Figure 536. Check Out Dialog Box



The following options can be configured in the **Check out** dialog box:

URL

The location of the repository directory *(on page 2193)* 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 2195)* 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 2084)*.

Check out to

Specify the location where you want to *check out (on page 2192)* 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 2159)* 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 2151)* 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

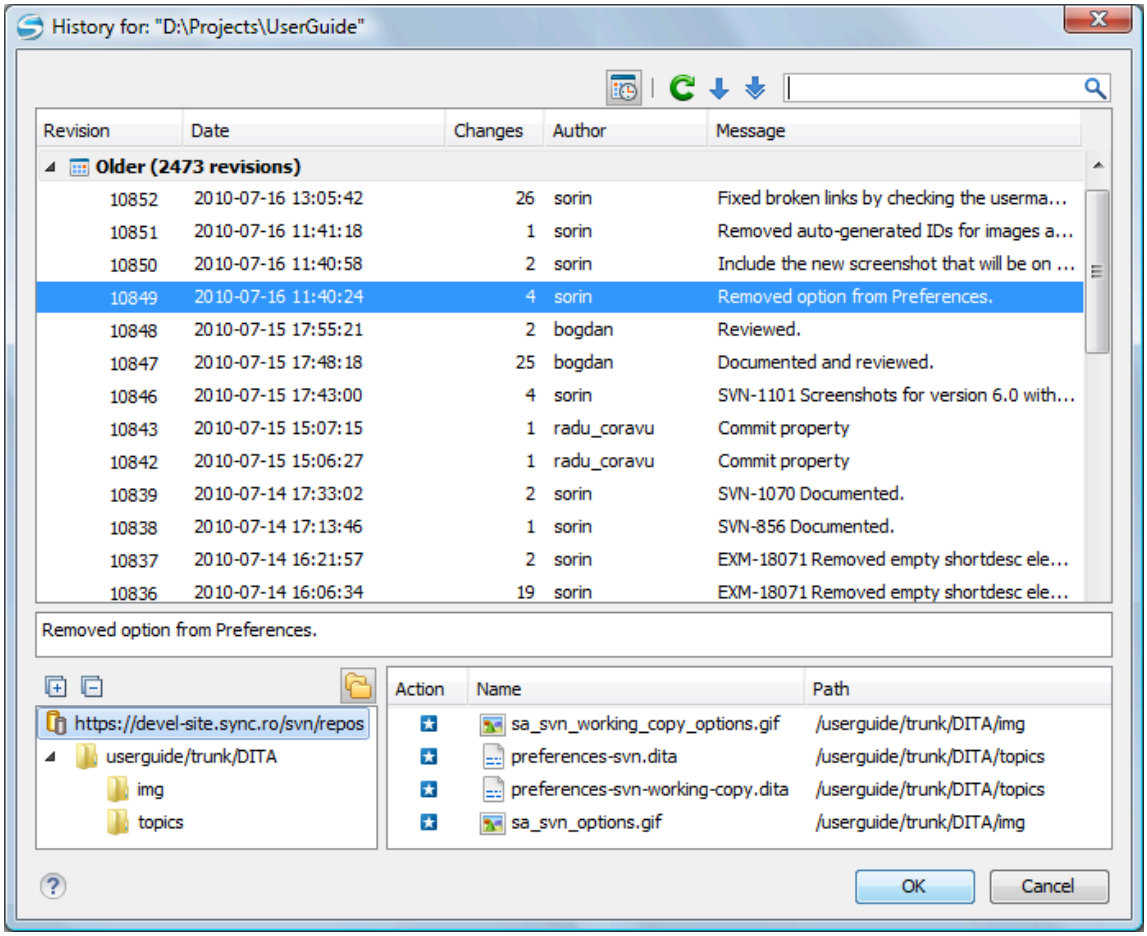
When selected, external items are ignored in the *check out* operation. This option is only available if you choose the **Recursive (infinity)** depth.

After a *check out*, the new working copy is added to the list in the **Working Copy** view *(on page 2157)* 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 2082)* or the **Branch / Tag** dialog box *(on page 2111)*. It presents information about revision, commit date, author, and commit comment.

Figure 537. History Dialog Box



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 2146)*. The following procedure assumes that you have an existing valid working copy on your file system.

1. Click the **Working Copies Manager** toolbar button (on macOS) in the **Working Copy** view *(on page 2157)*.

   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 2157)*.

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 2157)* 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 2157)*. 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 2157)*.

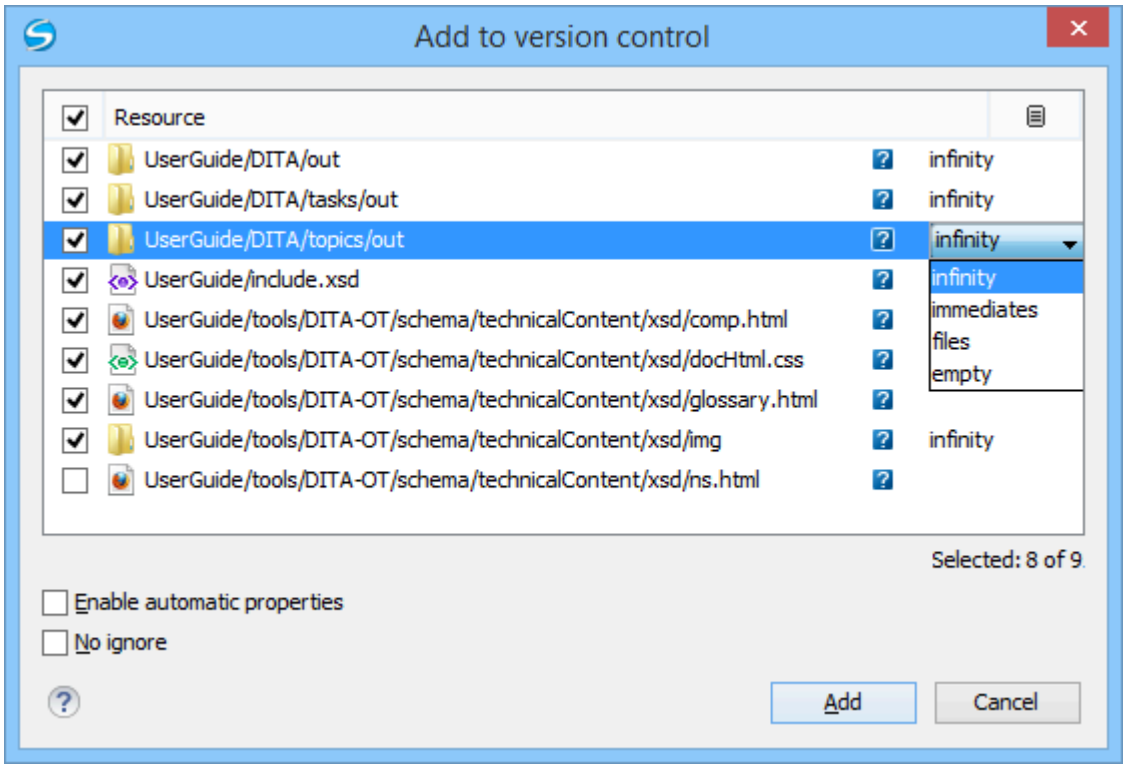
You can easily spot resources not under version control by the *unversioned* (?) icon displayed in the **Local file status** column. Resources scheduled for addition are displayed with this *added* (+) icon in the **Working Copy** view and are added in the repository after you commit them.

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 2104)*, 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.

Figure 538. 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 2163)*:

- *infinity* - When the working copy items are presented as a tree.
- *files* - When the working copy items are presented compressed.
- *empty* - When the working copy items are presented flat.

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 Developer options (open the **Preferences** dialog box **(Options > Preferences)** *(on page 79)* 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 2104)*, 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 2157)*.

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 2157)*. 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* ( ) 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 ( ) 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 2103)* 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 loosing any important remote modifications.

Finally, you can change your mind and revert *(on page 2100)* 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 2157)* 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 Developer 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 2159)* (*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 2159)* 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 2163)* 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 2157)* 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 2157)* 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 (check out). 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.

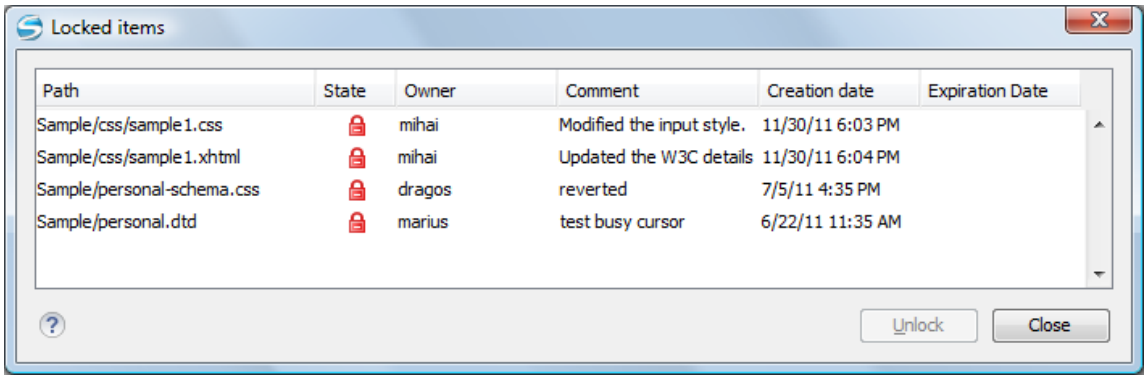
An SVN lock is a piece of metadata that grants exclusive access to a user. This user is called the lock owner. A lock is uniquely identified by a lock token (a string of characters). If someone else attempts to commit the file (or delete a parent of the file), the repository demands two pieces of information:

- 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 2157)* by selecting one or more resources, then right-clicking them and choosing the **Scan for Locks** action from the contextual menu.

Figure 539. Locked Items Dialog Box



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 2161)*
Repository Locks *(on page 2152)*

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 2157)*. 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 2157)* 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 2157)*.

View Differences

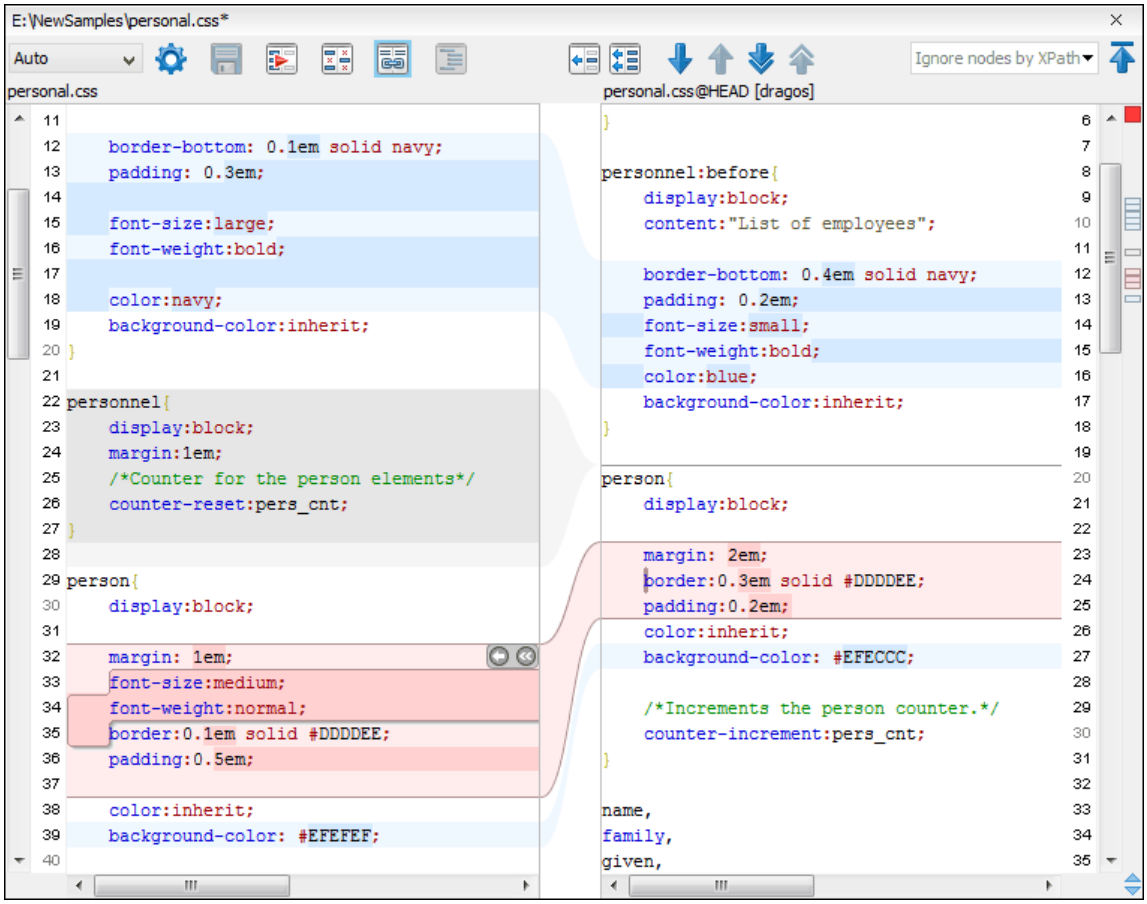
One of the most common requirements in project development is to see what changes have been made to the files from your Working Copy or to the files from the repository. You can examine these changes after a synchronize operation with the repository, by using the **Open in compare editor** action from the contextual menu.

The text files are compared using a built-in **Compare** view *(on page 2181)* 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 213)*). 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 211)*.

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 2157)*.

Figure 540. Compare View



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 2182)* 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 210)* 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 210)*. 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 210)*) is opened with the **Edit Conflict** action, which is available on the contextual menus of the **Working Copy** view *(on page 2157)* 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 214)* 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. If content is selected in the left panel, only that selection is affected, copying only the corresponding changes from the right to the selection on the left. The action name is also automatically updated to **Copy Changes in Selection from Right to Left** in this case.

 **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.

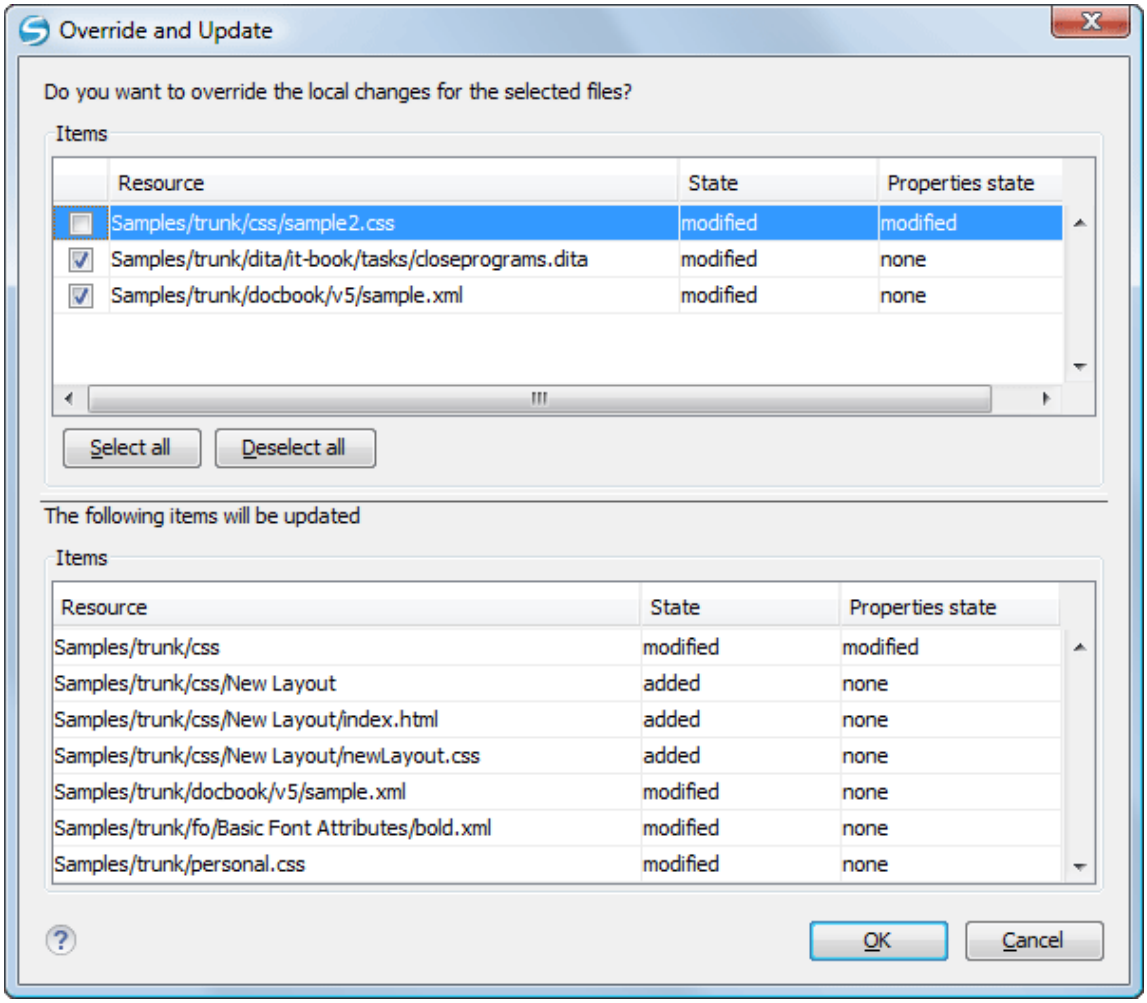


Note:
By default, a directory will be recursively reverted (including any other modified item it contains). However, if the directory has only property changes, you need to explicitly choose if the operation will include any modified items found inside it.

If you want some of your outgoing changes to be overridden you must first open the file in **Compare** view *(on page 2181)* 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 2182)*. 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.

Figure 541. Override and Update Dialog Box



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 2118)*.

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 2185)*. 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 2181)*, Editor *(on page 2178)* 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

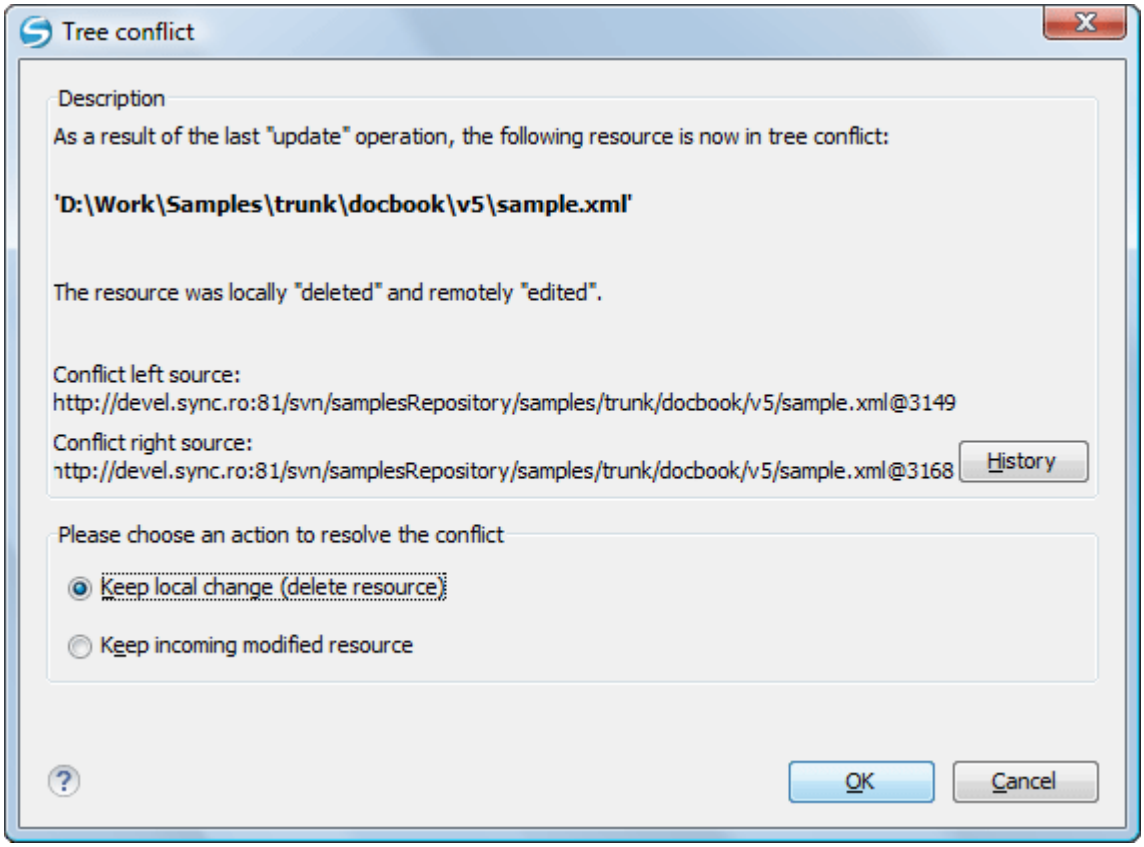
A *tree conflict* is a conflict at the directory tree structure level and occurs when the user runs an update action on a resource that has the following conditions:

- 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 542. 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 2157)*. 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 Developer 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 2172)* 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.

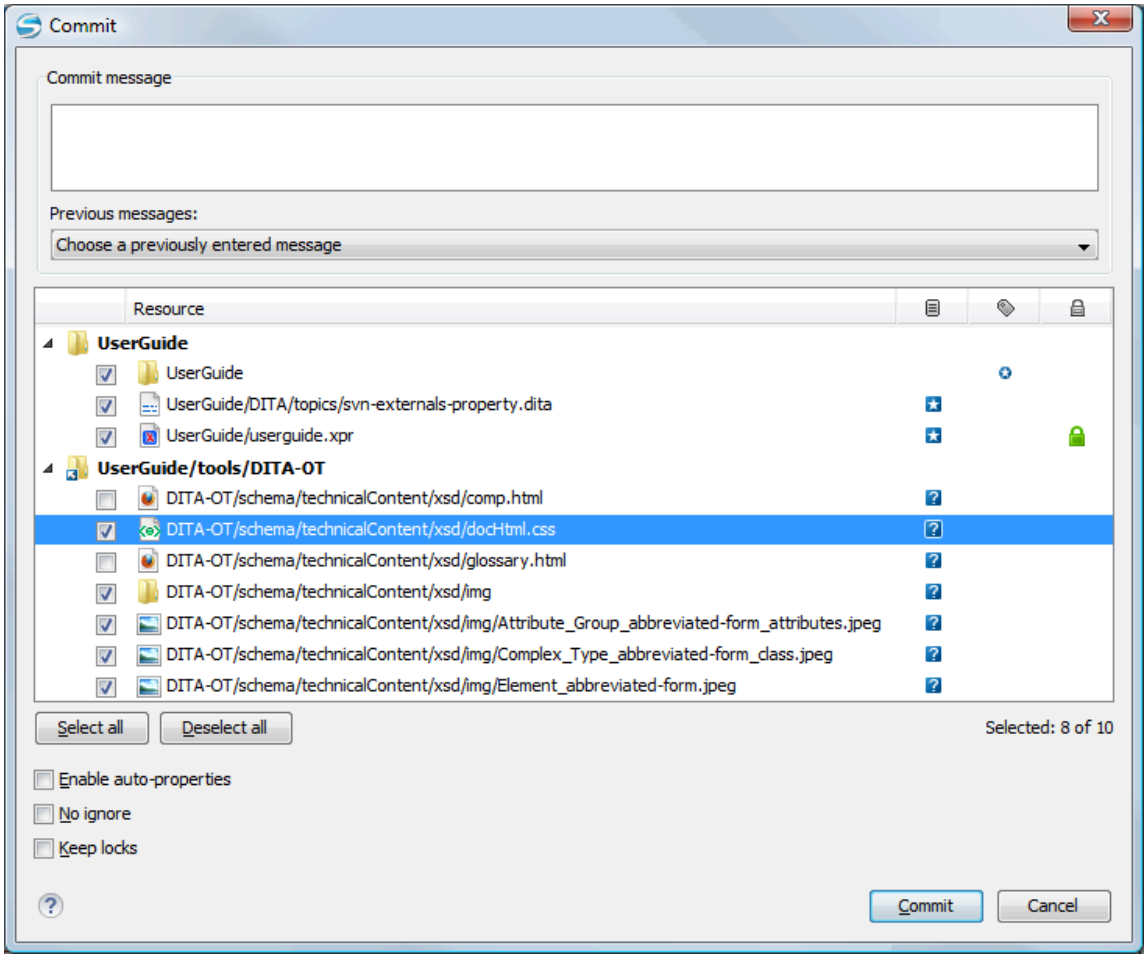
When the update fails with a message saying that there is already a local file with the same name Subversion tried to check out a newly versioned file, and found that an unversioned file with the same name already exists in your working folder. Subversion will never overwrite an unversioned file unless 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 543. 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* ( ) or *added* ( ) 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.



Note:
External files are always from the same repository as the parent directory that defines them, so they are always committed together with the changes from their parent directory.

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 2110)* 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 2104)* 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 2172)* 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 2157)* 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 2152)*, Working Copy view *(on page 2157)*, Revision Graph *(on page 2188)*, or Directory Change Set view *(on page 2177)*. 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 2172)*

Management of SVN Properties

In the **Properties** view *(on page 2185)* 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 2157)* and run the **Show properties** action from the contextual menu. The **Properties** view *(on page 2185)* 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 2187)* 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 *(on page 2185)* 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.

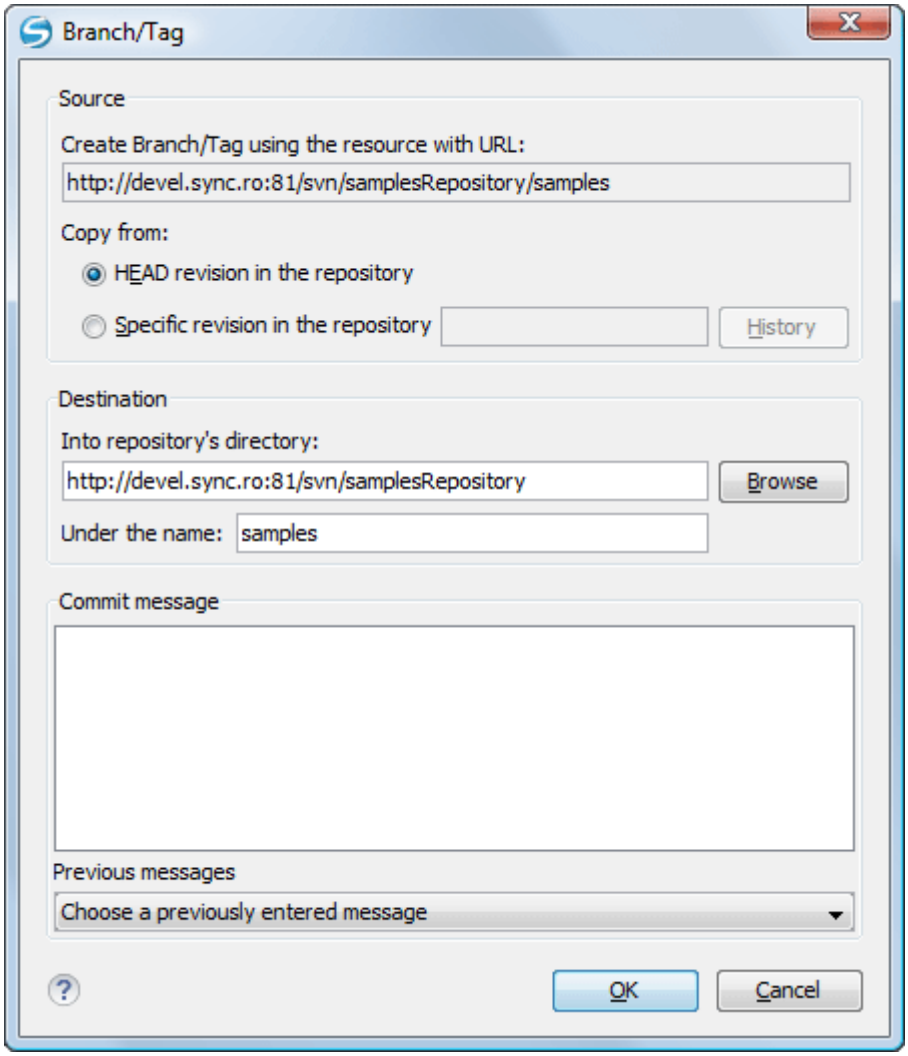
In Apache Subversion™, there is no difference between a *tag* and a *branch*. On the repository, both are ordinary directories that are created by copying. The trick is that they are cheap copies instead of physical copies. Cheap copies are similar to hard links in Unix, which means that they merely 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 *(on page 2157)* or **Repositories** view *(on page 2152)*, or from the contextual menu of the **Repositories** view.

Figure 544. Branch/Tag Dialog Box



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 2084)*. 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 2193)* 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.

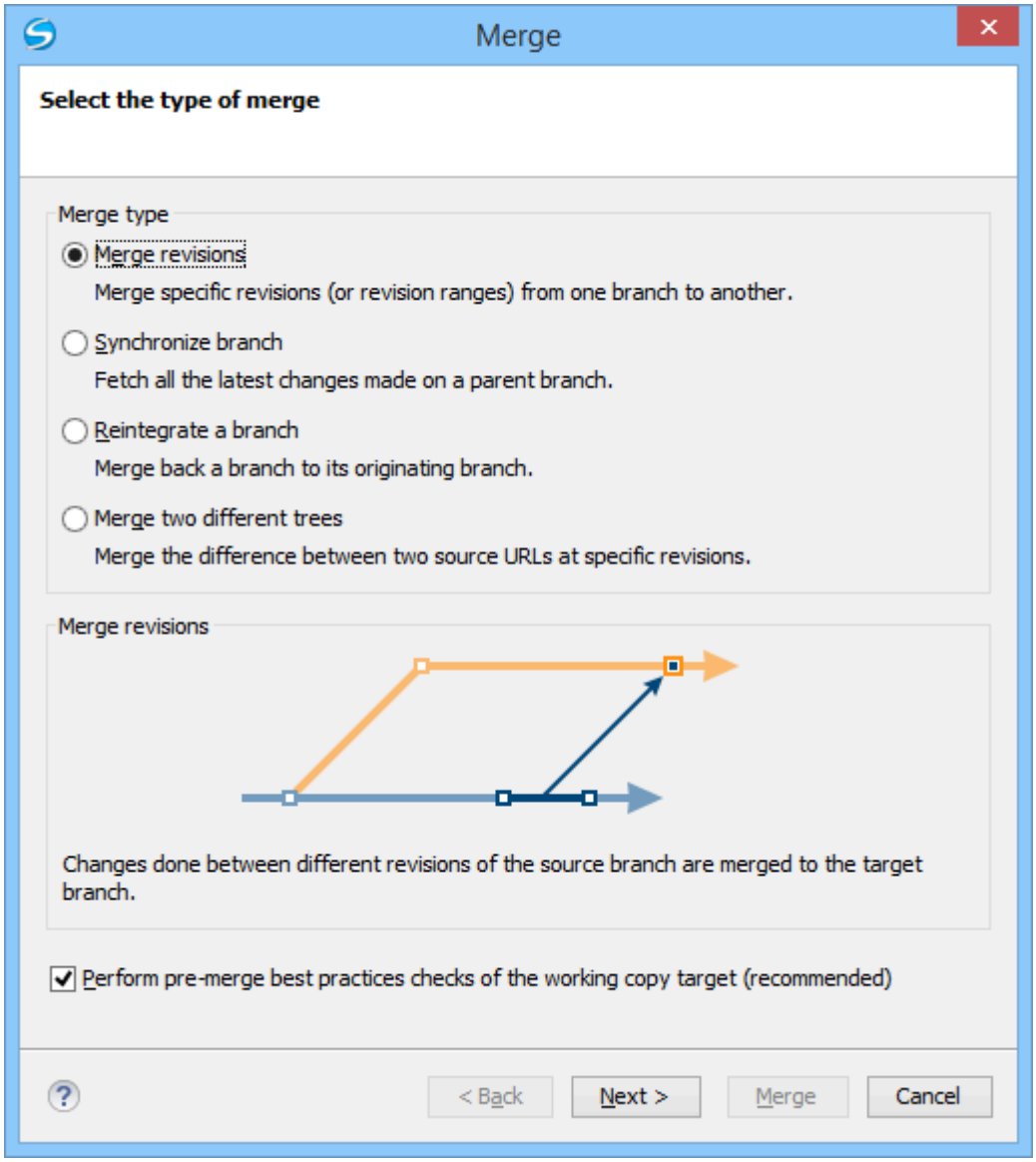
Merging

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.

Figure 545. Merge Wizard



The four types of merging are as follows:

- **Merge revisions** *(on page 2117)* - Port changes from one branch to another. Note that the *trunk* can also be considered a branch, in this context.
- **Synchronize branch** *(on page 2119)* - Fetch all the changes made on a parent branch (or the *trunk*) to a child branch.
- **Reintegrate a branch** *(on page 2121)* - Merge a branch back to its parent branch (can also be the *trunk*).
- **Merge two different trees** *(on page 2123)* - 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 2115)* - 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.

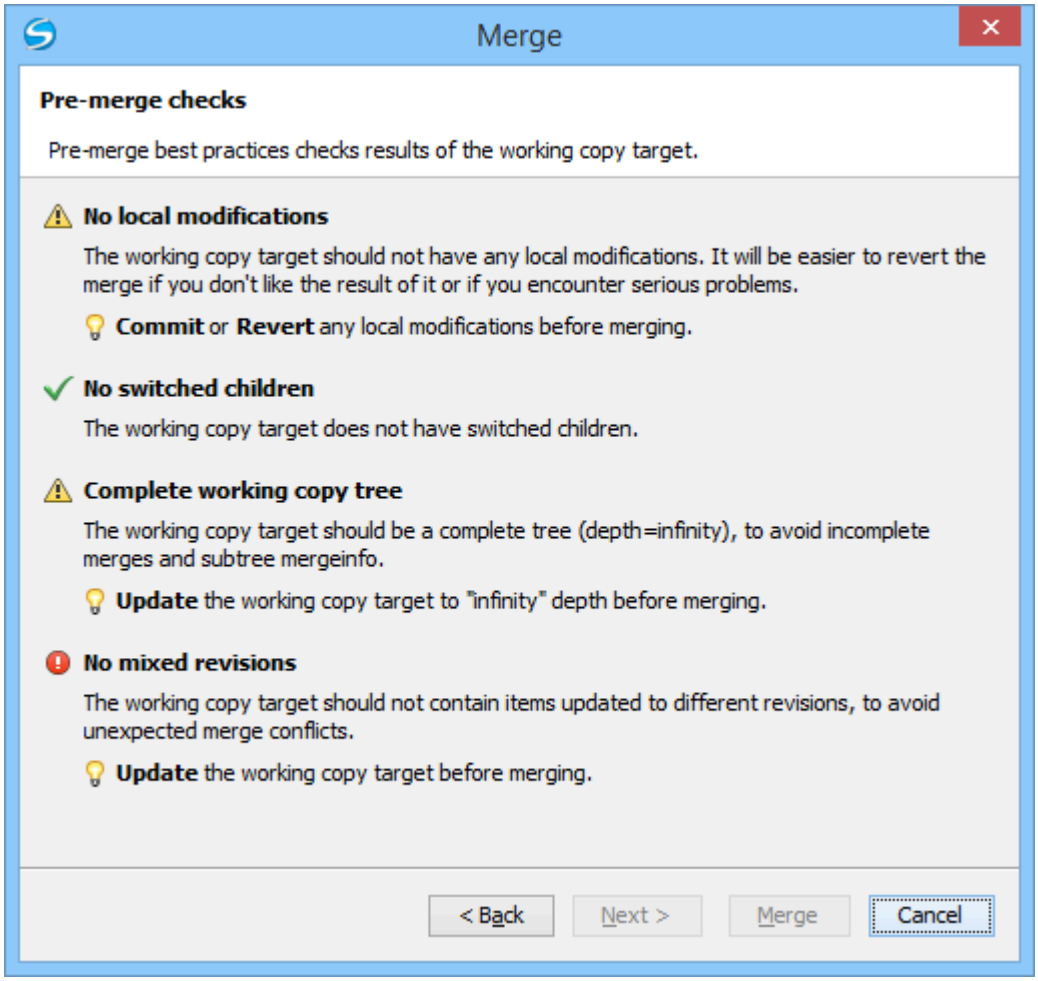
Remember:
It is a good idea to perform a merge into an unmodified working copy. If you have made changes to your working copy, commit them first. If the *merge* does not go as you expect, you may want to revert the changes and revert cannot recover your uncommitted modifications.

Important:
The above recommendation becomes mandatory when *reintegrating a branch (on page 2113)*.

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.

Figure 546. Pre-Merge Checks Wizard Page



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 2115)* 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 2193)* 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 2126)*) will be selected automatically (and you cannot change this). This is because the Subversion client cannot track changes between different repositories *(on page 2129)*.

> **Tip:**
> You can also specify a *peg revision (on page 2195)* 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 2084)* 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 2126)* for your merge, click the **Next** button.

The **Merge Options** wizard page *(on page 2126)* 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 2128)* 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 2115)* 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 2193)*. 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 2195)* 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 2126)* for your merge, click the **Next** button.
   The **Merge Options** wizard page *(on page 2126)* 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 2128)* 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 2115)* 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 2115)* 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 2193)*.
   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 2195)* 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 2126)* is displayed that allows you to configure options for the operation.

> **Note:**
> Since a *reintegrate merge* is so specialized, most of the merge options are not available, except for those in the **File Comparison** category.

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 2128)* 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 2115)* 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 2193)*.
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 2126)*) will be selected automatically (and you cannot change this). This is because the Subversion client cannot track changes between different repositories *(on page 2129)*.

    > **Tip:**
    > You can also specify a *peg revision (on page 2195)* 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 2193)*.

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 2195)* 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 2126)* for your merge, click the **Next** button. The **Merge Options** wizard page *(on page 2126)* 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.

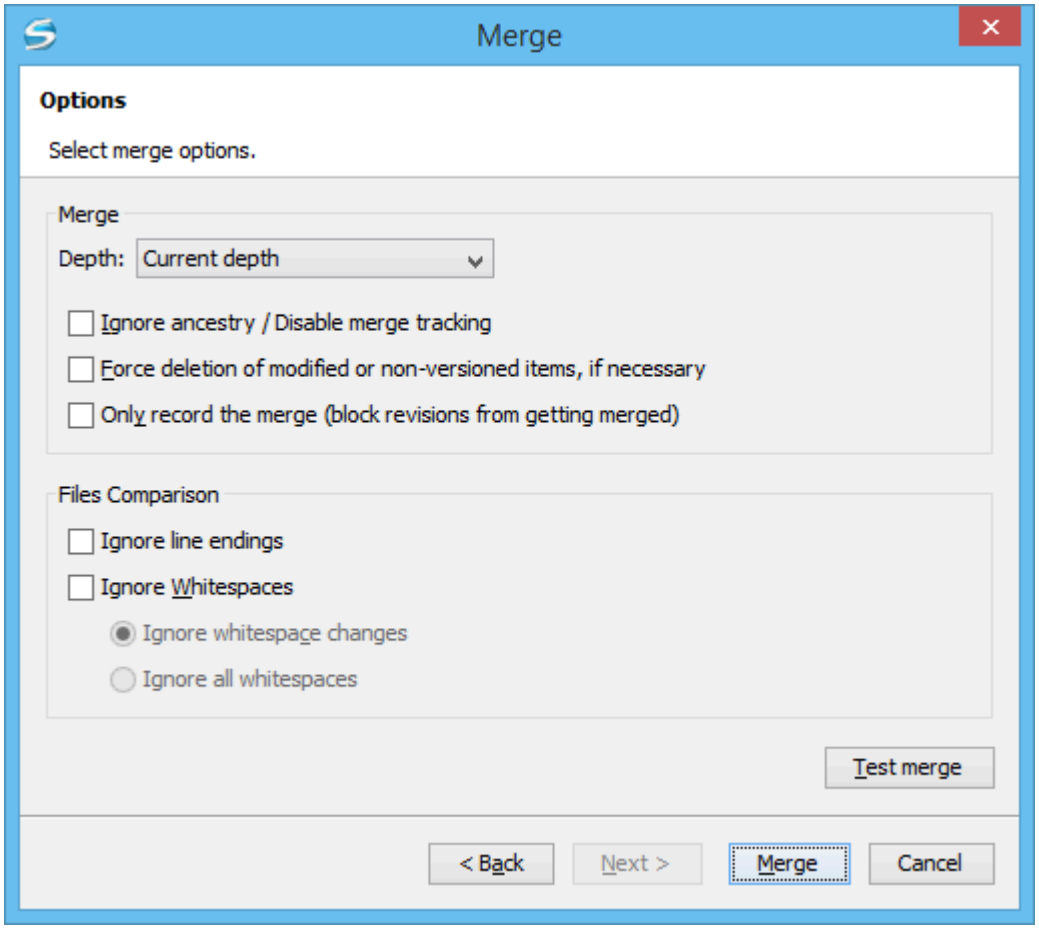
It is recommended to look at the results of the merge, in the working copy, to review the changes and see if it meets your expectations. Since merging can sometimes be complicated, you may need to resolve conflicts *(on page 2128)* after making major changes.

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 547. 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** - Obeys 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 2151)* 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 2129)*.

- **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 2127)* 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.

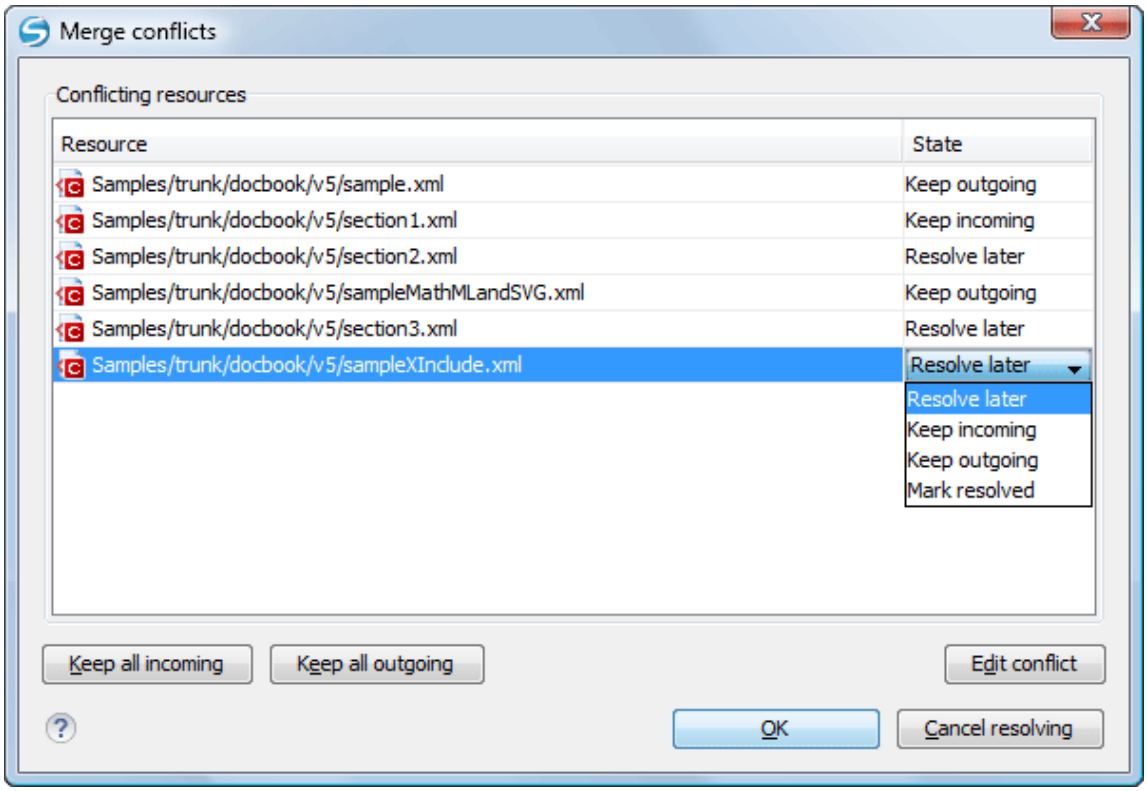
- **Ignore all whitespaces** - Ignores all types of whitespace changes.

- **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 548. 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 2127)* in the **Merge Options** wizard page *(on page 2126)* 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 2121)* operation. Also, trying to merge to mixed-revision working copies will fail in all types of merge operations.

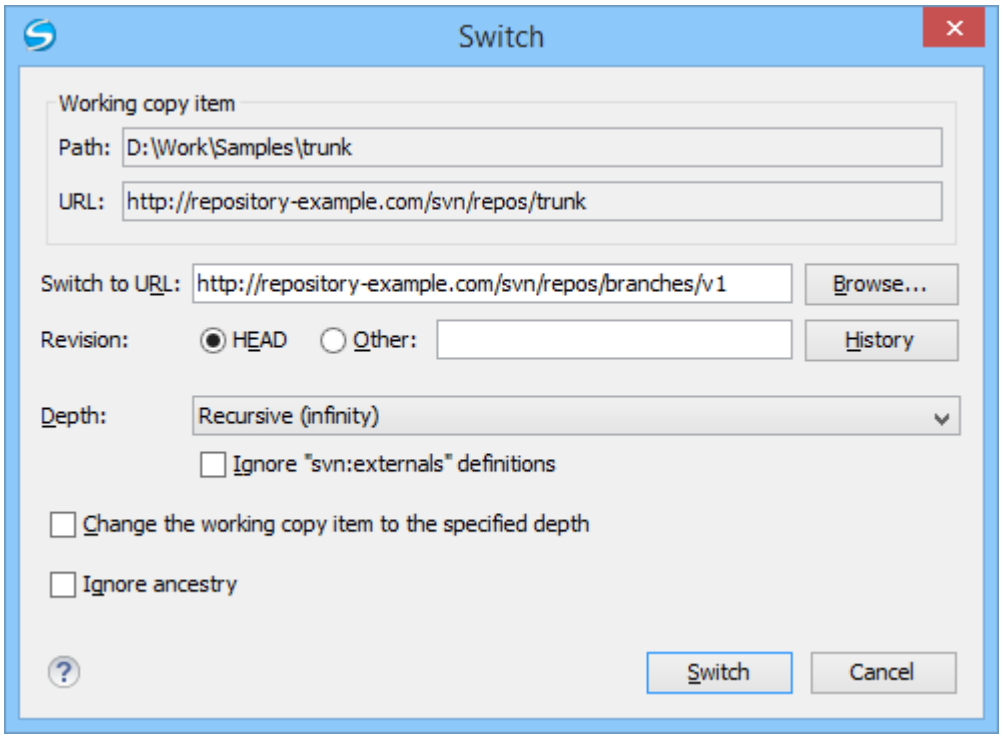
Remember:
The merge result is only in your local working copy and needs to be committed to the repository for it to be available to others.

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 2157)*.

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 549. 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 2193)* 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 2195)* at the end (for example, `URL@rev1234`).

> **Note:**
> For items that are already *switched (on page 2159)* 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 2193)*.

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 2145)*.

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.

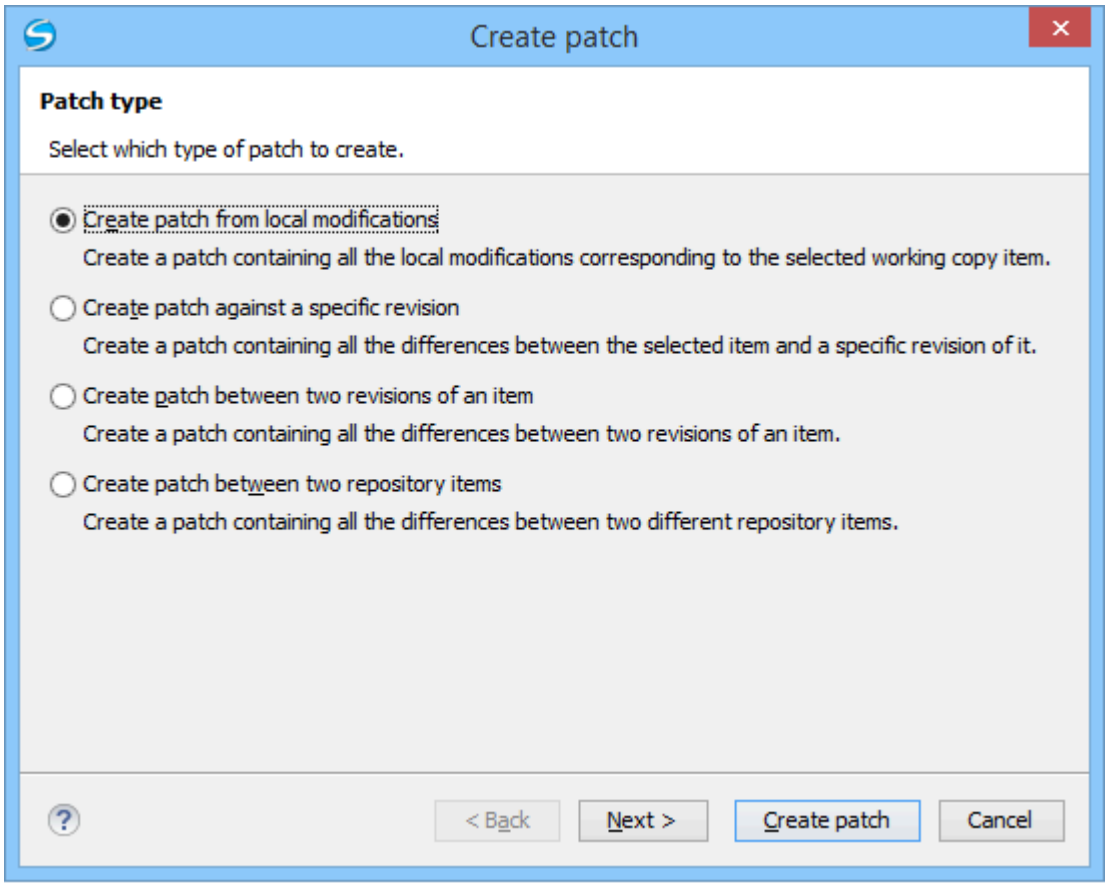
Warning:
The resulting patch file may contain content that was written using a mix of encodings, based upon the encodings of the files that were compared. If you open the generated patch file in a text editor, it may result in unrecognizable content.

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 550. Create Patch Wizard - Local Items



Create Patch from Local Modifications

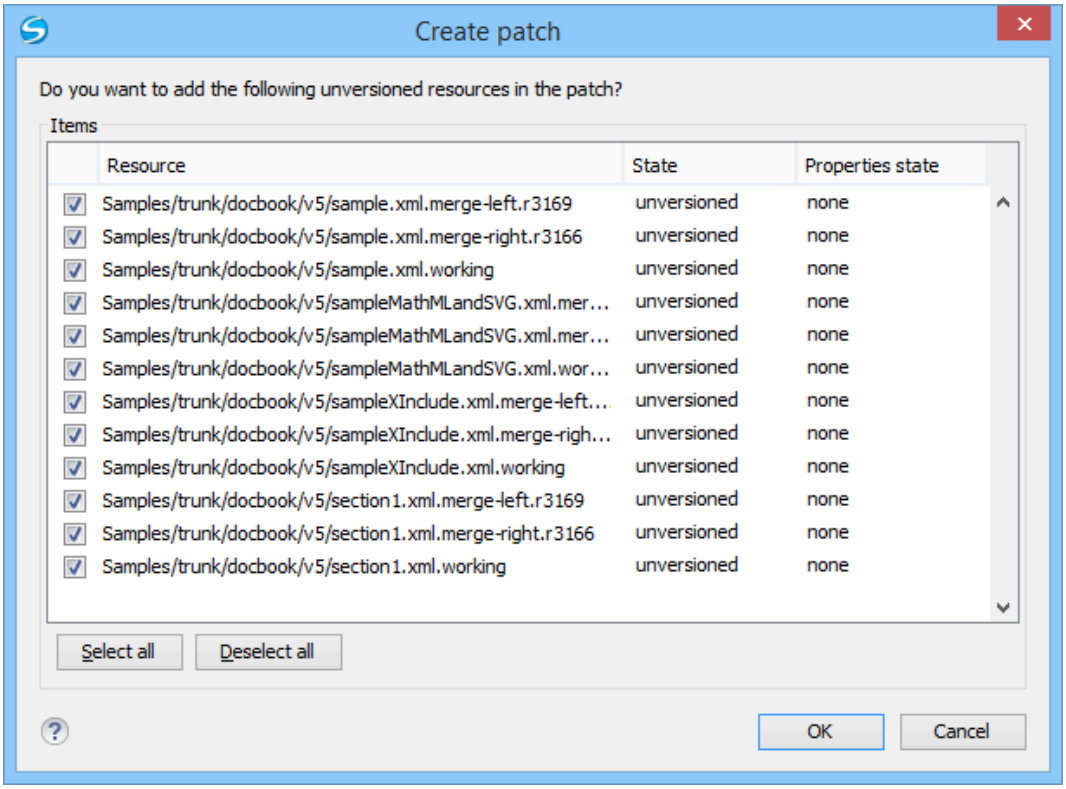
This is the most commonly used type of patch and contains only the local changes for the selected item.

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 2159)*.

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 2143)* 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 2159)*, this step of the wizard offers the option of selecting the ones that will be included in the patch.

Figure 551. Create Patch Dialog Box - Add Unversioned Resources



The patch is created and stored in the path specified in the **Output** section of the **Options** page *(on page 2143)* or in the default location.

Create Patch Against a Specific Revision

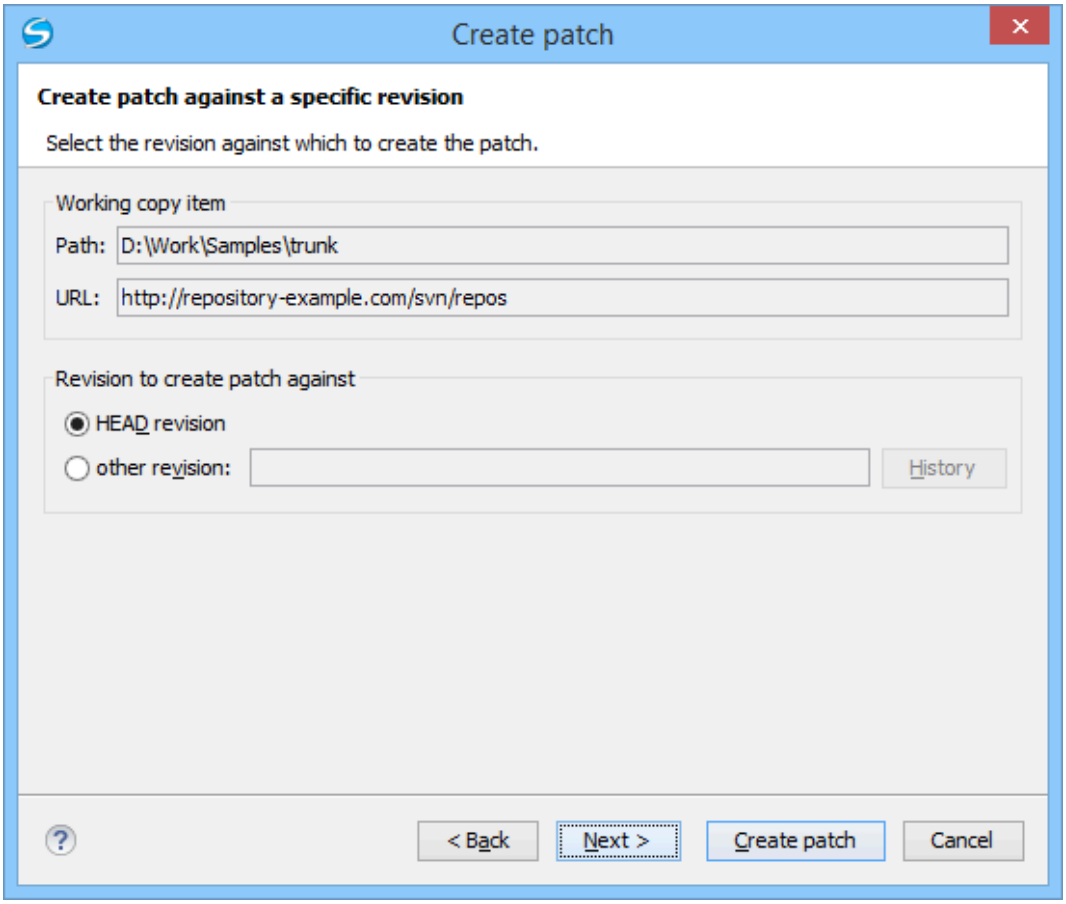
This type of patch contains changes between an old revision and the current content from the selected item within the working copy.

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 552. Create Patch Wizard - Step 2



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 2084)* to display a list of the item revisions.

   > **Note:**
   > If the **revision to create patch against** is older than the revision that the working copy item was updated for, the patch will include changes that were made **after** the selected revision.

5. Optionally, if you want to configure the options *(on page 2143)* 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 2143)* 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.

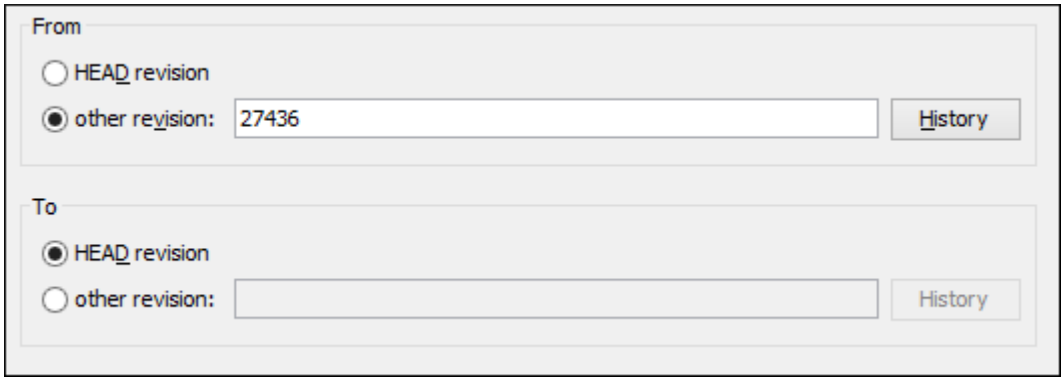
This option is useful if you want to share changes between two revisions with other team members.

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 2138)* 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 553. 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 2084)* 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 2143)* 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 2143)* 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.

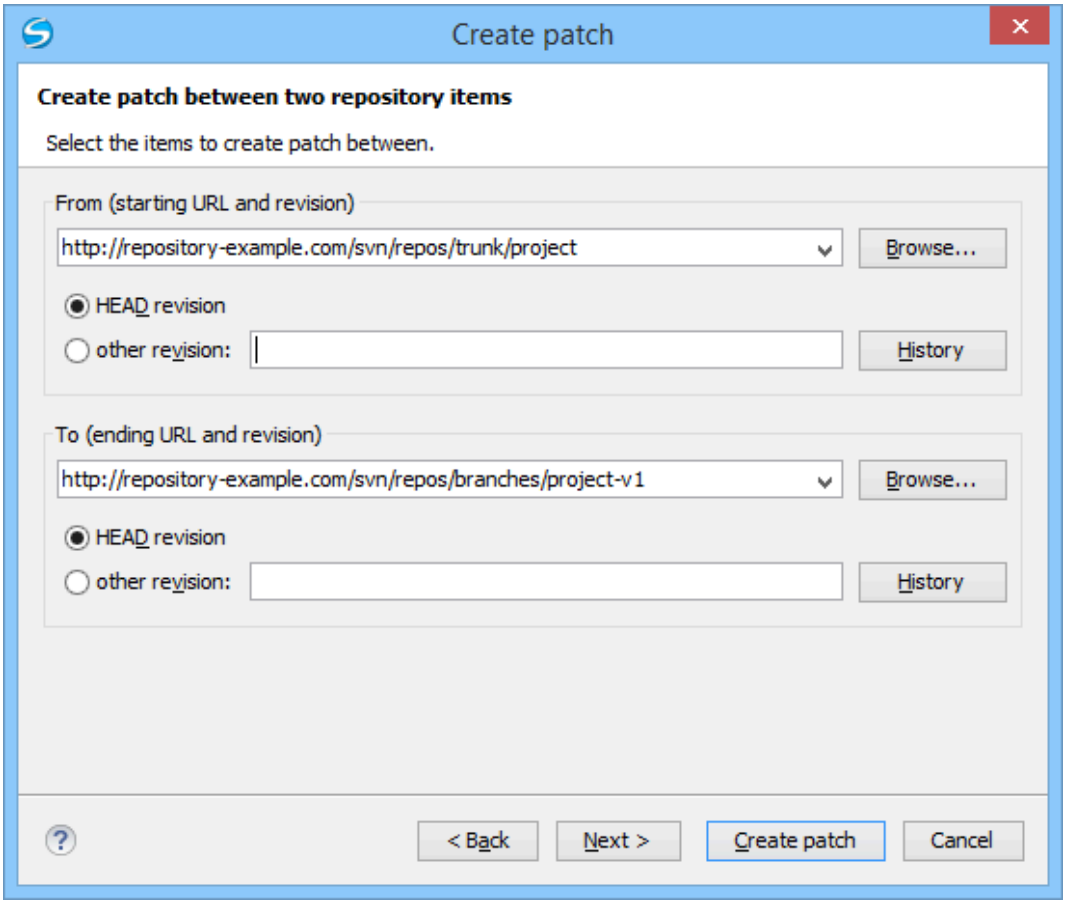


Tip:
To access an item that was deleted, moved, or replaced, you need to specify the original URL (before the item was removed) and use a *peg* revision *(on page 2195)* at the end (for example, `URL@rev1234`).

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 554. Create Patch Wizard - Step 2



4. Select the starting and ending URLs *(on page 2193)* 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 2084)* 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 2143)* 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.

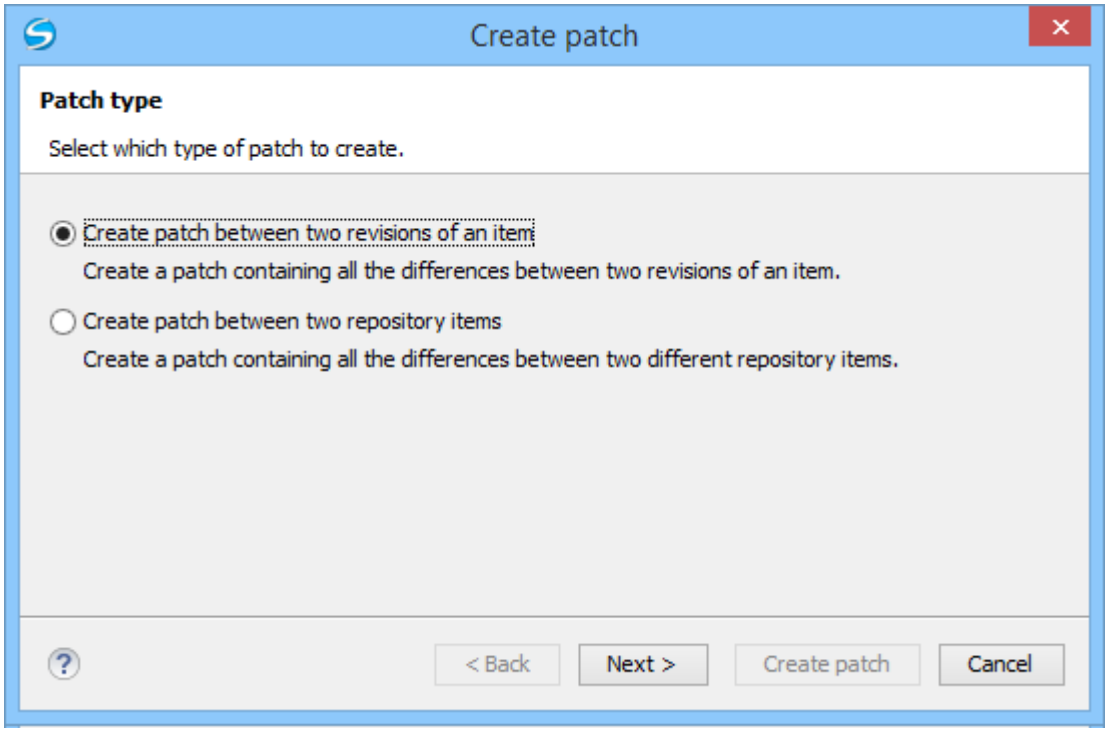
6. Click the **Create patch** button.
   The patch is created and stored in the path specified in the **Output** section of the **Options** page *(on page 2143)* or in the default location.

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.

Figure 555. Create Patch Wizard - Remote Items



Create Patch Between Two Revisions of an Item

This type of patch contains historical changes between two revisions of a selected item.

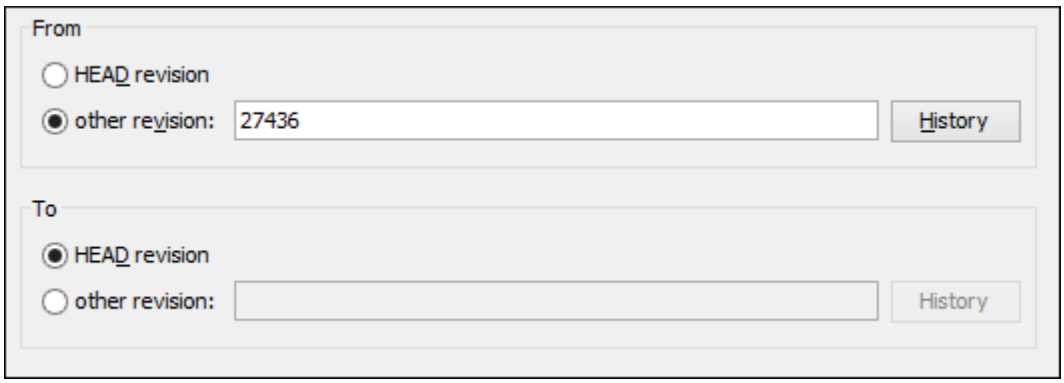
This option is useful if you want to share changes between two revisions with other team members.

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 2138)* 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 556. 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 2084)* 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 2143)* 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 2143)* 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.

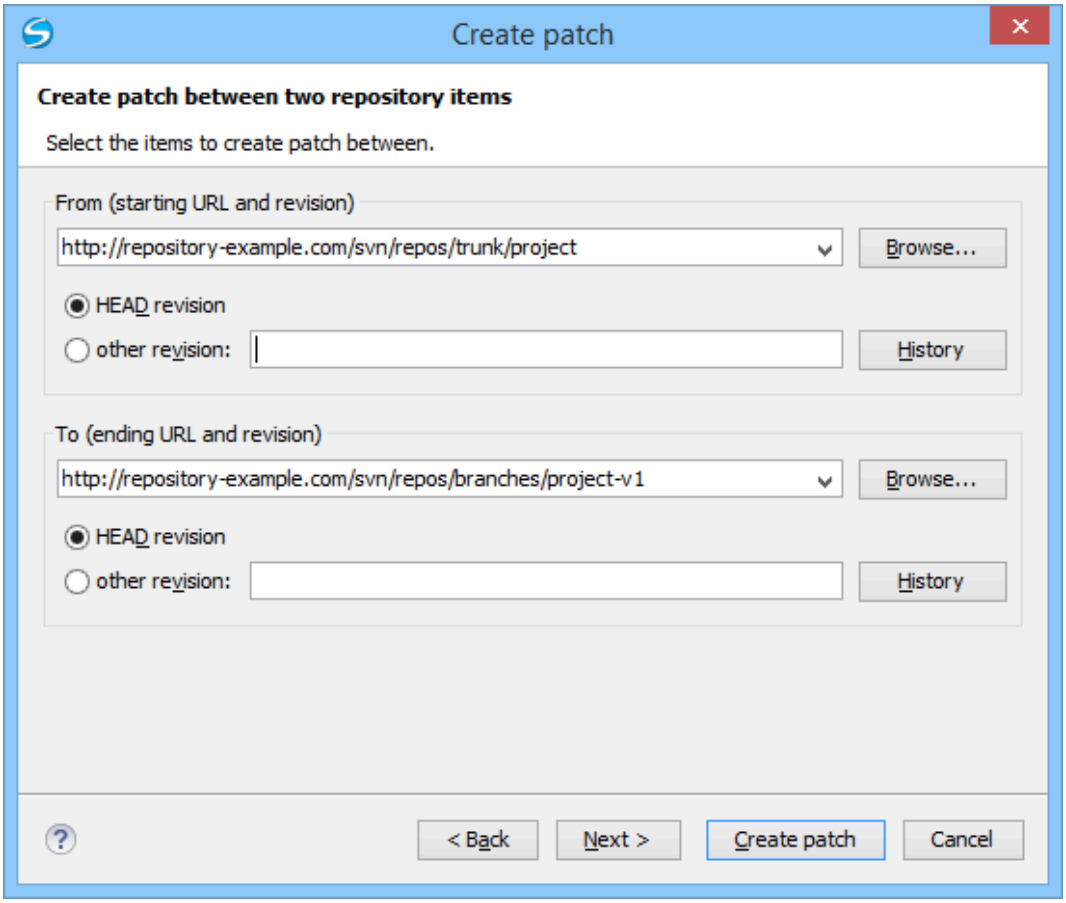
This option is useful for generating a patch that contains changes between existing, or even previously deleted, moved, or replaced items from different branches. This is the default option when you do not have a working copy loaded, when no repository items are selected, or when exactly two repository 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 2195)* 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 557. Create Patch Wizard - Step 2



4. Select the starting and ending URLs *(on page 2193)* 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 2084)* to display a list of the item revisions.

Important:
Both URLs must point to items from the same repository.



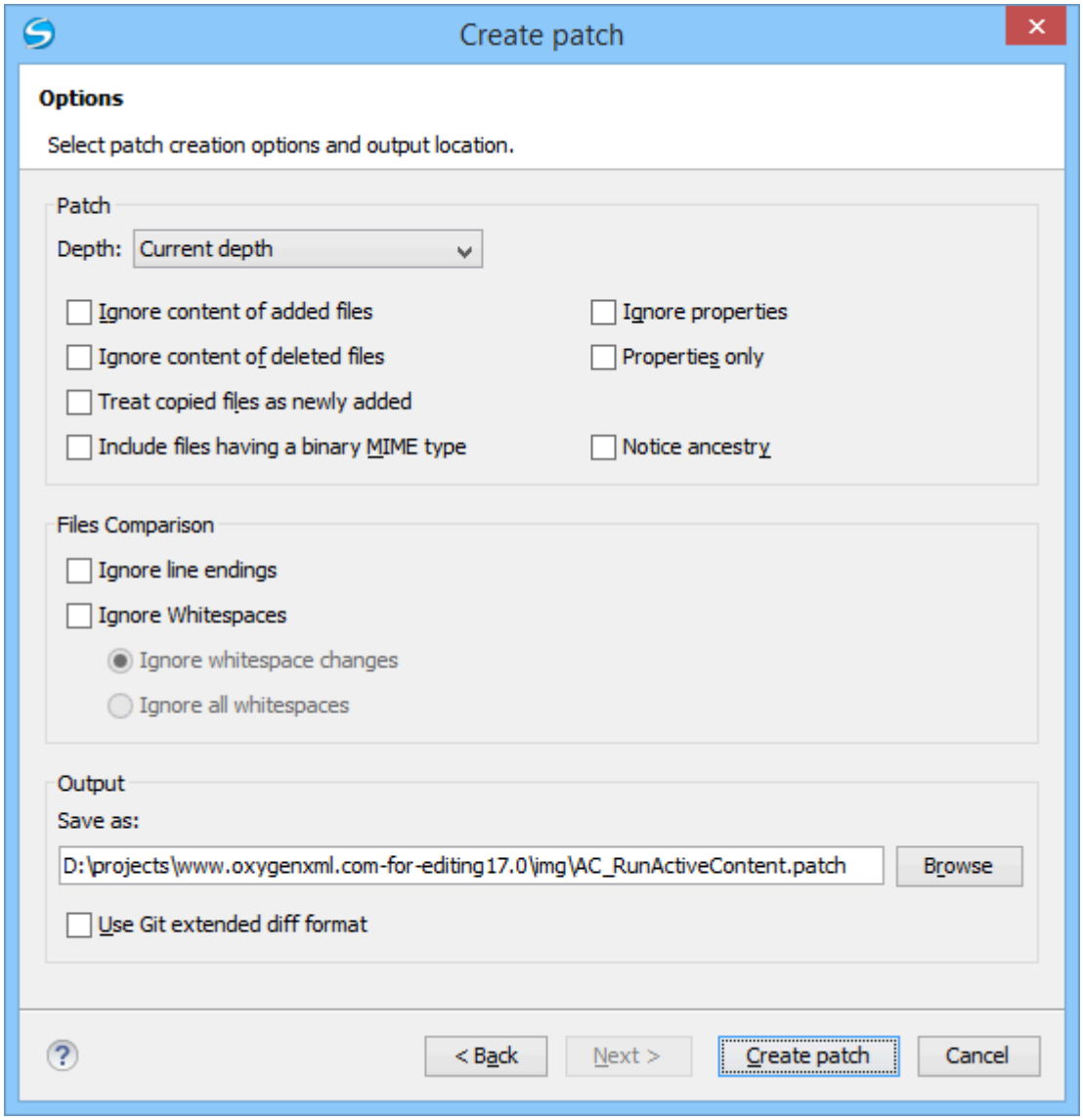
Note:
If you use a *peg* revision in the URL field, anything specified in the **other revision** field is ignored.

5. Optionally, if you want to configure the options *(on page 2143)* 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 2143)* or in the default location.

Patch Options

Figure 558. 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 `--properties-only` option of the `svn diff` command.

> **Note:**
> The **Ignore properties** and **Properties only** options are mutually exclusive.

Notice ancestry

If selected, the SVN common ancestry is taken into consideration when calculating the differences. This option corresponds to the `--notice-ancestry` option of the `svn diff` command.

Files Comparison Section

Ignore line endings

If selected, the differences in line endings are ignored when the patch is generated. This option corresponds to the `--ignore-eol-style` option of the `svn diff` command.

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** (`--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).

  > **Note:**
  > Whitespaces that are added or removed are still considered to be changes.

- **Ignore all whitespaces** (`--ignore-all-space`) - Ignores all types of whitespace changes.

Output Section

Save as

The patch will be created and saved in the specified file.

Use Git extended diff format

When selected, the patch is generated using the *Git* format. This option corresponds to the `--git` option of the `svn diff` command.

Working with Repositories

This section explains how to locate and browse SVN repositories in Syncro SVN Client.

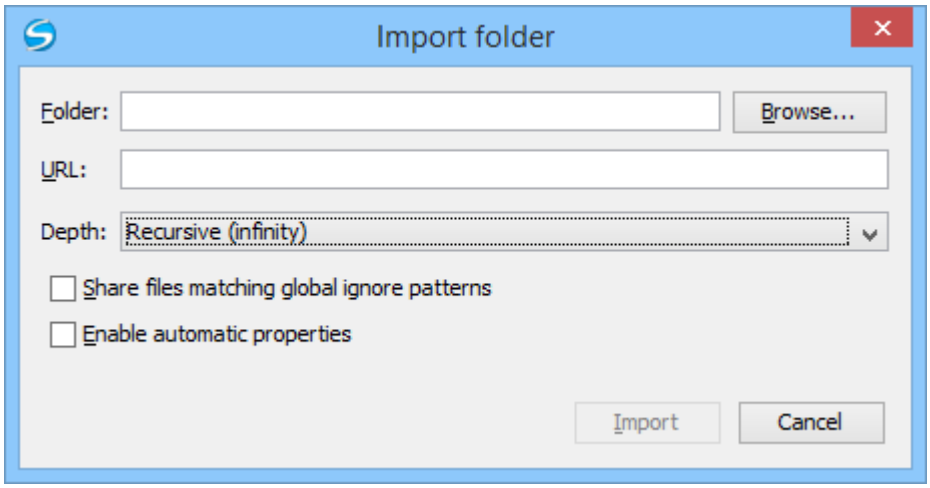
Importing Resources Into a Repository

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 2085)*.

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 559. Import Folder Dialog Box



You can configure the following options:

Folder

Specify the local folder *(on page 2192)* 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 2193)* 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 2192)*.
- The **File name ignore patterns** option *(on page 213)* in the **SVN > Working Copy** preferences page *(on page 212)*.

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 2192)*.



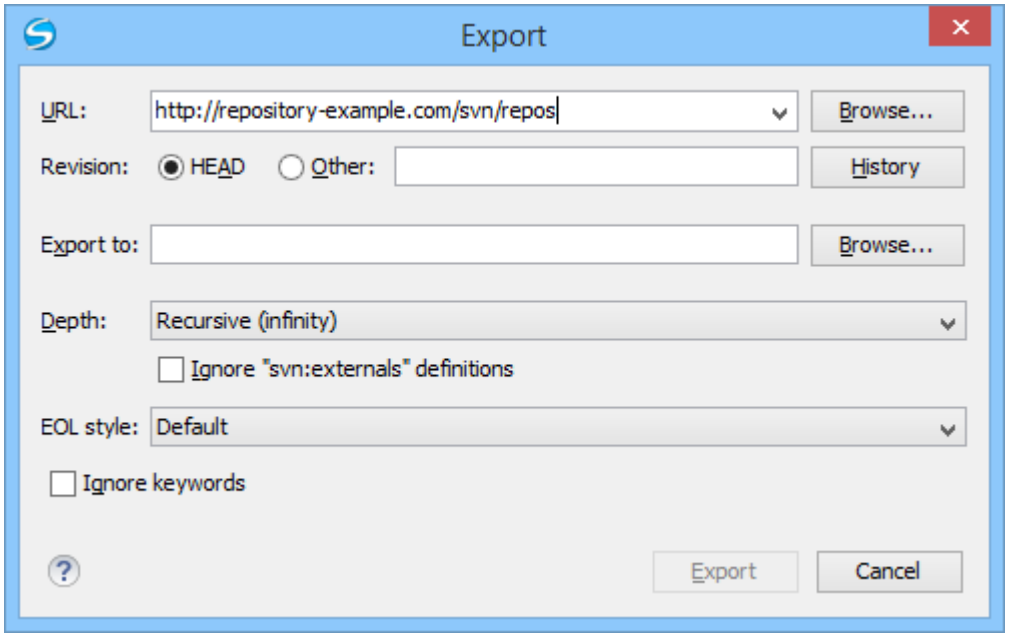
Note:
This option is available only when there are defined properties to be applied automatically for newly added items under version control. You can define these properties in the SVN `config` file (in the `auto-props` section). Based on the value of the `enable-auto-props` runtime configuration directive, 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 560. Export from Repository Dialog Box



You can configure the following options:

URL

Specify the source directory from the repository *(on page 2193)* 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 2195)* 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 2192)* 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.
- **CRLF** - The Windows-specific *end-of-line* marker (*carriage return - line feed*).
- **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

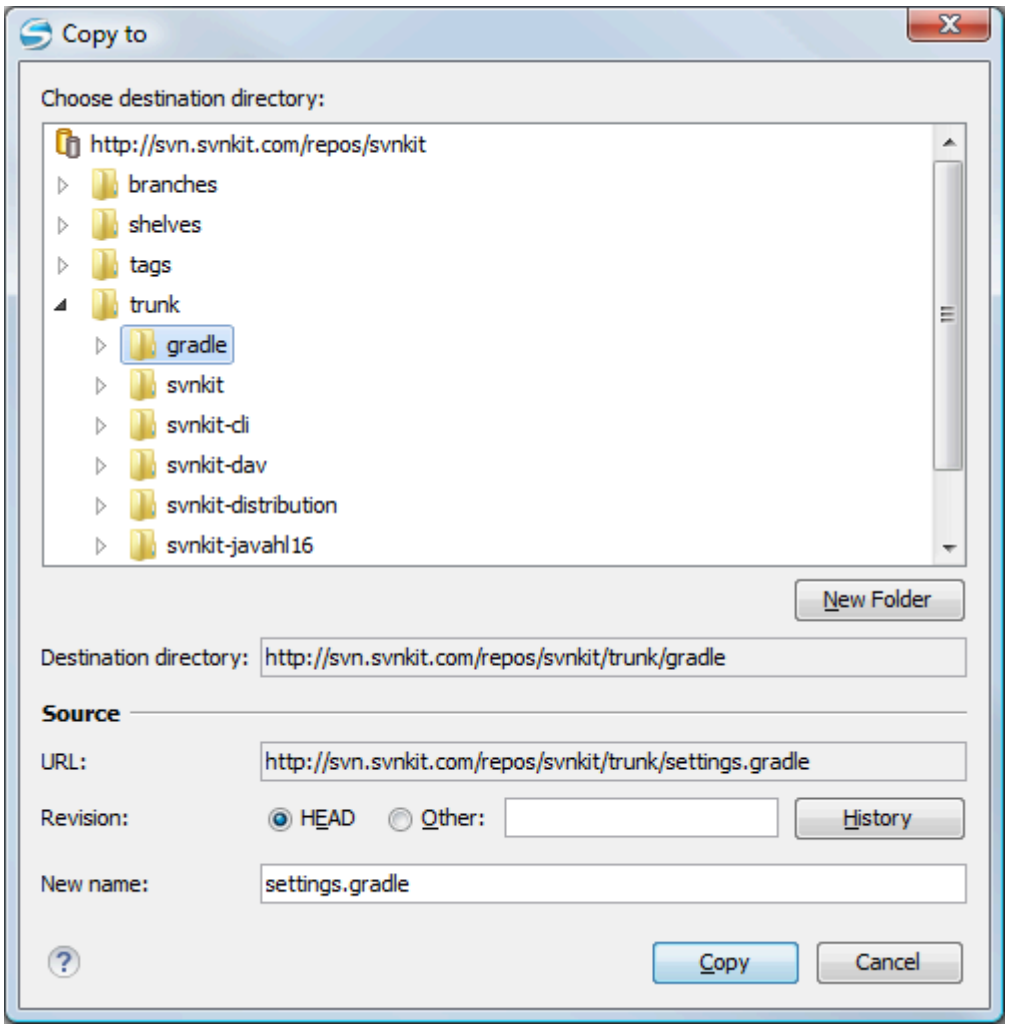
When selected, the export operation does not expand the *SVN keywords* found inside the files.

Copy / Move / Delete Resources From a Repository

Once you have a location defined in the **Repositories** view *(on page 2152)*, 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 561. 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 2153)*) and then recursively update only the needed directories (using the **Update** action from the **Working Copy** view *(on page 2165)*). 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 2157)* 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 2152)*
- Working Copy View *(on page 2157)*
- History View *(on page 2172)*
- Console View *(on page 2187)*

The other views that support the main working area are also presented in this section.

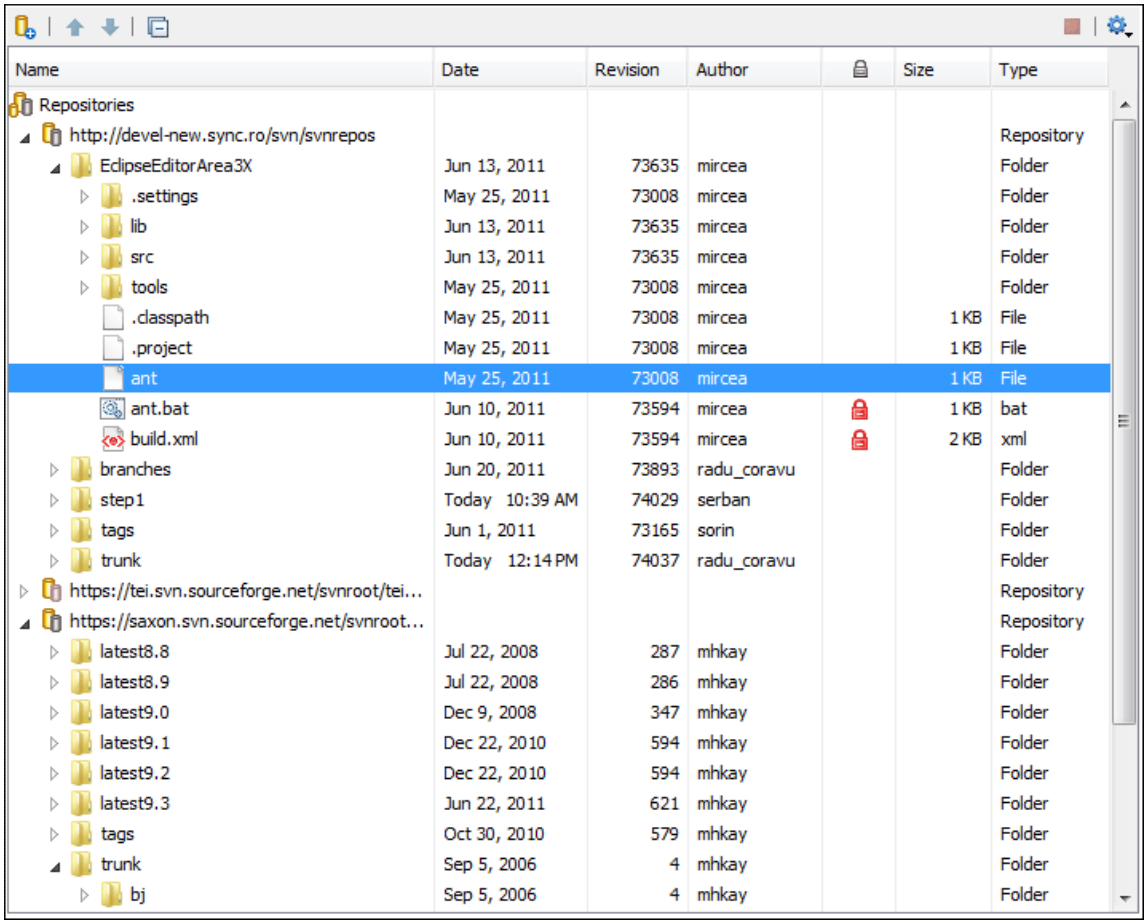
Repositories View

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 2077)*. 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.

Figure 562. Repositories View



Toolbar

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.

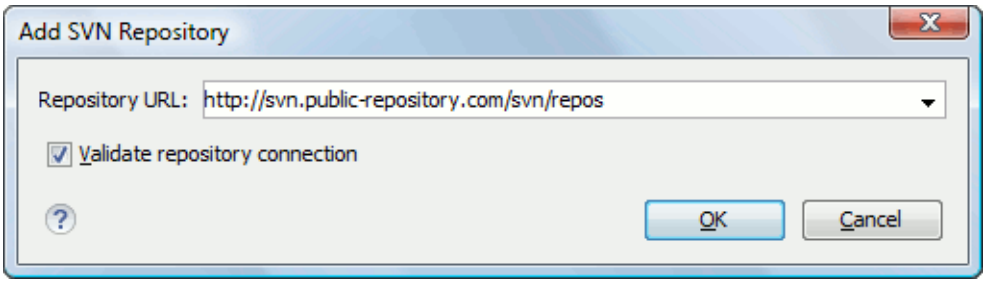
Repositories View Contextual Menu Actions

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 + Option + N on macOS))

Displays the **Add SVN Repository** dialog box. This dialog box allows you to define a new repository location.

Figure 563. 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 + Option + 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 + Option + 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 + Option + 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 + Option + 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 2082)*.

Branch/Tag

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 2111)* section.

Share project

Allows you to share a new project *(on page 2080)* 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 2146)*.

> **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 files system into the selected folder in the repository.

Export

Opens the **Export** dialog box *(on page 2147)* 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 2179)*, 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 2188)*. This operation is available for any resource selected in the **Repositories** view or **Working Copy** view.

Copy URL Location (Ctrl + Alt + U (Command + Option + 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 2093)*.

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 2185)* 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.

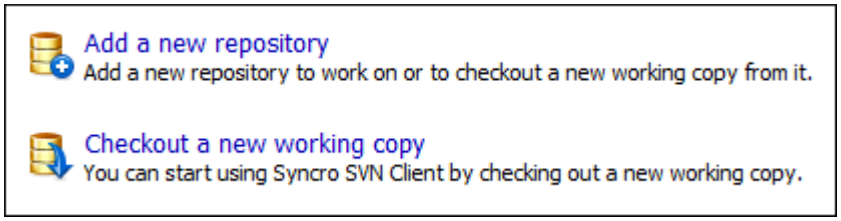
Show SVN Information (Ctrl + I (Command + I on macOS))

Provides additional information for the selected resource. For more details, go to Obtain information for a resource *(on page 2109)*.

Assistant Actions

When there is no repository configured, the **Repositories** view mode lists the following two actions:

Figure 564. Repositories View Actions



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.

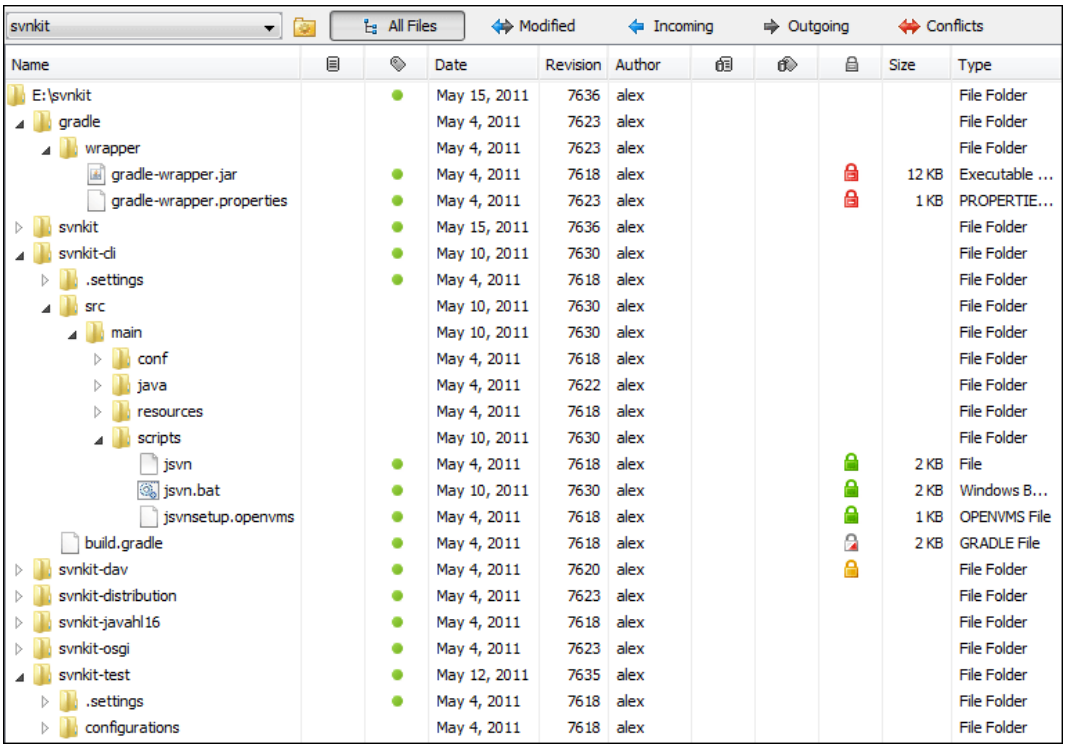
The toolbar contains the following:

- 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:

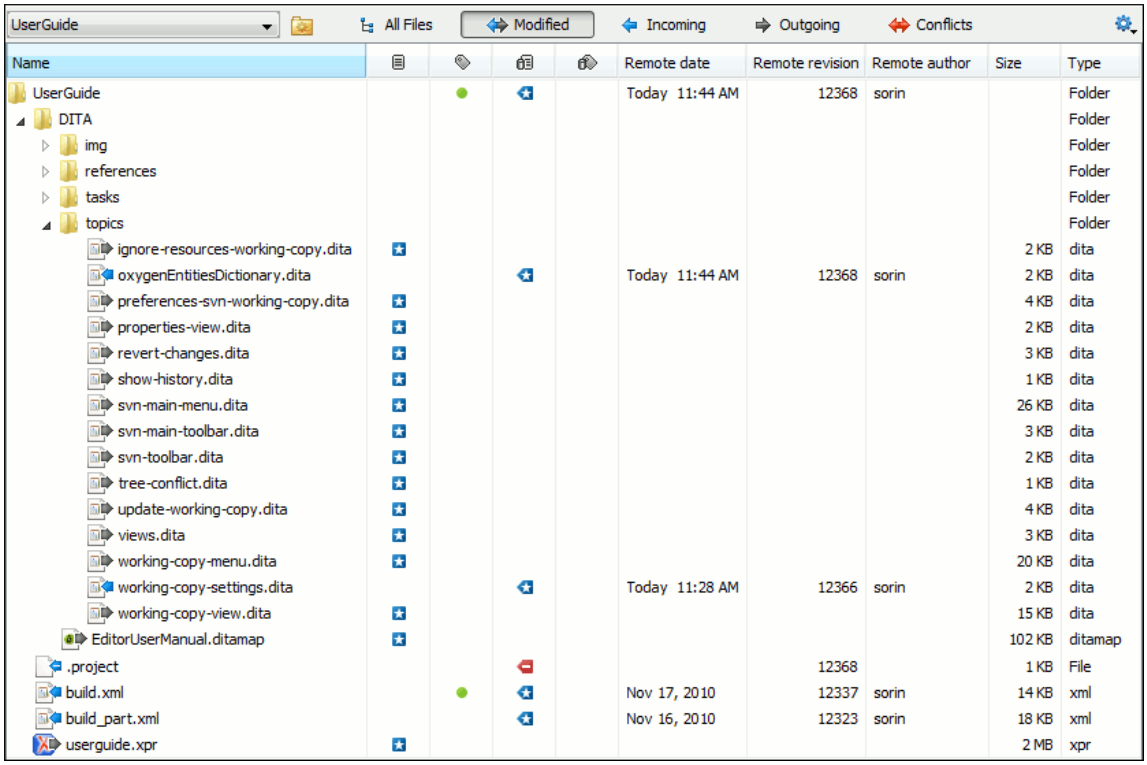
- **All Files** - Resources (files and folders) are presented in a hierarchical structure with the root of the tree representing the location of the working copy on the file system. Each resource has an icon representation that describes the type of resource and also depicts the state of that resource with a small overlay icon.

Figure 565. 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 566. 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** - 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** - 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** - 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 2130)* executed from the contextual menu of the Working Copy view.
    - **Grayed** - 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)** (a variant of sparse checkout *(on page 2151)*) - The directory contains only direct file and folder children. Child folders ignore their content.
    - **File children only (files)** (a variant of sparse checkout *(on page 2151)*) - The directory contains only direct file children, disregarding any child folders.
    - **This folder only (empty)** (a variant of sparse checkout *(on page 2151)*) - 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 Developer 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*).
  - - 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:

        - *global-ignores* section in the SVN client-side *config* file *(on page 2073)*.

          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 213)* of Oxygen XML Developer.
        - The value of a *svn:ignore* property *(on page 2088)* 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 2096)*.
    - - 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 2096)* 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 **Clean up** 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 **Repositories** 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 Developer 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 (on macOS)** - Opens a dialog box that displays the working copies Oxygen XML Developer 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 Developer 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.

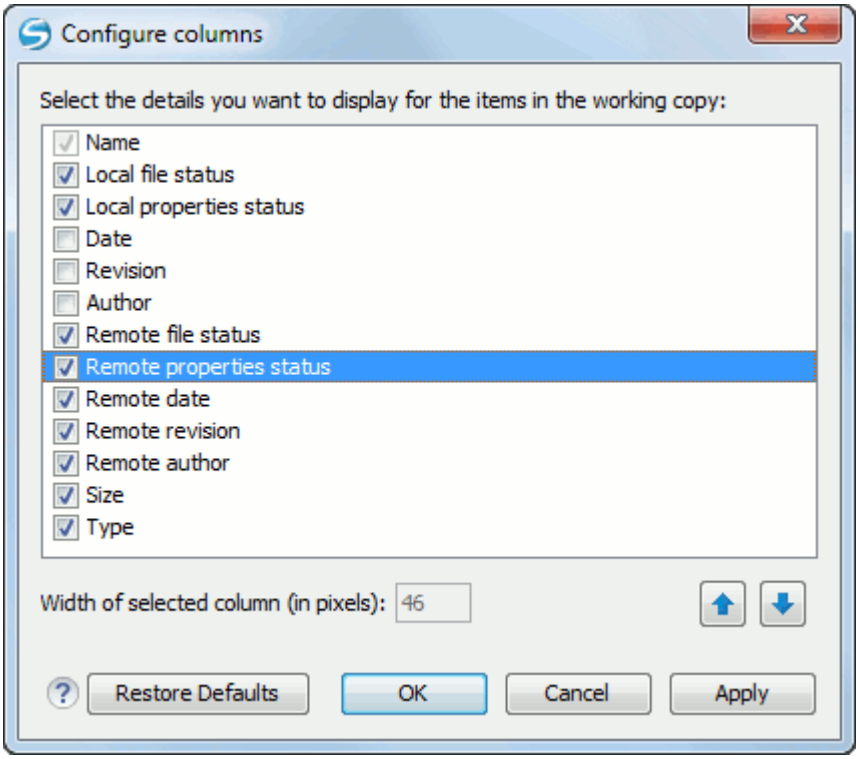
  > **Note:**
  > If this option is not selected, it will be ignored for items that, after a synchronize, are reported as incoming from the repository. This applies for all working copy modes, except **All Files**.

- **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 567. 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 212)*.

To change how working copy formats are handled, open the **Preferences** dialog box **(Options > Preferences)** *(on page 79)*, go to **SVN > Working copy**, and configure the options in the **Administrative area** *(on page 212)* 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 Developer 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 2098)*.

Open in Compare Editor (Ctrl + Alt + C (Command + Option + 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 Developer 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 + Option + 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 212)* 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 2151)* 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 2100)* 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 2100)* 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 2102)*.

 **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 2101)*.

 **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 2101)* section for more information about how you can solve the pseudo-conflicts.

 **Create patch (Ctrl + Alt + P (Command + Option + 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 2135)*.

Compare with:

- **Latest from HEAD (Ctrl + Alt + H (Command + Option + 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 + Option + C on macOS))** - Compares the working copy file with the BASE revision file (the so-called *pristine copy*).
- **Revision (Ctrl + Alt + R (Command + Option + 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 2193)* (**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 2195)* 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 `svn:ignore` or `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 2172)* and requesting the history for a resource *(on page 2110)*.

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 2179)*, 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 2188)*.

Copy URL Location (Ctrl + Alt + U (Command + Option + 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 2193)*, 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 2195)* 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 + Option + 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 2088)* section.

Add to version control (Ctrl + Alt + V (Command + Option + 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 2086)* 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 (Comman + 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 2092)*.
- **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 2093)*.
- **Unlock (Ctrl + Alt + K (Command + Option + 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 2185)* 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 2109)*.

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.

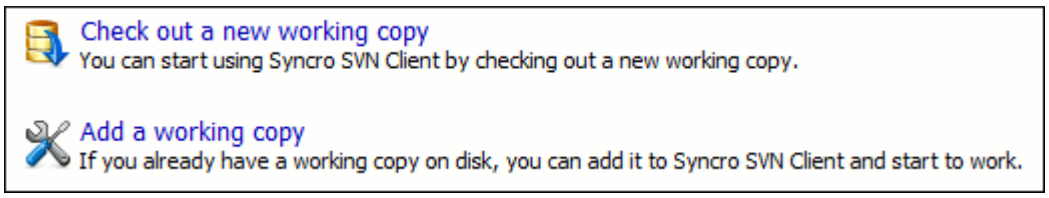
Attention:
When you drag items from the working copy to a different application, the performed operation is controlled by that application. This means that the moved items are left as *missing* in the working copy (items are moved in the file system only, but no SVN versioning meta-data is changed).

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 568. 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 2152)*, Working Copy view *(on page 2157)*, Revision Graph *(on page 2188)*, or Directory Change Set view *(on page 2177)*. 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:

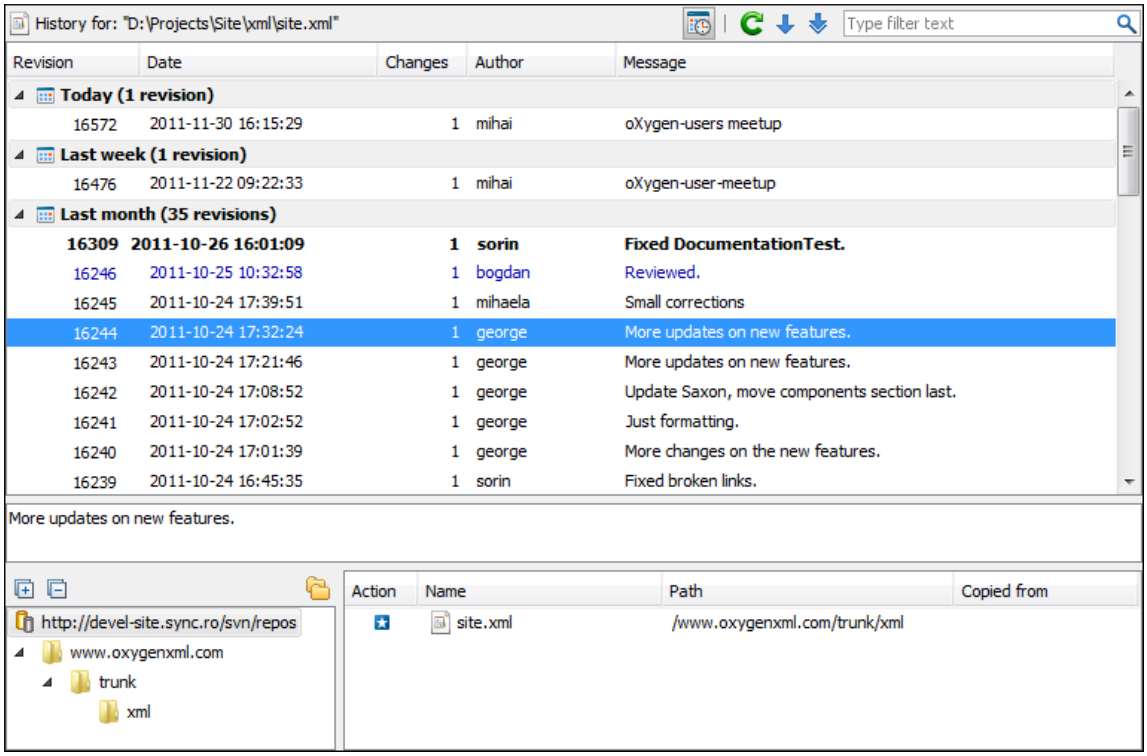
- The table showing details about each revision, such as revision number, commit date and time, number of changes (more details available in the tooltip), author's name, and a fragment of the commit message.

  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.

Figure 569. History View

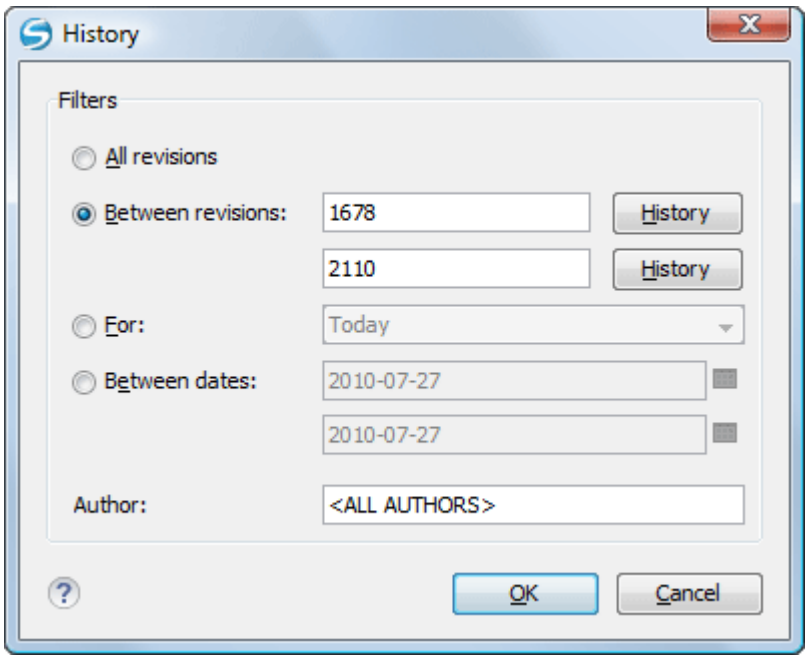


You can group revisions in predefined time frames (today, yesterday, this week, this month), by pressing the **Group by date** button from the toolbar.

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 570. 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 2147)* 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 2179)*, 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

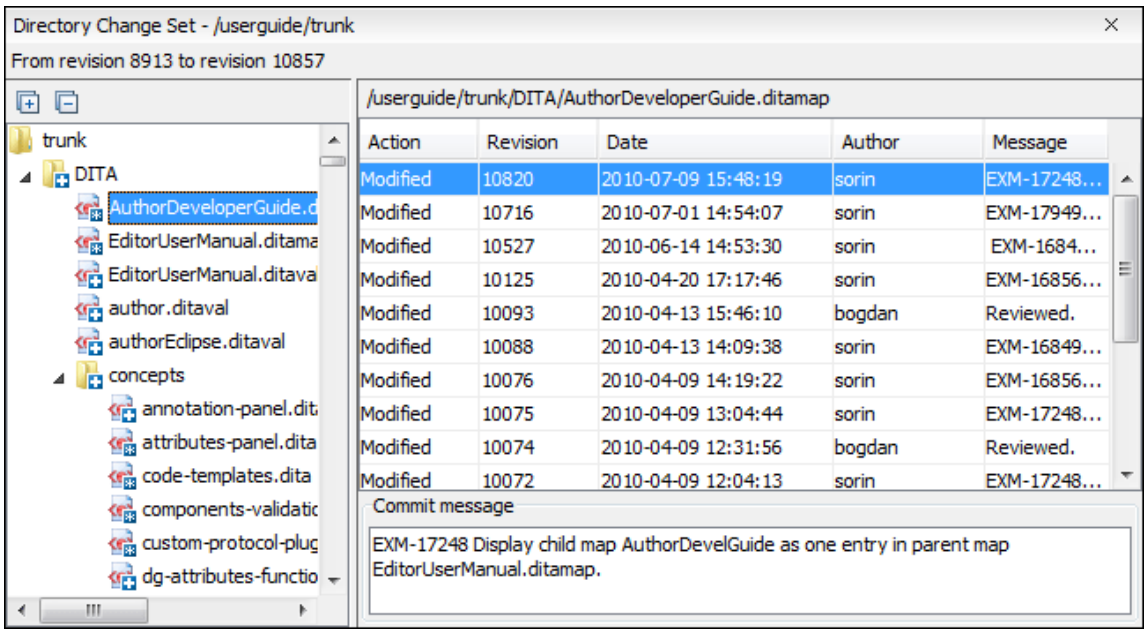
Similar to the `svn merge` command, it merges two selected revisions into the working copy resource. This action is only available when the resource history was requested for a working copy item.

For more information about the **History view** and its features, see the Request history for a resource *(on page 2110)* and Using the resource history view *(on page 2172)* 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 571. 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 2147)* 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 2179)*, 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 2109)*.

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 226)*.

The internal editor can be accessed either from the Working copy view *(on page 2157)* or from the History view *(on page 2172)*. 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 2061)* 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 2157)*.

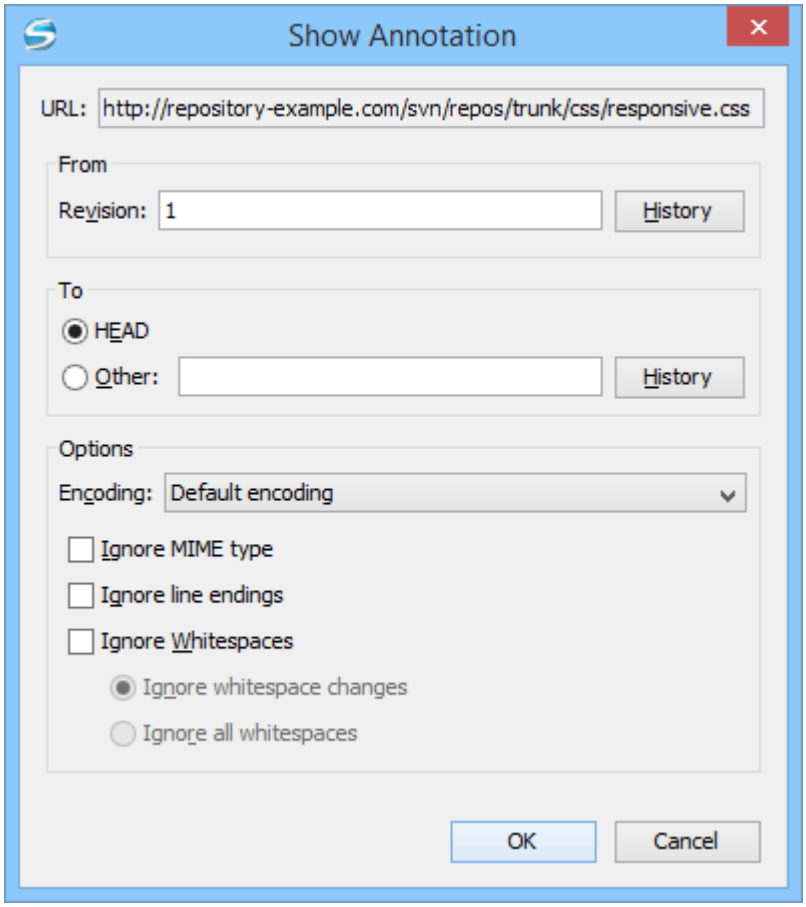
If the internal editor associated with a file type is not the XML Editor, then the encoding set in the preferences for *Encoding for non XML files (on page 123)* is used for opening and saving a file of that type. This is necessary because in the case of XML files, the encoding is usually declared at 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 2152)*, **Working copy** view *(on page 2157)*, **History** view *(on page 2172)*, and **Directory Change Set** view *(on page 2177)*.

This action opens a dialog box that allows you to configure some options for showing the annotations.

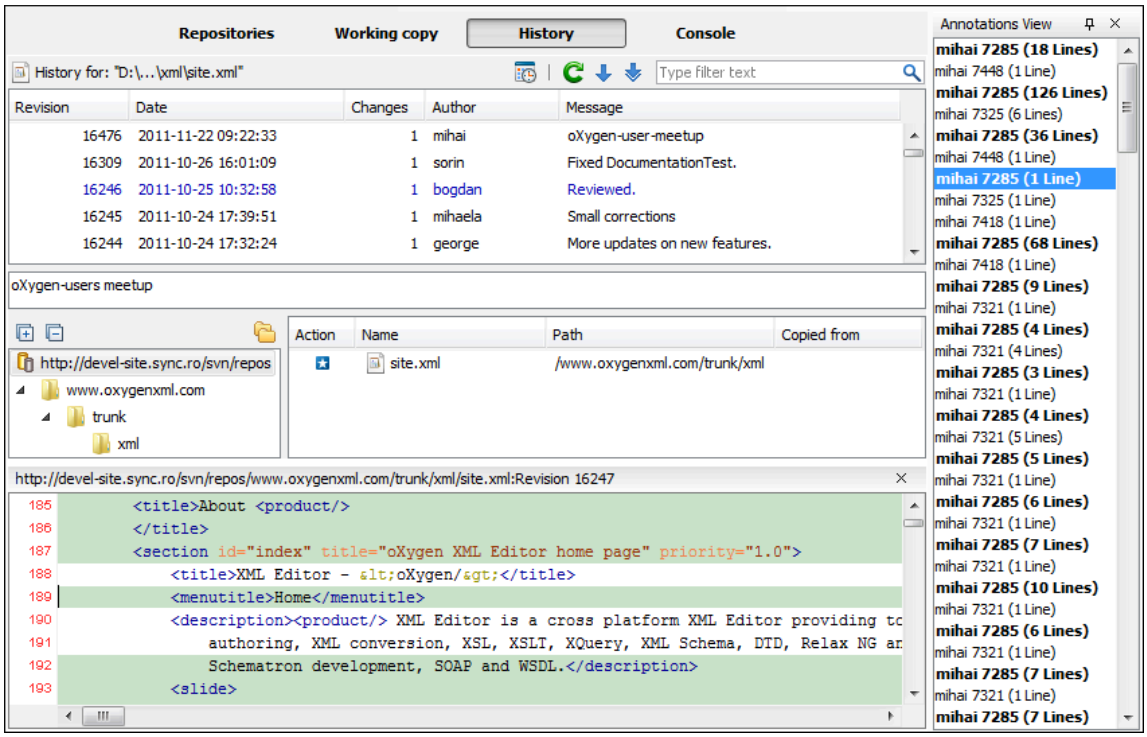
Figure 572. Show Annotation Options Dialog Box



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 573. 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 2084)* 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 2084)* 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.

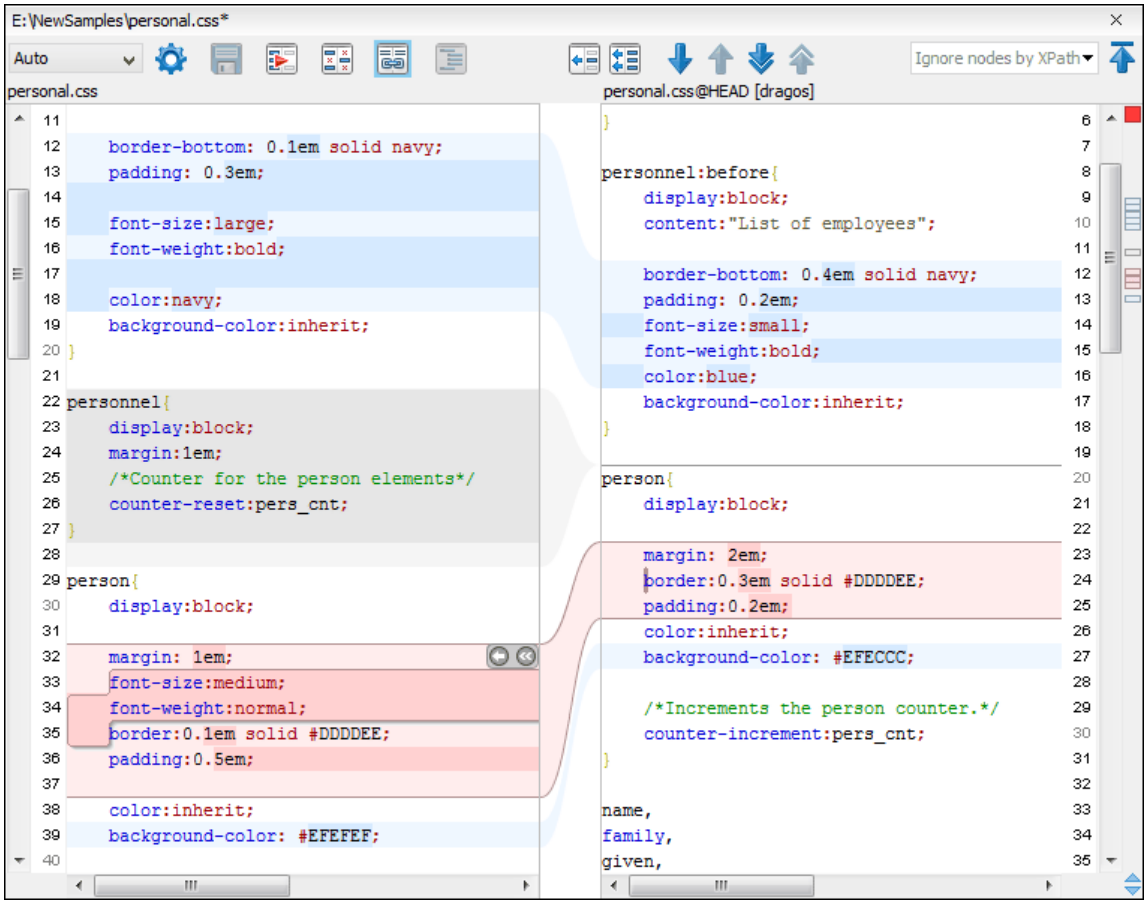
After you configure the options and click **OK**, the annotations will be computed and the **Annotations** view is displayed, where all the users that modified the selected resource will be presented, along with the specific lines and revision numbers modified by each user.

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 Developer, 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.

Figure 574. Compare View



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 2094)*.

Compare View Toolbar

The toolbar of the **Compare** view contains the operations that can be performed on the source and target files.

Figure 575. Compare View Toolbar



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.
- **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. If content is selected in the left panel, only that selection is affected, copying only the corresponding changes from the right to the selection on the left. The action name is also automatically updated to **Copy Changes in Selection from Right to Left** in this case.

 **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 1726)* 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 217)* 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 2061)* 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 2157)* or from the Repository view *(on page 2152)*. It can also be used from the History view *(on page 2172)* 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

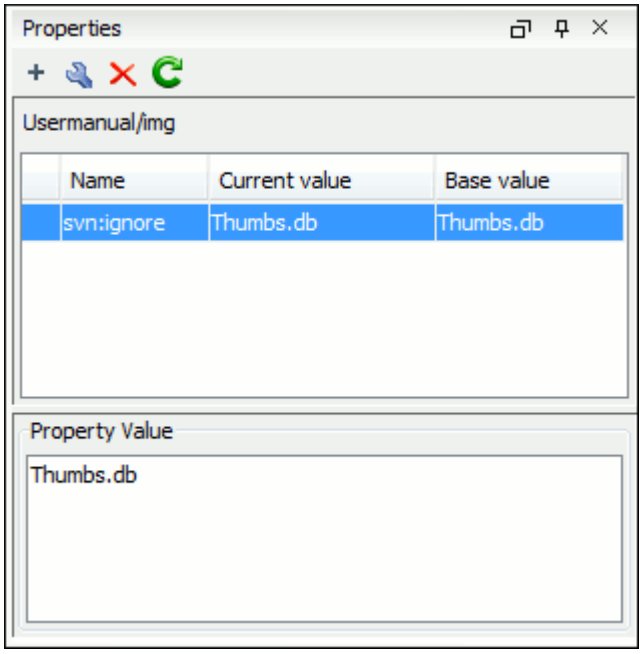
The images are compared using the **Compare Images** view. This view is automatically opened if you select two image files and use the **Compare with > Each Other** action in the contextual menu. The images are presented in the left and right part of the view, scaled to fit the available area. You can 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 576. 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 2169)*.

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 210)* page. By default, this value is set to 100.

Dynamic Help View

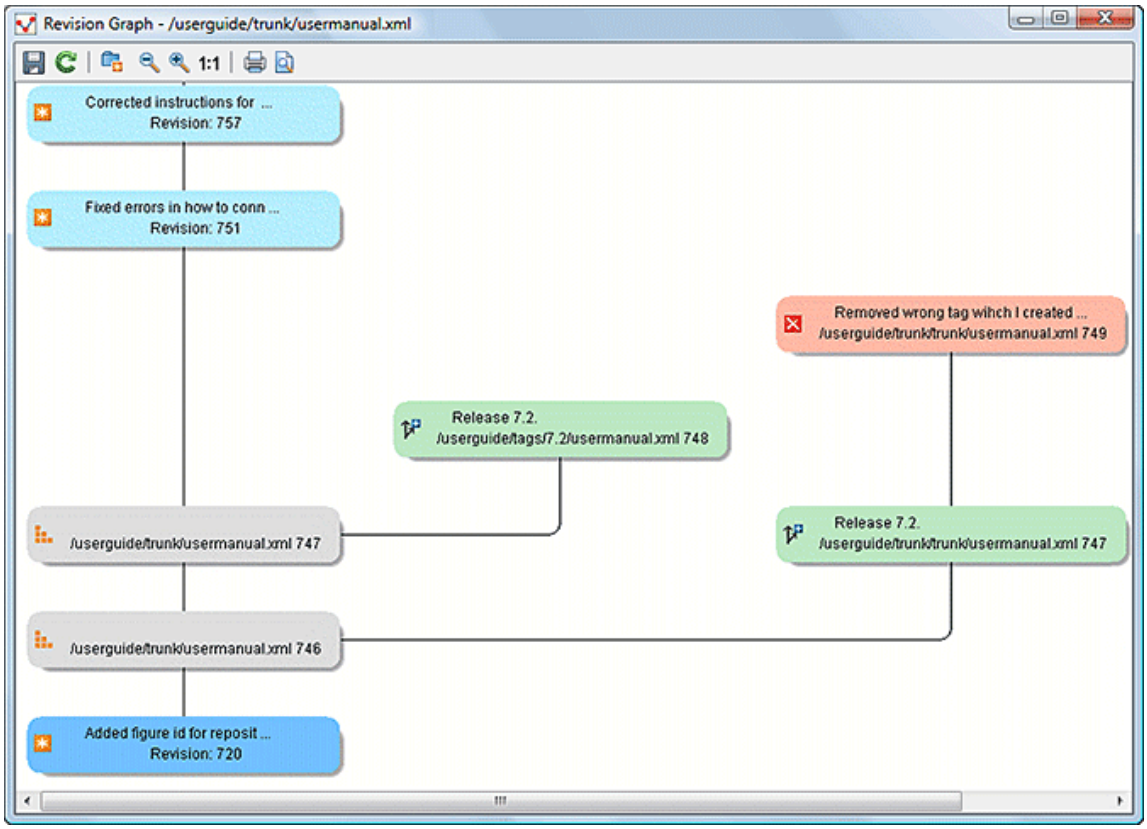
Dynamic Help view is a help window that changes its content to display the help section that is specific to the currently selected view. As you change the focused view, you can read a short description of it and its functionality. If the view is not displayed, it can be opened by selecting it from the **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 2157)* and the *Repository* view *(on page 2152)*.

Figure 577. Revision Graph of a File Resource



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 2084)*.

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 578. Revision Graph of a Directory (Direct Changes)

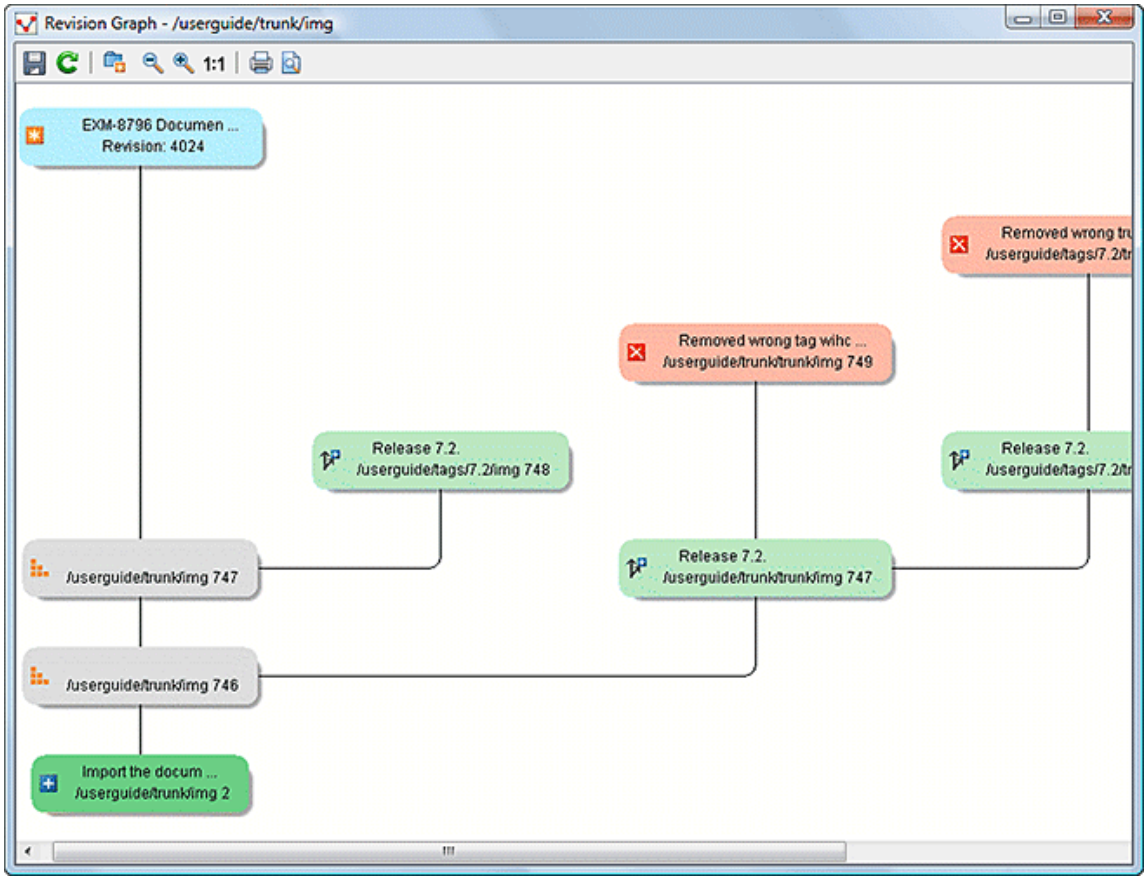
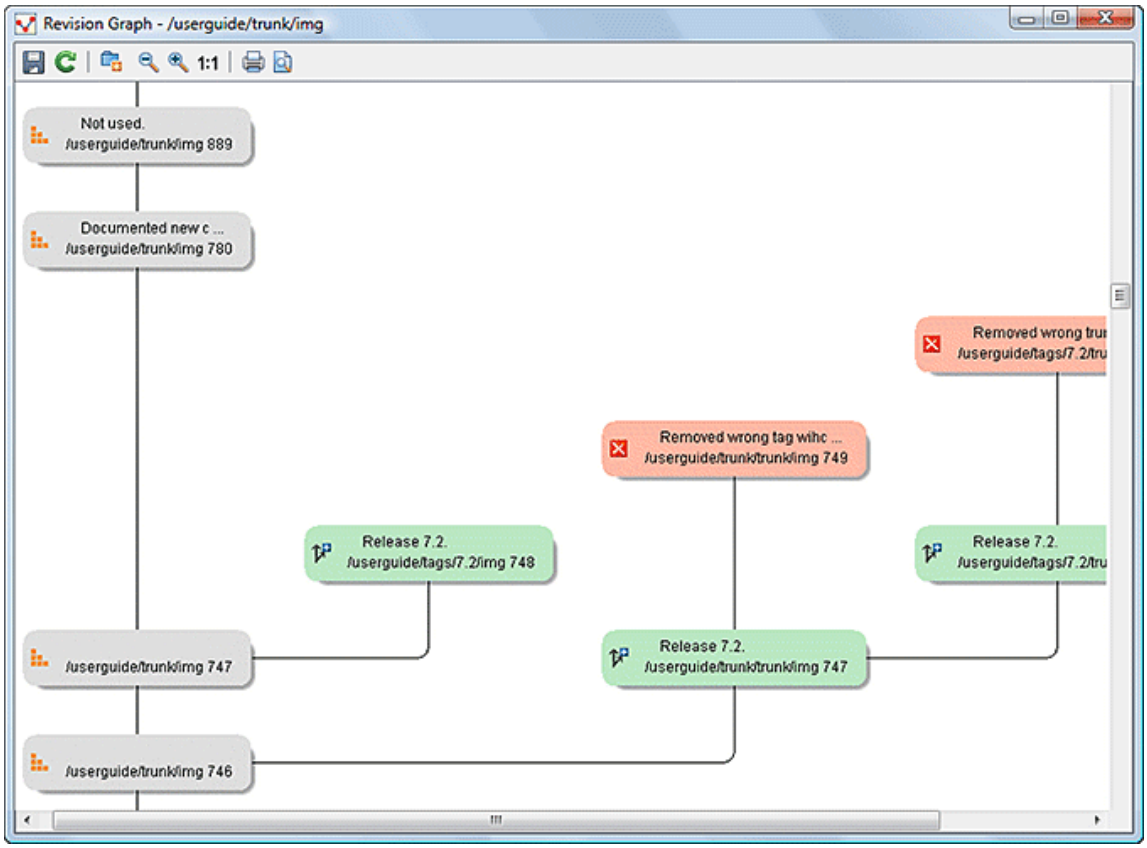


Figure 579. Revision Graph of a Directory (Including Indirect Changes)



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 267)*. 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 2172)*. Available for both files and directories.

Check out

Checks out *(on page 2082)* the selected revision of the directory. Available only for directories.

Export

Opens the **Export** dialog box *(on page 2147)* 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 2177)* 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 Developer SVN Preferences

The options used in the SVN client are saved and loaded independently from the Oxygen XML Developer options. However, if Oxygen XML Developer 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 Developer 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 Developer 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 Developer expects absolute paths for local file system items.
- *Relative Paths* - The Oxygen XML Developer only accepts relative paths in the form `~[/...]`, where `~` is the user home directory.
- *Path Validation* - Oxygen XML Developer 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 Developer 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 Developer expects absolute paths for local file system items.
  - *Relative Paths* - The Oxygen XML Developer only accepts relative paths in the form `~[/...]`, 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 2195)* (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 Developer 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

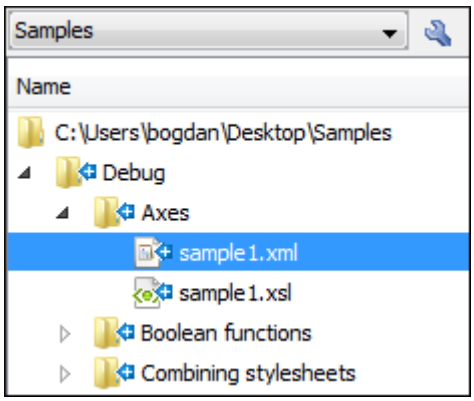
If Syncro SVN Client prompts you to enter the authentication certificate, although you already provided it in a previous session, then you should make sure that your local machine user account has the necessary rights to store certificate files in the *Subversion* configuration folder (write access to *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 580. 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 2166)* 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 2159)* 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 2159)*.
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 Developer user interface as if from the command line of the operating system shell. Oxygen XML Developer 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 79)* and go to **External Tools** *(on page 221)* (or select **Configure** from the **Tools > External Tools** menu).

The **External Tools** preferences page *(on page 221)* presents a list of the external tools that have been configured. Once a tool has been configured *(on page 221)*, 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 291)*). You can also assign a keyboard shortcut *(on page 223)* to be used to launch the tool.

If the external tool is applied on one of the files opened in Oxygen XML Developer, the **Save all files before calling external tools** option *(on page 135)* (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 Developer:

1. Download and install *Apache Ant (on page 2303)* 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 `C:\projects\XMLproject\build.xml`:
   ```
   ant clean
   ```
3. Open the **Preferences** dialog box **(Options > Preferences)** *(on page 79)* 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
   ```

AI Positron Overview

Oxygen XML Developer offers an **Oxygen AI Positron** add-on that provides an advanced solution for using AI assistance to enhance the documentation content creation process. By leveraging the power of AI platforms, the **AI Positron** streamlines and enhances the writing experience by providing a variety of useful AI based actions and features, the possibility to chat and refine responses, and lots of customization possibilities for defining new custom AI actions and behaviors.

The add-on contributes an **AI Positron** side-view. To initiate the installation process for the add-on, select **AI Positron** from the **Tools** menu. Once installed, selecting the action opens the **AI Positron** view.

For information about the **Oxygen AI Positron** and its features, see the Oxygen AI Positron for Desktop user guide.

AI Translator Overview

Oxygen XML Developer offers an **Oxygen AI Translator** add-on provides AI-powered translation capabilities for XLIFF (XML Localization Interchange File Format) files. This add-on streamlines the translation workflow by leveraging advanced AI models to produce high-quality translations while maintaining the structural integrity of your XML content.

The add-on contributes an **AI Translator** side-view. To initiate the installation process for the add-on, select **AI Translator** from the **Tools** menu. Once installed, selecting the action opens the **AI Translator** view.

For information about the **Oxygen AI Translator** and its features, see the Oxygen AI Translator add-on.

20.
Troubleshooting

This section provides a collection of common performance and other types of problems that might be encountered when using Oxygen XML Developer, along with their possible solutions.

Getting Help

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

- The Oxygen XML Developer Support Section of the Website
- The Oxygen XML Developer Forum
- The Oxygen XML Developer Video Tutorials
- The Common Problems and Solutions Section of the User Manual *(on page 2200)*
- The Online Technical Support Form

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

Help Menu

The Oxygen XML Developer **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 Developer.

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.

Show Dynamic Help view

Use this action to open a view that loads the latest online WebHelp version of the Oxygen XML Developer User Manual, and dynamically opens a topic that is relevant to the focused editor, view, or dialog box.

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

Download User Manual

Opens the appropriate HTML page for downloading the Oxygen XML Developer User Manual. This action is helpful if your internet access is restricted and you need to access the user guide.

Install new add-ons

Opens a dialog box that allows you to install new *add-ons (on page 2307)* to extend the functionality of Oxygen XML Developer.

> **Attention:**
> To ensure safety, when using an add-on site, make sure it is reputable and trustworthy. Do not install add-ons that are unsigned or from sources that you do not fully trust.

Check for add-ons updates

Opens a dialog box that allows you to check for updates on installed *add-ons (on page 2307)*.

Manage add-ons

Opens a dialog box that allows you to manage installed *add-ons (on page 2307)*.

Check for a New Version

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

Check for notifications

Use this action to have the application check **Oxygen's** website for news and events that will be displayed in a dialog box.

Browse Oxygen Website

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

Register

If you encounter problems with your Oxygen XML Developer 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 56)* 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 2307)* with your report.
- **Frameworks** - You can choose whether or not to include details about your installed *frameworks (on page 2305)* with your report.

Support Center

Use this option to open the Oxygen XML Developer 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 Developer. 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 2203)*.

> **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 Developer.

About

Use this option to open a dialog box that contains information about Oxygen XML Developer 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 Developer 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 2305)* that are bundled with Oxygen XML Developer.
- **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 2215)*

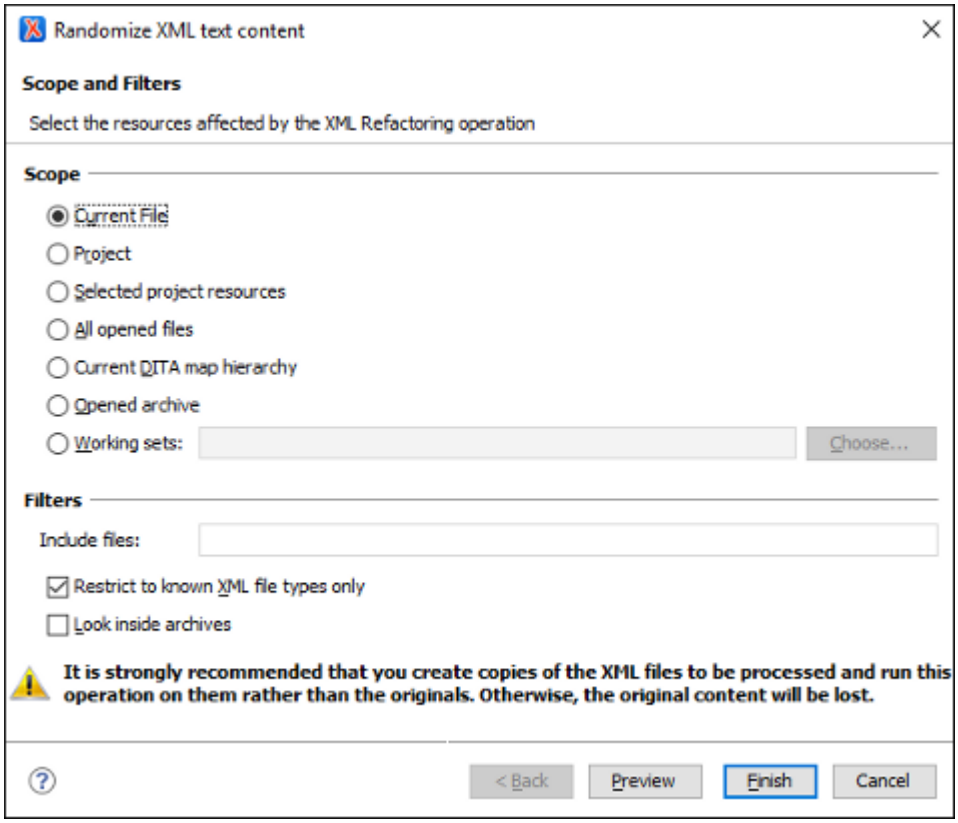
Randomize XML Text Content

Oxygen XML Developer 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.

Figure 581. Randomize XML Text Content Dialog Box



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 2304)* hierarchy for DITA projects, etc.) or you can define your own set of resources by creating a *working set (on page 2309)*.

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.

Performance Problems and Solutions

This section contains solutions for some common performance problems that may appear when running Oxygen XML Developer.

Related Information:
Documents with Long Lines *(on page 399)*
Loading Large Documents *(on page 397)*
External Tools *(on page 2197)*

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 267)*: **-Dsun.java2d.pmoffscreen=false**.

Out of Memory on External Processes

Problem

Oxygen XML Developer 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 79)*, go to **XML > PDF Output > FO Processors**, and increase the value of the **Memory available to the Apache FOP** option *(on page 193)*.
- 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 192)*
Custom Engines Preferences *(on page 191)*
External Tools Preferences *(on page 221)*
How to Enable Debugging for FO Processor Transformations *(on page 1177)*

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 Developer 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 Developer can have tens of threads), so using a very large value here can backfire and leave Oxygen XML Developer without memory.

Related Information:
Setting a Java Virtual Machine Parameter when Launching Oxygen XML Developer *(on page 267)*

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 Developer by setting the **-Xmx** parameter in a configuration file *(on page 267)* 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 Developer 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 Developer 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 Developer 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 Developer with Remote Desktop

Problem

When trying to run Oxygen XML Developer in a Remote Desktop Protocol (RDP) environment, the performance is slow and choppy.

Cause

Running a standalone version of Oxygen XML Developer over a slow RDP connection may result in performance issues.

Solution

As a workaround, you try to run Oxygen XML Developer 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 Developer 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 Developer installation folder (may need admin access).
2. Restart Oxygen XML Developer.
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 Developer, 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 78)*, 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 78)*, then re-install.

If you intentionally want to load extra Java libraries with Oxygen XML Developer, 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 Developer 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 Developer `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 Developer seems to take an abnormally long amount of time to start.

Cause 1

Some anti-virus software can cause Java applications, including Oxygen XML Developer, to start very slowly due to scanning compressed archives (such as the *JAR* libraries that all Java applications use). During the process, various files would be accessed by Oxygen XML Developer (including installed plugins) files from its installation folder, its user home folder (e.g. `C:\Users\<username>\AppData\Roaming\com.oxygenxml`), its project folder, or local files from the Git projects (if the Git Client is used).

Solution

A possible solution is to add the aforementioned folders to the list of exceptions in the anti-virus software settings.

Cause 2

If files/folders from the Oxygen XML Developer installation folder, its user home folder (e.g. `C:\Users\<username>\AppData\Roaming\com.oxygenxml`), its project folder, or local files from the Git projects (if the Git Client is used), project, Git projects) are located remotely, it sometimes slows down the process. For example, Oxygen XML Developer (or Java) may wait until these connections time out before proceeding.

Solution 2

Keep a copy of those files/folders locally, rather than using them from the remote location.

Cause 3

The built-in file browser may be slow to open if it is trying to access shortcuts to inaccessible network shares or drives on the Windows desktop.

Solution 3

Remove the shortcuts to inaccessible network shares or drives.

Archive Distribution Fails to Run on macOS 10.12 (Sierra)

Problem

For versions prior to 18.1, the classic archive distributions of Oxygen XML Developer (`.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 Developer was already installed when you upgraded to macOS 10.12 (Sierra), there are no problems.

Solution

If Oxygen XML Developer 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 Developer 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 Developer is located.
2. Run the following command:

   ```
   xattr -dr com.apple.quarantine oxygen/
   ```

   where "*oxygen*" is the actual name of the Oxygen XML Developer directory.

   If Oxygen XML Developer 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.

Blank Window is Shown When Starting the App Over an RDP Connection on Linux

Problem

When starting Oxygen XML Developer or its installer on *Linux*, a blank window is displayed when started over an RDP connection.

Cause

Oxygen XML Developer and its installer are Java Swing apps that require a 24 bit color depth from the X server.

Solution

1. If you are using *xrdp*, find the `/etc/xrdp/xrdp.ini` file.
2. Uncomment the `xserverbpp=24` line.

3. Save your files and close all the apps (the subsequent step will terminate your remote session so you could lose your progress if you do not save your files first).
4. Restart the *xrdp* service:

```
sudo systemctl restart xrdp.service
```



Note:
Alternatively, you can try setting `max_bpp=24` in the same `/etc/xrdp/xrdp.ini` file.

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:

```
<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 logging files are named `oxygen.log` and `oxygen#.log.qz` (for example, `oxygen.log`, `oxygen1.log.gz`, `oxygen2.log.gz`, etc.) and are located in the `Desktop\oxygenLog` folder.

Cannot Open Files from Desktop/Downloads/OneDrive on macOS

Problem

When using Oxygen XML Developer on *macOS*, the application cannot open files from Desktop/Downloads/OneDrive.

Cause

Sometimes, macOS shows a popup about allowing the application access to some special folders (e.g. Downloads or Desktop), and unless you explicitly agree, it will leave it unchecked (the app does not allowed access). The popup can go unnoticed and disappears after a while, so it is easy to overlook it and the application will not have access to that folder.

Solution

Go to the macOS **System preferences > Security & Privacey > Privacy tab > Files and Folders**. You should find *Oxygen* in that list and the folders you were prompted to access. Check the box for the folder (i.e. Downloads or Desktop), if there is one unchecked.

> **Note:**
> If that does not work, look at "Full Disk Access" from the same Privacy tab. Add *Oxygen* there so that it has full access. However, only use this method as a last resort.

Cannot Uninstall Oxygen XML Developer in Windows

Problem

When I try to uninstall Oxygen XML Developer 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 Developer 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 79)*, go to **Appearance > Fonts**, and set the value of the **Text antialiasing** option *(on page 89)* to ON.



Note:
If this workaround does not resolve the problem, set the **Text antialiasing** option *(on page 89)* to other values.

Crash at Startup on Windows with an Error About the nvoglv32.dll File

Problem

I try to start Oxygen XML Developer on Windows but it crashed with an error message about *"Fault Module Name: nvoglv32.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 Developer install folder (if you start Oxygen XML Developer 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 Developer (if you start Oxygen XML Developer with the `oxygen.bat` script). The home folder of the Java virtual machine that runs Oxygen XML Developer is the value of the `java.home` property that is available in the **System properties** tab of the **About** dialog box (**Help > About**).

Damaged File Associations on macOS

Problem

After upgrading macOS and Oxygen XML Developer, 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 Developer installations on your hard drive.
2. Delete them by dragging them to the Trash.
3. Clear the Trash.
4. Unpack the Oxygen XML Developer 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:

   ```
   /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:

   ```
   killall Finder
   ```

8. Create an XML or XSD file on your desktop. It should have the Oxygen XML Developer icon.
9. Double-click the file.
10. Accept the confirmation.

Result: When you start Oxygen XML Developer, 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 2202)* 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 Developer log files, follow these steps:

1. Create a text file called `logback.xml` in the application installation folder, with the following content:

```
<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.qz` (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 Developer 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 Developer 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 Developer 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 2304) 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 the **Proxy** preferences page *(on page 230)*

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:

```
<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 Developer *(on page 2206)*. 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 Developer 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)*.

Solution

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 1177)*

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 Developer, and look in it at the line specified in the error message `(See position 179:-1)`.

Related Information:
How to Enable Debugging for FO Processor Transformations *(on page 1177)*

DITA PDF CSS-based Processing Common Errors

For information about possible common errors that could be encountered when processing CSS-based DITA to PDF output, see the Troubleshooting section in the CSS-based PDF Customization guide *(on page 1706)*.

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.

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.

Fonts Installed in Windows Do Not Appear in Fonts Preferences Page

Problem

I installed a font in Windows and it does not appear in the list of available fonts in the **Preferences > Fonts** page *(on page 88)* (in the **Editor** section).

Cause

Oxygen XML Developer is a Java application and Java does not detect fonts that are installed at the user level. This most likely occurred because you installed the font via the **Install for me** option in Windows.

Solution

Reinstall the font using the **Install for all users** option in Windows. You will need Administrator privileges to access this option.

Format and Indent Fails

Problem

When I use the  **Format and Indent** function, I get an error message that indicates the *Format and Indent failed*.

Cause

This happens because the application tries to limit the exposure to XXE attacks so the parser blocks the expansion of system entities (the ones that are loaded from disk or from remote sources).

Solution

If you are in complete control of the XML documents (you manage or trust those who are creating and editing the documents), you can disable this particular check by selecting the **Enable system parameter entity expansion in other entity definitions** option *(on page 171)* in the **XML Parser** preferences page.

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 11)*.

> **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 Developer, 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 Developer 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 Developer.
2. Open the `%APPDATA%\com.oxygenxml` folder and look for a file called something like **HunspellCrashGuard*.txt**. Delete that file.
3. Restart Oxygen XML Developer.

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

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, Oxygen XML Developer does not automatically scale high-resolution images when using the system's scaling settings.

Cause

This happens because Java 11 does not detect the system scaling setting for HiDPI displays on Linux operating system.

Solution

In the Oxygen XML Developer 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

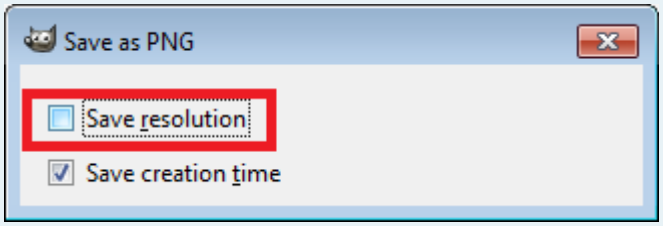
PDF output from XML content is obtained by first obtaining an intermediary XML format called XSL-FO and then applying an XSL-FO processor to it to obtain the PDF. This stretching problem is caused by the fact that all XSL-FO processors take into account the DPI (dots-per-inch) resolution when computing 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.



This allows you to control the image resolution from the configuration file for all referenced images.

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)*.

Java Virtual Machine (JVM) Crash on macOS

Problem

Oxygen XML Developer 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*.

Keyboard Language Resets to Default on Windows

Problem

In Windows, I have set a specific language for my keyboard and while using Oxygen XML Developer, 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 Developer 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, 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 223)* 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 223)* 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 223)* 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 Developer.

Causes

By default, the Touch Bar function keys are not enabled for Oxygen XML Developer.

Solution

To enable the Touch Bar function keys for Oxygen XML Developer, 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**.

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** 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.

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 Developer, 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 134)* 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 Developer *(on page 2206)*.

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 126)*. 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 2007)*.

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 2304)* 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 Developer or using the `transform` script.

- 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 Developer 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 Developer 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**).
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 Developer.

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 Developer). 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 Developer 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 88)*. If it is a text file, you must also set the encoding used for non XML files *(on page 123)*. 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 2304)* 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.
4. Edit `frameset.html` file.
5. Locate element `<frame name="contentwin" src="concepts/about.html">`.
6. Replace `"concepts/about.html"` with `"index.html"`.

Text on Buttons and Labels is Invisible for Linux Installer

Problem

After starting the Linux installer, the text on buttons and labels is invisible.

Cause

This seems to be a font issue between Oracle Java SE 8 (bundled with the installer) and Fedora/Gnome.

Solution

There are two possible workarounds:

- Run the installer with the default (non-native) Java L&F by using the `-Dinstall4j.nolaf=true` argument. For example:

  ```
  oxygen-64bit-openjdk.sh -Dinstall4j.nolaf=true
  ```

- Run the installer in console mode using the `-c` argument. For example:

  ```
  oxygen-64bit-openjdk.sh -c
  ```

XML Document Takes a Long Time to Open

Problem

Oxygen XML Developer 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 137)* (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 134)* (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 Developer. *(on page 2206)*

XSLT Debugger Is Very Slow

Problem

When I run a transformation in the **XSLT Debugger** *perspective (on page 2307)*, 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 188)* during the transformation process. To view the XHTML output result do one of the following:

- Run the transformation in the **Editor** *perspective (on page 2307)* and make sure the **Open in Browser/ System Application** option *(on page 1112)* is selected.
- Run the transformation in the **XSLT Debugger** *perspective (on page 2307)*, save the text output area to a file, and use a browser application for viewing it (for example, Firefox or Chrome).

21.

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 Developer 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 Developer 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 Developer

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 36)* like this:

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

Screen Reader Software (Windows OS)

If you are using a text-to-speech narrator, Oxygen XML Developer supports this since it is a Java application and it is periodically tested on Windows using both the NVDA and JAWS screen readers on the Windows operating system.

Using the JAWS Screen Reader (Windows)

The JAWS (Job Access With Speech) 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 Developer: http://docs.oracle.com/javase/7/docs/technotes/guides/access/enable_and_test.html.

To enable the Java access bridge:

1. Since Oxygen XML Developer 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 Developer installation directory (for example, in Windows, it would be something like this:

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

2. Then run the following command:

```
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 Developer starts, the narrator will start reading content from Oxygen XML Developer 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 (Windows)

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 Developer: http://docs.oracle.com/javase/7/docs/technotes/guides/access/enable_and_test.html.

To enable the Java access bridge:

1. Since Oxygen XML Developer 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 Developer installation directory (for example, in Windows, it would be something like this:

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

2. Then run the following command:

```
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 Developer starts, the narrator will start reading content from Oxygen XML Developer 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 Developer 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 Developer.

Besides the main editing area, Oxygen XML Developer 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 Developer.
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 Developer 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 Developer **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 2303)*. 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.

Screen Reader Software (macOS)

On **macOS**, the application can be used with **VoiceOver** but is not rigorously tested with it. Because of processing limitations of the **VoiceOver** application, various limitations may be encountered. To avoid blocking the application, when tree-structure user interface controls are used (e.g. the **Project** or **Outline** view), if a node in the tree contains more than 500 child nodes, only the first 500 child nodes are accessible to the screen reader.

Oxygen XML Developer VPAT Accessibility Conformance Report

International Edition

VPAT® Version 2.5Rev – April 2025

Name of Product/Version

Oxygen XML Developer 28.1

Product Description

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

Report Date

March 2026

Contact Information

support@oxygenxml.com

Notes

Oxygen XML Developer 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 Accessibility *(on page 2233)*.

Evaluation Methods Used:

Accessibility evaluation was conducted using:

- Manual keyboard testing
- Screen reader testing using NVDA and JAWS assistive technologies
- Internal QA testing

Applicable Standards/Guidelines

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

Standard/Guideline	Included In Report
Web Content Accessibility Guidelines 2.0	Level A - Yes Level AA - Yes Level AAA - No
Web Content Accessibility Guidelines 2.1	Level A - Yes Level AA - Yes Level AAA - No
Web Content Accessibility Guidelines 2.2	Level A - Yes Level AA - Yes Level AAA - No
Revised Section 508 standards published January 18, 2017 and corrected January 22, 2018	Yes

Standard/Guideline	Included In Report
EN 301 549 Accessibility requirements suitable for public procurement of ICT products and services in Europe, - V3.1.1 (2019-11)	No

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
- Chapter 6 – 602.3 Electronic Support Documentation

Note:
This report follows the WCAG 2.2 Conformance Requirements; the conformance model is that of WCAG 2.x; target conformance: Levels A and AA.

Table 1: Success Criteria, Level A

Criteria	Conformance Level	Remarks and Explanations
1.1.1 Non-text Content (Level A) Also applies to: Revised Section 508 • 501 (Web)(Software) • 504.2 (Authoring Tool) • 602.3 (Support Docs)	Supports	Text alternatives are provided for non-text content.
1.2.1 Audio-only and Video-only (Prerecorded) (Level A)	Not Applicable	The product does not play audio and video content to end users.

Criteria	Conformance Level	Remarks and Explanations
Also applies to: Revised Section 508 • 501 (Web)(Software) • 504.2 (Authoring Tool) • 602.3 (Support Docs)		
1.2.2 Captions (Prerecorded) (Level A) Also applies to: Revised Section 508 • 501 (Web)(Software) • 504.2 (Authoring Tool) • 602.3 (Support Docs)	Not Applicable	The product does not provide pre-recorded media that requires captions.
1.2.3 Audio Description or Media Alternative (Prerecorded) (Level A) Also applies to: Revised Section 508 • 501 (Web)(Software) • 504.2 (Authoring Tool) • 602.3 (Support Docs)	Not Applicable	The product does not provide pre-recorded media that requires alternate descriptions.
1.3.1 Info and Relationships (Level A) Also applies to: Revised Section 508 • 501 (Web)(Software) • 504.2 (Authoring Tool) • 602.3 (Support Docs)	Partially Supports	Information, structure, and relationships conveyed through presentation can be programmatically determined or are available in text, with exceptions that include: • **Grid** editing mode. • Schema **Design** editing mode. • The editing of profiling attributes using the **Edit profiling at-**

Criteria	Conformance Level	Remarks and Explanations
		tributes and **Insert/Edit Topic Reference** dialog boxes. • The editing of a DITA map in the **Author** editing mode.
1.3.2 Meaningful Sequence (Level A) Also applies to: Revised Section 508 • 501 (Web)(Software) • 504.2 (Authoring Tool) • 602.3 (Support Docs)	Supports	The product presents content in a meaningful sequence. Authors should use Unicode right-to-left mark (RLM) or left-to-right mark (LRM) to mix text direction inline.
1.3.3 Sensory Characteristics (Level A) Also applies to: Revised Section 508 • 501 (Web)(Software) • 504.2 (Authoring Tool) • 602.3 (Support Docs)	Supports	The product provides textual identification for understanding and operating content.
1.4.1 Use of Color (Level A) Also applies to: Revised Section 508 • 501 (Web)(Software) • 504.2 (Authoring Tool) • 602.3 (Support Docs)	Supports	Color is not used as the only visual means of conveying information, indicating an action, prompting a response, or distinguishing a visual element.
1.4.2 Audio Control (Level A) Also applies to: Revised Section 508	Not Applicable	There is no sound that plays automatically by default.

Criteria	Conformance Level	Remarks and Explanations
• 501 (Web)(Software) • 504.2 (Authoring Tool) • 602.3 (Support Docs)		
2.1.1 Keyboard (Level A) Also applies to: Revised Section 508 • 501 (Web)(Software) • 504.2 (Authoring Tool) • 602.3 (Support Docs)	Partially Supports	Most of the content is operable through a keyboard interface, with exceptions that include: • Some toolbars in the application not being accessible via a keyboard. • Combo boxes and buttons located inside the table from the dialog box used to create or edit a validation scenario.
2.1.2 No Keyboard Trap (Level A) Also applies to: Revised Section 508 • 501 (Web)(Software) • 504.2 (Authoring Tool) • 602.3 (Support Docs)	Partially Supports	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.
2.1.4 Character Key Shortcuts (Level A 2.1 only) Also applies to: Revised Section 508 – Does not apply	Not Applicable	The product does not include character key shortcuts.
2.2.1 Timing Adjustable (Level A) Also applies to: Revised Section 508	Not Applicable	The product does not include time limits.

Criteria	Conformance Level	Remarks and Explanations
• 501 (Web)(Software) • 504.2 (Authoring Tool) • 602.3 (Support Docs)		
2.2.2 Pause, Stop, Hide (Level A) Also applies to: Revised Section 508 • 501 (Web)(Software) • 504.2 (Authoring Tool) • 602.3 (Support Docs)	Supports	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.
2.3.1 Three Flashes or Below Threshold (Level A) Also applies to: Revised Section 508 • 501 (Web)(Software) • 504.2 (Authoring Tool) • 602.3 (Support Docs)	Supports	The product does not have content that flashes more than three times in any one second.
2.4.1 Bypass Blocks (Level A) 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	Not Applicable	The application does not contain blocks of repeated content.
2.4.2 Page Titled (Level A) Also applies to: Revised Section 508	Supports	Each side view and dialog box in the application has a title.

Criteria	Conformance Level	Remarks and Explanations
• 501 (Web)(Software) • 504.2 (Authoring Tool) • 602.3 (Support Docs)		
2.4.3 Focus Order (Level A) Also applies to: Revised Section 508 • 501 (Web)(Software) • 504.2 (Authoring Tool) • 602.3 (Support Docs)	Supports	Focusable components receive focus in an order that preserves meaning and operability.
2.4.4 Link Purpose (In Context) (Level A) Also applies to: Revised Section 508 • 501 (Web)(Software) • 504.2 (Authoring Tool) • 602.3 (Support Docs)	Supports	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.
2.5.1 Pointer Gestures (Level A 2.1 and 2.2) Also applies to: Revised Section 508 – Does not apply	Not Applicable	The product does not have functionality that requires multi-point or path-based gestures.
2.5.2 Pointer Cancellation (Level A 2.1 and 2.2) Also applies to: Revised Section 508 – Does not apply	Partially Supports	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.
2.5.3 Label in Name (Level A 2.1 and 2.2) Also applies to:	Supports	The names of the user interface components contain the text that is presented visually.

Criteria	Conformance Level	Remarks and Explanations
Revised Section 508 – Does not apply		
2.5.4 Motion Actuation (Level A 2.1 and 2.2) Also applies to: Revised Section 508 – Does not apply	Not Applicable	The product does not contain functionality that can be operated by device or user motion.
3.1.1 Language of Page (Level A) Also applies to: Revised Section 508 • 501 (Web)(Software) • 504.2 (Authoring Tool) • 602.3 (Support Docs)	Does Not Support	The product does not report the default language for each open document.
3.2.1 On Focus (Level A) Also applies to: Revised Section 508 • 501 (Web)(Software) • 504.2 (Authoring Tool) • 602.3 (Support Docs)	Supports	No changes of context occur when any component receives focus.
3.2.2 On Input (Level A) Also applies to: Revised Section 508 • 501 (Web)(Software) • 504.2 (Authoring Tool) • 602.3 (Support Docs)	Supports	Changing the setting of any user interface component does not automatically cause a change of context.
3.2.6 Consistent Help (Level A 2.2 only) Revised Section 508 – Does not apply	Supports	Help is consistently available through the Help menu in the application menu bar and through a Help button located in the bottom-left corner of dialogs

Criteria	Conformance Level	Remarks and Explanations
		and preference pages. Additional contextual help can be accessed via the F1 key across the application.
3.3.1 Error Identification (Level A) Also applies to: Revised Section 508 • 501 (Web)(Software) • 504.2 (Authoring Tool) • 602.3 (Support Docs)	Supports	Product shows error messages when user input is invalid.
3.3.2 Labels or Instructions (Level A) Also applies to: Revised Section 508 • 501 (Web)(Software) • 504.2 (Authoring Tool) • 602.3 (Support Docs)	Partially Supports	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.
3.3.7 Redundant Entry (Level A 2.2 only) Revised Section 508 – Does not apply	Supports	The application does not require users to re-enter information within multi-step processes. Data entered in wizard dialogs is retained when navigating between steps and is not requested again during the same workflow.
4.1.1 Parsing (Level A) Also applies to: Revised Section 508 • 501 (Web)(Software) • 504.2 (Authoring Tool) • 602.3 (Support Docs)	Supports	All labels presenting HTML content in the product have valid HTML content.

Criteria	Conformance Level	Remarks and Explanations
4.1.2 Name, Role, Value (Level A) Also applies to: Revised Section 508 • 501 (Web)(Software) • 504.2 (Authoring Tool) • 602.3 (Support Docs)	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

Criteria	Conformance Level	Remarks and Explanations
1.2.4 Captions (Live) (Level AA) Also applies to: Revised Section 508 • 501 (Web)(Software) • 504.2 (Authoring Tool) • 602.3 (Support Docs)	Not Applicable	The product does not contain live audio content.
1.2.5 Audio Description (Prerecorded) (Level AA) Also applies to: Revised Section 508 • 501 (Web)(Software) • 504.2 (Authoring Tool) • 602.3 (Support Docs)	Not Applicable	The product does not provide prerecorded video content that requires audio description.
1.3.4 Orientation (Level AA 2.1 and 2.2) Also applies to: Revised Section 508 – Does not apply	Supports	Content does not restrict its view and operation to a single display orientation.

Criteria	Conformance Level	Remarks and Explanations
1.3.5 Identify Input Purpose (Level AA 2.1 and 2.2) Also applies to: Revised Section 508 – Does not apply	Not Applicable	The content does not contain input fields that collect information about the user.
1.4.3 Contrast (Minimum) (Level AA) Also applies to: Revised Section 508 • 501 (Web)(Software) • 504.2 (Authoring Tool) • 602.3 (Support Docs)	Partially Supports	The product has sufficient contrast between the foreground and background colors, with few exceptions, including: • Change tracking content in the document for some of the author colors. • Change tracking content in some review panel items, when the item is selected. • Placeholders shown in empty elements. • Comments marked as done.
1.4.4 Resize text (Level AA) Also applies to: Revised Section 508 • 501 (Web)(Software) • 504.2 (Authoring Tool) • 602.3 (Support Docs)	Partially Supports	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. Sizes of certain text components cannot be increased.
1.4.5 Images of Text (Level AA) Also applies to: Revised Section 508	Supports	The few buttons with icons containing text characters also provide text alternatives.

Criteria	Conformance Level	Remarks and Explanations
• 501 (Web)(Software) • 504.2 (Authoring Tool) • 602.3 (Support Docs)		
1.4.10 Reflow (Level AA 2.1 and 2.2) Also applies to: Revised Section 508 – Does not apply	Partially Supports	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.
1.4.11 Non-text Contrast (Level AA 2.1 and 2.2) Also applies to: Revised Section 508 – Does not apply	Supports	User interface components and states have sufficient contrast against adjacent colors.
1.4.12 Text Spacing (Level AA 2.1 and 2.2) Also applies to: Revised Section 508 – Does not apply	Supports	There is no loss of content or functionality occurs by setting line height (line spacing), spacing following paragraphs, letter spacing, and word spacing.
1.4.13 Content on Hover or Focus (Level AA 2.1 and 2.2) Also applies to: Revised Section 508 – Does not apply	Partially Supports	The tooltips for most user interface simple components (buttons) are not hoverable and persistent.
2.4.5 Multiple Ways (Level AA) Also applies to: Revised Section 508	Supports	There are multiple ways to navigate between the open documents.

Criteria	Conformance Level	Remarks and Explanations
• 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		
2.4.6 Headings and Labels (Level AA) Also applies to: Revised Section 508 • 501 (Web)(Software) • 504.2 (Authoring Tool) • 602.3 (Support Docs)	Supports	Headings and labels describe the topic or purpose.
2.4.7 Focus Visible (Level AA) Also applies to: Revised Section 508 • 501 (Web)(Software) • 504.2 (Authoring Tool) • 602.3 (Support Docs)	Partially Supports	The product has a visible indication of focus for almost all user interface controls (buttons, text fields, combo boxes, etc.) Exceptions include: • Combo boxes located in the **DITA Maps Manager** and **SharePoint Browser** view. • Filtering buttons located in the **Templates** tab from the **New/Edit scenario** and the **DITA Reusable Components** view. • Tab buttons located in the **Insert/Edit Topic Reference** dialog box. • Toggle buttons in their active state.
2.4.11 Focus Not Obscured (Minimum) (Level AA 2.2 only) Revised Section 508 – Does not apply	Partially Supports	In most areas of the product, the element receiving keyboard focus remains visible and is not obscured by other interface components when navigating using the keyboard.

Criteria	Conformance Level	Remarks and Explanations
		An exception occurs in the **Outline** view. When navigating deeply nested items using the keyboard, the horizontal scroll position does not always automatically adjust to reveal the focused element. As a result, the focused item may be partially or fully outside the visible viewport of the panel.
2.5.7 Dragging Movements (Level AA 2.2 only) Revised Section 508 – Does not apply	Partially Supports	The application provides drag-and-drop functionality for tasks such as reorganizing files and elements. These actions also have alternative methods available through menu commands and keyboard actions. However, moving and resizing side views rely on drag interactions and do not currently provide alternative mechanisms that do not require dragging. A "Reset Layout" action is available to restore the default panel configuration.
2.5.8 Target Size (Minimum)(Level AA 2.2 only) Revised Section 508 – Does not apply	Partially Supports	Most interactive controls meet the minimum 24 × 24 pixel target size. Some compact toolbar controls, such as split buttons (e.g., "Validate and Check for Completeness" in DITA Maps Manager view), include smaller dropdown activation areas due to dense toolbar layouts. Equivalent functionality is available through menus.
3.1.2 Language of Parts (Level AA) Also applies to: Revised Section 508	Does Not Support	The language of parts is not specified.

Criteria	Conformance Level	Remarks and Explanations
• 501 (Web)(Software) • 504.2 (Authoring Tool) • 602.3 (Support Docs)		
3.2.3 Consistent Navigation (Level AA) 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	Supports	The product has a consistent navigation mechanism.
3.2.4 Consistent Identification (Level AA) 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	Supports	The product components are identified consistently.
3.3.3 Error Suggestion (Level AA) Also applies to: Revised Section 508 • 501 (Web)(Software) • 504.2 (Authoring Tool) • 602.3 (Support Docs)	Supports	The product provides suggestions for the input error if there are any available.
3.3.4 Error Prevention (Legal, Financial, Data) (Level AA)	Not Applicable	The product does not process legal commitments or financial transactions or modify user-controllable data.

Criteria	Conformance Level	Remarks and Explanations
Also applies to: Revised Section 508 • 501 (Web)(Software) • 504.2 (Authoring Tool) • 602.3 (Support Docs)		
3.3.8 Accessible Authentication (Minimum) (Level AA 2.2 only) Revised Section 508 – Does not apply	Supports	Where authentication is required, password fields allow users to paste credentials. The authentication process does not rely on cognitive function tests such as CAPTCHA or puzzles.
4.1.3 Status Messages(Level AA 2.1 2.2) Also applies to: Revised Section 508 – Does not apply	Not Applicable	The product does not contain status messages as defined by this criterion.

Table 3: Success Criteria, Level AAA

Criteria	Conformance Level	Remarks and Explanations
1.2.6 Sign Language (Prerecorded) (Level AAA) Revised Section 508 – Does not apply	Not Evaluated	
1.2.7 Extended Audio Description (Prerecorded) (Level AAA) Revised Section 508 – Does not apply	Not Evaluated	
1.2.8 Media Alternative (Prerecorded) (Level AAA) Revised Section 508 – Does not apply	Not Evaluated	
1.2.9 Audio-only (Live) (Level AAA)	Not Evaluated	

Criteria	Conformance Level	Remarks and Explanations
Revised Section 508 – Does not apply		
1.3.6 Identify Purpose (Level AAA 2.1 and 2.2) Revised Section 508 – Does not apply	Not Evaluated	
1.4.6 Contrast Enhanced (Level AAA) Revised Section 508 – Does not apply	Not Evaluated	
1.4.7 Low or No Background Audio (Level AAA) Revised Section 508 – Does not apply	Not Evaluated	
1.4.8 Visual Presentation (Level AAA) Revised Section 508 – Does not apply	Not Evaluated	
1.4.9 Images of Text (No Exception) Control (Level AAA) Revised Section 508 – Does not apply	Not Evaluated	
2.1.3 Keyboard (No Exception) (Level AAA) Revised Section 508 – Does not apply	Not Evaluated	
2.2.3 No Timing (Level AAA) Revised Section 508 – Does not apply	Not Evaluated	
2.2.4 Interruptions (Level AAA) Revised Section 508 – Does not apply	Not Evaluated	
2.2.5 Re-authenticating (Level AAA) Revised Section 508 – Does not apply	Not Evaluated	

Criteria	Conformance Level	Remarks and Explanations
2.2.6 Timeouts (Level AAA 2.1 and 2.2) Revised Section 508 – Does not apply	Not Evaluated	
2.3.2 Three Flashes (Level AAA) Revised Section 508 – Does not apply	Not Evaluated	
2.3.3 Animation from Interactions (Level AAA 2.1 and 2.2) Revised Section 508 – Does not apply	Not Evaluated	
2.4.8 Location (Level AAA) Revised Section 508 – Does not apply	Not Evaluated	
2.4.9 Link Purpose (Link Only) (Level AAA) Revised Section 508 – Does not apply	Not Evaluated	
2.4.10 Section Headings (Level AAA) Revised Section 508 – Does not apply	Not Evaluated	
2.4.12 Focus Not Obscured (Enhanced) (Level AAA 2.2 only) Revised Section 508 – Does not apply	Not Evaluated	
2.4.13 Focus Appearance (Level AAA 2.2 only) Revised Section 508 – Does not apply	Not Evaluated	
2.5.5 Target Size (Level AAA 2.1 and 2.2) Revised Section 508 – Does not apply	Not Evaluated	

Criteria	Conformance Level	Remarks and Explanations
2.5.6 Concurrent Input Mechanisms (Level AAA 2.1 and 2.2) Revised Section 508 – Does not apply	Not Evaluated	
3.1.3 Unusual Words (Level AAA) Revised Section 508 – Does not apply	Not Evaluated	
3.1.4 Abbreviations (Level AAA) Revised Section 508 – Does not apply	Not Evaluated	
3.1.5 Reading Level (Level AAA) Revised Section 508 – Does not apply	Not Evaluated	
3.1.6 Pronunciation (Level AAA) Revised Section 508 – Does not apply	Not Evaluated	
3.2.5 Change on Request (Level AAA) Revised Section 508 – Does not apply	Not Evaluated	
3.3.5 Help (Level AAA) Revised Section 508 – Does not apply	Not Evaluated	
3.3.6 Error Prevention (All) (Level AAA) Revised Section 508 – Does not apply	Not Evaluated	
3.3.9 Accessible Authentication (Enhanced) (Level AAA 2.2 only) Revised Section 508 – Does not apply	Not Evaluated	

Revised Section 508 Report

N/A

Chapter 3: Functional Performance Criteria (FPC)

Criteria	Conformance Level	Remarks and Explanations
302.1 Without Vision	Partially Supports	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.
302.2 With Limited Vision	Supports	The product is operable with limited vision.
302.3 Without Perception of Color	Supports	Color is not used as the only visual means of conveying information, indicating an action, prompting a response, or distinguishing a visual element.
302.4 Without Hearing	Supports	The product does not require hearing for use.
302.5 With Limited Hearing	Supports	The product does not require hearing for use.
302.6 Without Speech	Supports	The product does not require speech for use.
302.7 With Limited Manipulation	Partially Supports	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.
302.8 With Limited Reach and Strength	Supports	The product is functional with limited reach and limited strength. It supports operating system tools such as *StickyKeys* and *FilterKeys*.
302.9 With Limited Language, Cognitive, and Learning Abilities	Partially Supports	The product is operable by users with limited language, cognitive, and learning abilities. To accommodate users with limited cognition, the UI provides

Criteria	Conformance Level	Remarks and Explanations
		icons, text, or a combination of both, for controls. It provides the ability to change the interface language to 6 languages.

Chapter 4: Hardware

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

Chapter 5: Software

501 General

Criteria	Conformance Level	Remarks and Explanations
501.1 Scope – Incorporation of WCAG 2.0 AA	See WCAG 2.x section *(on page 2238)*	See information in WCAG section

502 Interoperability with Assistive Technology

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

502.3 Accessibility Services

Criteria	Conformance Level	Remarks and Explanations
502.3.1 Object Information	Partially Supports	Most of the object information can be programmatically determined. As noted in **1.3.1 Info and Relationships**, the content from the **Grid** and schema **Design** editing modes is not exposed programmatically.
502.3.2 Modification of Object Information	Partially Supports	States and properties that can be set by the user can be set programmati-

Criteria	Conformance Level	Remarks and Explanations
		cally. As noted in **1.3.1 Info and Relationships**, there are few exceptions.
502.3.3 Row, Column, and Headers	Partially Supports	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.
502.3.4 Values	Partially Supports	The current values of an object can be programmatically determined. As noted in **1.3.1 Info and Relationships**, there are few exceptions. The content from the **Grid** and schema **Design** editing modes is not exposed programmatically.
502.3.5 Modification of Values	Partially Supports	Values that can be set by the user are capable of being set programmatically. As noted in **1.3.1 Info and Relationships**, there are few exceptions. The content from the **Grid** and schema **Design** editing modes cannot be set programmatically.
502.3.6 Label Relationships	Partially Supports	Information, structure, and relationships conveyed through presentation can be programmatically determined or are available in text. As noted in **1.3.1 Info and Relationships**, there are few exceptions.
502.3.7 Hierarchical Relationships	Partially Supports	The hierarchical relationships are, in general, accessible programmatically, but as noted in **1.3.1 Info and Relationships**, there are few exceptions.
502.3.8 Text	Partially Supports	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**

Criteria	Conformance Level	Remarks and Explanations
		Info and Relationships, there are few exceptions, including the text from the **Grid** and schema **Design** modes.
502.3.9 Modification of Text	Partially Supports	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.
502.3.10 List of Actions	Supports	Actions that can be executed on an object can be determined programmatically from the context menu or content completion menu.
502.3.11 Actions on Objects	Supports	Actions on objects can performed by users, including those using assistive technologies.
502.3.12 Focus Cursor	Partially Supports	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.
502.3.13 Modification of Focus Cursor	Partially Supports	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.
502.3.14 Event Notification	Supports	The changes in states and other properties are communicated through notifications that are communicated to assistive technology.

Criteria	Conformance Level	Remarks and Explanations
502.4 Platform Accessibility Features	Not Applicable	This product is not platform software.

503 Applications

Criteria	Conformance Level	Remarks and Explanations
503.2 User Preferences	Partially Supports	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.
503.3 Alternative User Interfaces	Not Applicable	The application does not provide an alternative user interface that functions as assistive technology.

503.4 User Controls for Captions and Audio Description

Criteria	Conformance Level	Remarks and Explanations
503.4.1 Caption Controls	Not Applicable	The product does not provide controls for volume adjustment.
503.4.2 Audio Description Controls	Not Applicable	The product does not provide controls for program selection.

504 Authoring Tools

Criteria	Conformance Level	Remarks and Explanations
504.2 Content Creation or Editing (if not authoring tool, enter “not applicable”)	See the WCAG 2.x section *(on page 2238)*	See information in WCAG section

Criteria	Conformance Level	Remarks and Explanations
504.2.1 Preservation of Information Provided for Accessibility in Format Conversion	Partially Supports	For the main XML vocabulary supported in the application (DITA), the accessibility information is preserved in the generated main formats (WebHelp, PDF).
504.2.2 PDF Export	Supports	The product is capable of publishing PDF files that conform to PDF/UA-1.
504.3 Prompts	Partially Supports	For DITA documents, there is an optional Schematron that performs accessibility checks on the content and prompts the authors whenever it detects errors.
504.4 Templates	Does Not Support	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.

Chapter 6: Support Documentation and Services

601.1 Scope

602 Support Documentation

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

603 Support Services

Criteria	Conformance Level	Remarks and Explanations
603.2 Information on Accessibility and Compatibility Features	Supports	The support services cover the accessibility features.
603.3 Accommodation of Communication Needs	Supports	Support is provided over a variety of channels including email and phone.

Legal Disclaimer

This report describes **Oxygen XML Developer** 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*.

© 2026 Syncro Soft SRL. All rights reserved.

22.
Scripting Oxygen

Although Oxygen XML Developer 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 Developer. To run the script, you are required to purchase a special scripting commercial license.

The **Validate and Check For Completeness** 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 (found in the `scripts` subfolder inside **Oxygen's** installation directory) 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/validate-check-completion.yml. The GitHub action calls a Gradle script target named `runValidation`: https://github.com/oxygenxml/blog/blob/master/build/build.gradle.

Installing DITA OT Specialization Plugins

If you are using specialized DITA XML content, you must install extra plugins in the `DITA-OT` publishing engine bundled with the all-platforms distribution of Oxygen XML Developer. To do that, you can copy your

DITA-OT specialization plugins to the `OXYGEN_ALL_PLATFORMS_FOLDER/frameworks/dita/DITA-OT/plugins` folder, then open a command line console in the `DITA-OT/bin` folder, and execute the `dita --install` command to install the extra plugins in the DITA OT.

Using Schematron Rules For Validation

In the application's **DITA Maps Manager** view, you can configure the **Validate and Check For Completeness** dialog box's settings to reference a custom Schematron schema. The settings dialog box has an **Export settings** option that can be used to export the settings to an XML configuration file. Once you have this configuration file, you can provide it as a parameter in the validation script as exemplified above.

Transform

Attention:

- This script is bundled with the all platforms distribution of Oxygen XML Developer. 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 a 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 a Oxygen Publishing Engine license.

The **Transform** script (`transform.sh`, found in the `scripts` subfolder inside **Oxygen's** installation directory) helps you to execute a transformation scenario. You can run the scenarios for the existing document types (frameworks) *(on page 2305)* 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 Developer 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 2305)*.

-v

This argument can be specified to activate verbose logging for DITA-OT and ANT scenarios. It is useful for debugging.

Tip:
For a GitHub use case sample of this script, see Oxygen Transformation Template.

Validate

Attention:
This script is bundled with the all platforms distribution of Oxygen XML Developer. To run the script, you are required to purchase a special scripting commercial license.

All the validations possible in Oxygen XML Developer can also be performed from scripting. This includes validating various types of XML schemas, such as XSD, RNG, RNC, DTD, and NVDL, as well as JSON Schema, XProc, ANT, XSLT, XQuery, CSS, LESS, HTML, WSDL, OpenAPI, and of course, XML with XML schema, and JSON/YAML with JSON Schema.

The **Validate** script (`validate.sh`, found in the `scripts` subfolder inside **Oxygen's** installation directory) can be used to validate a file or a directory and get the validation results in various formats.

Arguments for the Transform Script

```
sh scripts/validate.sh fileOrDirPath [-s schemaFilePath | -sn scenarioName] [-sf scenariosFilePath] [-if includeFilesFilter]
 [-ef excludeFilesFilter] [-ed excludeSubdirsFilter] [-rf reportFile] [-rft reportFormat] [-v] [-help | --help | -h | --h]
```

fileOrDirPath

Mandatory argument that specifies the path of the file or directory to validate (it can also be provided as a URL, but if you are validating directories, the only protocol considered is `'file://'`).

-s schemaFilePath

Optional argument that specifies the file path of the schema to validate against (it can also be provided as a URL).

-sn scenarioName

Optional argument that specifies the name of the validation scenario to be applied.

-sf scenariosFilePath

Optional argument that specifies the path of the file that stores the validation scenarios (either an **Oxygen** scenarios file or an **Oxygen** project file). It can also be provided as a URL.

Notes:

- The file that stores the validation scenarios must have a similar format to that generated from **Oxygen** by invoking **Export Global Validation Scenarios** from the **Options** menu. This type of **Oxygen**-generated scenarios files has a `.scenarios` file extension by default and contains all the necessary information about custom validation scenarios created in **Oxygen**.
- **Oxygen** also saves the custom validation scenarios (as well as the scenario associations made explicitly for the files you work with) in special formatted **Oxygen** project files (usually with the `.xpr` file extension). Therefore, by using the arguments provided through **-sn** and **-sf** options, you can apply any scenario that was previously stored in either a scenarios file or an **Oxygen** project file.
- The **-s** and **-sn** options are mutually exclusive. Specifying both in the same command line is not allowed.

-if includeFilesFilter

Use this argument to only validate the files that match the specified pattern (e.g. `.xml,.json`). The default value is `*`.

-ef excludeFilesFilter

Excludes the files that match the specified pattern (e.g. `test.wsdl,draft.xsl`) from the validation.

-ed excludeSubdirsFilter

Excludes the sub-directories that match the specified pattern (e.g. `.svn,_svn,.git`).

-rf reportFile

Specifies the path for the report file to save the validation results, instead of presenting them in the console. The content of the report file is formatted according to the **-rft** argument. The report file path can also be provided as a URL.

-rft reportFormat

Specifies the format of the validation report. Possible values: `txt`, `text`, `xml`, `json`, `html`, `htm`. Default values: `txt`, `text`.

-v

Prints additional information to the console (Verbose mode).

-help | --help | -h | --h

Displays help text.

Additional Notes:

- Avoid activating the Verbose mode (**-v** option) when opting to redirect the console (and the validation report implicitly) to a specific file. That is done using the **>** operator instead of the **-rf** option. The additional information provided through verbose mode is also saved to the report file, making it to be reported as invalid when inspected in specialized editors. However, that information is placed at the beginning of the report, as plain text. If removed, the report should become valid.
- If the validation uses the **Saxon** engine and you do not have a commercial license, then the script automatically uses the **Saxon Home Edtion** distribution that does not require a license. However, if the validation involves specific **Saxon Personal / Enterprise Edition** advanced features, then the validation report clearly signals that an appropriate **Saxon** license was not found. Placing a valid **Saxon** license file in the `lib` directory from the **Oxygen** installation folder solves the problem and the validation operation works as expected.

Examples of the Validate Script

Example 1: Validate a File by Applying a Custom Validation Scenario

```
sh scripts/validate.sh "workspace/xmlFolder/xmlFile.xml" -sn "xmlValScn"
 -sf "workspace/scn/valScn.scenarios"
```

This command implies validating `xmlFile.xml` by applying the validation scenario named **xmlValScn**, described in the `valScn.scenarios` file. If you want to apply more than one validation scenario, you can use the `-sn scenarioName` construct multiple times.

Example 2: Validate a Directory by Applying an Oxygen Default Validation Scenario

```
sh scripts/validate.sh "workspace/DITAFolder" -sn "DITA"
```

A scenario name is provided, but without specifying a scenarios file. This command implies validating all files from `DITAFolder` by applying the **Oxygen** default validation scenario named **DITA** (in accordance with the association made in the **Document Type Configuration Dialog Box** *(on page 95)*).

Example 3: Validate a File by Applying Associated Scenarios Stored in an Oxygen Project File

```
sh scripts/validate.sh "workspace/mainFolder/main.xml" -sf "worksapce/proj/proj-1.xpr"
```

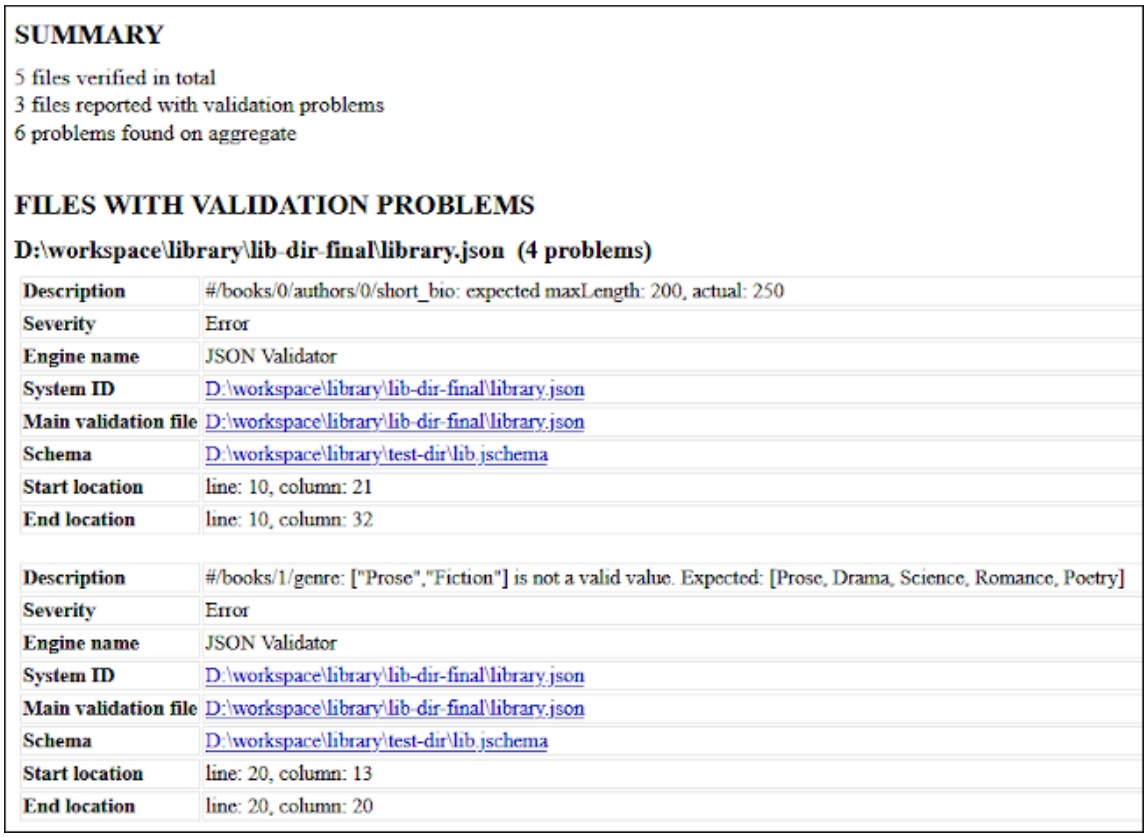
A scenarios file is provided, but without specifying a scenario name. In this case, the argument provided through the **-sf** option is assumed to be an **Oxygen** project file and it is used to search for validation scenario associations made for the `main.xml` file. This command line implies that if validation scenario associations for `main.xml` are found in `proj-1.xpr`, then those scenarios are identified and applied. Otherwise, the validation first considers the schema associations declared in `main.xml` (if any), or default **Oxygen** validation scenarios are applied in accordance with the type of the file to validate (e.g. XML in this example).

Example 4: Directory Default Validation and Custom Formatted Report Saved to a Specific Location

```
sh scripts/validate.sh ../important/xmlFolder -rft html -rf "../important/reports/validation rep.html"
```

No validation scenario name, no scenario file, and no schema provided. This command line involves validating all files from the `xmlFolder`. Each file is validated against the schema(s) internally associated (if any). Otherwise, the default **Oxygen** validation scenarios for the respective file type are applied. Also, the validation report is formatted in HTML and is saved to the `validation rep.html` file at the specified location.

Figure 582. Example of an HTML Validation Report



SUMMARY

5 files verified in total
3 files reported with validation problems
6 problems found on aggregate

FILES WITH VALIDATION PROBLEMS

D:\workspace\library\lib-dir-final\library.json (4 problems)

Description	#/books/0/authors/0/short_bio: expected maxLength: 200, actual: 250
Severity	Error
Engine name	JSON Validator
System ID	D:\workspace\library\lib-dir-final\library.json
Main validation file	D:\workspace\library\lib-dir-final\library.json
Schema	D:\workspace\library\test-dir\lib.jschema
Start location	line: 10, column: 21
End location	line: 10, column: 32

Description	#/books/1/genre: ["Prose","Fiction"] is not a valid value. Expected: [Prose, Drama, Science, Romance, Poetry]
Severity	Error
Engine name	JSON Validator
System ID	D:\workspace\library\lib-dir-final\library.json
Main validation file	D:\workspace\library\lib-dir-final\library.json
Schema	D:\workspace\library\test-dir\lib.jschema
Start location	line: 20, column: 13
End location	line: 20, column: 20

Resources

For more information about the validation script, see the following resources:

- Video: **Validating XML and JSON Using Oxygen Command Line Scripts**
- Webinar: **Validating XML and JSON Documents Using Oxygen Scripting**

Tip:
For some GitHub use case samples of this script, see Oxygen Validation Template and Oxygen Validation Action.

XML Refactoring

Attention:

- This script is bundled with the all platforms distribution of Oxygen XML Developer. To run the script, you are required to purchase a special scripting commercial license.
- To execute an XQuery refactoring operation using this script, in additional to the scripting commercial license, you are required to purchase a Saxon EE license.

The **XML Refactoring** script (`xmlRefactoring.sh`, found in the `scripts` subfolder inside **Oxygen's** installation directory) can be used to execute XML refactoring operations *(on page 578)*. 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 Developer 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 Developer. To run the script, you are required to purchase a special scripting commercial license.

The **DITA Translation Package Builder** (`translationPackageBuilder.sh`, found in the `scripts` subfolder inside **Oxygen's** installation directory) 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 Developer. 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 Developer. To run the script, you are required to purchase a special scripting commercial license.

The **Batch Converter** script (`batchConverter.sh`, found in the `scripts` subfolder inside **Oxygen's** installation directory) 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
- OpenAPI 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 Oxygen XML Developer.

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**, **openapi**, **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, DocBook to DITA, and OpenAPI 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 Developer.

A custom *framework (on page 2305)* (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.

The **Compile Framework Script** (`compileFrameworkScript.sh`, found in the `scripts` subfolder inside **Oxygen's** installation directory) helps you to compile the script into the `*.framework` file that represents the framework configuration. Although Oxygen XML Developer 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 Developer.

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 (*.exf) or directories in file syntax form. If a directory is specified, the Framework Extension Script Files 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 Developer. To run the script, you are required to purchase a special scripting commercial license.

You can generate documentation for XSLT Stylesheets from Oxygen XML Developer 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 `stylesheetDocumentation.sh` script (found in the `scripts` subfolder inside **Oxygen's** installation directory) 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]
```

Tip:
For some GitHub use case samples of this script, see Oxygen Documentation Template and Oxygen Documentation Action.

Related information
XML Schema Documentation *(on page 2275)*
WSDL Documentation (Deprecated) *(on page 2279)*

XML Schema Documentation

Attention:
This script is bundled with the all platforms distribution of Oxygen XML Developer. To run the script, you are required to purchase a special scripting commercial license.

You can generate documentation for XML Schemas from Oxygen XML Developer 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 `schemaDocumentation.sh` script (found in the `scripts` subfolder inside **Oxygen's** installation directory) to generate XML Schema documentation from the command line.

Sample Command-Line Arguments for the Generate XML Schema Documentation Script

```
sh scripts/schemaDocumentation.sh  schemaFile [-cfg:configFile] [-out:outputFile]
[-format:<value>] [-xsl:xslFile] [-split:<value>] [-openInBrowser:<value>]
```

Tip:
For some GitHub use case samples of this script, see Oxygen Documentation Template and Oxygen Documentation Action.

Related information
XSLT Stylesheets Documentation *(on page 2274)*
WSDL Documentation (Deprecated) *(on page 2279)*

JSON Schema Documentation

Attention:
This script is bundled with the all platforms distribution of Oxygen XML Developer. To run the script, you are required to purchase a special scripting commercial license.

You can generate documentation for a JSON Schema file from Oxygen XML Developer by using the **Tools > Generate Documentation > JSON Schema Documentation** main menu action. This tool requires an additional add-on to be installed, so the first time you invoke the action, Oxygen XML Developer presents a dialog box asking if you want to install it. Once installed, you need to restart Oxygen XML Developer and the **JSON Schema Documentation** action will invoke the tool.

You can use the `jsonSchemaDocGen.sh` script (found in the `scripts` subfolder inside **Oxygen's** installation directory) to generate documentation from a JSON schema file using the command line.



Tip:
For some GitHub use-case samples for this script, see Oxygen Documentation Template and Oxygen Documentation Action.

Arguments for the JSON Schema Documentation Script

```
sh scripts/jsonSchemaDocGen.sh filePath [-ofp outputFilePath] [-sb split by (location, components)] [-ea exclude
 annotations] [-ec exclude constraints] [-ep exclude properties] [-epp exclude pattern properties] [-ee exclude
 enumerations] [-es exclude source] [-eub exclude used by] [-ecp exclude compositions] [-ed exclude diagram] [-eim exclude
 image map] [-il include location] [-help | --help | -h | --h]
```

filePath

Mandatory argument specifying the path of the file that needs to generate documentation (it can also be provided as a URL).

-ofp output file path

Optional argument that specifies the path of the output file.

-sb split by (location, components)

Optional argument that specifies what type of split scenario is used (location, components). By default, one file scenario is used.

-ea exclude annotations

Optional argument that specifies if the JSON schema annotations should be present. Default value is true.

-ec exclude constraints

Optional argument that specifies if the JSON schema constraints should be present. Default value is true.

-ep exclude properties

Optional argument that specifies if the JSON schema properties should be present. Default value is true.

-epp exclude pattern properties

Optional argument that specifies if the JSON schema pattern properties should be present. Default value is true.

-ee exclude enumerations

Optional argument that specifies if the JSON schema enumerations should be present. Default value is true.

-es exclude source

Optional argument that specifies if the JSON schema source should be present. Default value is true.

-eub exclude used by

Optional argument that specifies if the JSON schema *used by* should be present. Default value is true.

-ecp exclude compositions

Optional argument that specifies if the JSON schema compositions should be present. Default value is true.

-ed exclude diagram

Optional argument that specifies if the JSON schema diagram should be present. Default value is true.

-eim exclude image map

Optional argument that specifies if the JSON schema image map should be present. Default value is true.

-il include location

Optional argument that specifies if the JSON schema location should be present. Default value is false.

-help | --help | -h | --h

Displays help text.

OpenAPI Documentation



Attention:
This script is bundled with the all platforms distribution of Oxygen XML Developer. To run the script, you are required to purchase a special scripting commercial license.

You can generate documentation for an OpenAPI file from Oxygen XML Developer by using the **Tools > Generate Documentation > OpenAPI Documentation** main menu action. This tool requires an additional add-on to be installed, so the first time you invoke the action, Oxygen XML Developer presents a dialog box asking if you want to install it. Once installed, you need to restart Oxygen XML Developer and the **OpenAPI Documentation** action will invoke the tool.

You can use the `openApiDocGen.sh` script (found in the `scripts` subfolder inside **Oxygen's** installation directory) to generate documentation from an OpenAPI file using the command line.



Tip:
For some GitHub use case samples of this script, see Oxygen Documentation Template and Oxygen Documentation Action.

Arguments for the OpenAPI Documentation Script

```
sh scripts/openApiDocGen.sh filePath [-ofp outputFilePath] [-sb split by (location, components)] [-ea exclude annotations]
 [-ec exclude constraints] [-ep exclude properties] [-epp exclude pattern properties] [-ee exclude enumerations] [-es
 exclude source] [-eub exclude used by] [-ecp exclude compositions] [-ed exclude diagram] [-eim exclude image map] [-il
 include location] [-help | --help | -h | --h]
```

filePath

Mandatory argument specifying the path of the file that needs to generate documentation (it can also be provided as a URL).

-ofp output file path

Optional argument that specifies the path of the output file.

-sb split by (location, components)

Optional argument that specifies what type of split scenario is used (location, components). By default, one file scenario is used.

-ea exclude annotations

Optional argument that specifies if the JSON schema annotations should be present. Default value is true.

-ec exclude constraints

Optional argument that specifies if the JSON schema constraints should be present. Default value is true.

-ep exclude properties

Optional argument that specifies if the JSON schema properties should be present. Default value is true.

-epp exclude pattern properties

Optional argument that specifies if the JSON schema pattern properties should be present. Default value is true.

-ee exclude enumerations

Optional argument that specifies if the JSON schema enumerations should be present. Default value is true.

-es exclude source

Optional argument that specifies if the JSON schema source should be present. Default value is true.

-eub exclude used by

Optional argument that specifies if the JSON schema *used by* should be present. Default value is true.

-ecp exclude compositions

Optional argument that specifies if the JSON schema compositions should be present. Default value is true.

-ed exclude diagram

Optional argument that specifies if the JSON schema diagram should be present. Default value is true.

-eim exclude image map

Optional argument that specifies if the JSON schema image map should be present. Default value is true.

-il include location

Optional argument that specifies if the JSON schema location should be present. Default value is false.

-help | --help | -h | --h

Displays help text.

WSDL Documentation (Deprecated)

Attention:
This script is bundled with the all platforms distribution of Oxygen XML Developer. To run the script, you are required to purchase a special scripting commercial license.

You can generate documentation for WSDL documents from Oxygen XML Developer 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 `wsdlDocumentation.sh` script (found in the `scripts` subfolder inside **Oxygen's** installation directory) 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]
```

Tip:
For some GitHub use case samples of this script, see Oxygen Documentation Template and Oxygen Documentation Action.

Related information

XSLT Stylesheets Documentation *(on page 2274)*
XML Schema Documentation *(on page 2275)*

XML Instance Generator

Attention:
This script is bundled with the all platforms distribution of Oxygen XML Developer. 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 Developer 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 `xmlGenerator.sh` script (found in the `scripts` subfolder inside **Oxygen's** installation directory) 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). This argument can also be provided as a URL.

-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 Developer. 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 Developer by using the **Tools > Flatten Schema** main menu action. You can use the equivalent `flattenSchema.sh` script (found in the `scripts` subfolder inside **Oxygen's** installation directory) to flatten an XML schema from the command line.

Sample Command-Line Arguments for the Flatten Schema Script

```
sh scripts/flattenSchema.sh [-in:inputSchemaURL -outDir:outputDirectory
             [-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 Developer. To run the script, you are required to purchase a special scripting commercial license.

The **Compare Directories** script (`compareDirs.sh`/`compareDirs.bat`, found in the `scripts` subfolder inside **Oxygen's** installation directory) can be used to compare two directories and get the comparison results in various formats.

Arguments for the Compare Directories Script

There are 2 ways that the script's arguments/options can be retrieved (with slightly different command-line syntax), depending on whether or not a configuration file is specified for retrieving the arguments.

Syntax 1 (Using the Command Line to Specify Arguments)

```
sh scripts/compareDirs.sh firstDirPath [-a dirPathAlias] secondDirPath [-a dirPathAlias] [[baseDirPath [-a dirPathAlias]]
 [-sl suppressLinks] [-if includeFilesFilter] [-id includeSubdirsFilter] [-ia includeArchives] [-ef excludeFilesFilter]
 [-ed excludeSubdirsFilter] [-cm comparisonMode] [-alg comparisonAlg] [-als algStrength] [-iws ignoreWS] [-ipi ignorePI]
 [-icm ignoreComments] [-idt ignoreDocType] [-itn ignoreText] [-ins ignoreNS] [-ind ignoreNSDecl] [-inp ignorePrefixes]
```

```
[-iao ignoreAttrOrder] [-iee ignoreExpStateForEmptyElems] [-enx XPathExprToExcludeNodes] [-out outputFormat] [-outfile
outputFile] [-merge mergeOperation] [-mergeout outputDirPathForMerge]] [-help | --help | -h | --h]
```

firstDirPath

Mandatory argument that specifies the first directory path (it can also be provided as a URL using the 'file://' protocol).

secondDirPath

Mandatory argument that specifies the second directory path (it can also be provided as a URL using the '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.

-a dirPathAlias

Optional argument used as the directory path alias to be displayed in the HTML-formatted operation report instead of a long directory path. If present, it must appear immediately after a directory path.

Syntax 2 (Using a Configuration File to Provide Arguments)

```
sh scripts/compareDirs.sh -argsFile argsFilePath [[-fp firstDirPath] [-fpa dirPathAlias] [-sp secondDirPath] [-spa
 dirPathAlias] [-bp baseDirPath] [-bpa dirPathAlias] [-sl suppressLinks] [-if includeFilesFilter] [-id includeSubdirsFilter]
 [-ia includeArchives] [-ef excludeFilesFilter] [-ed excludeSubdirsFilter] [-cm comparisonMode] [-alg comparisonAlg]
 [-als algStrength] [-iws ignoreWS] [-ipi ignorePI] [-icm ignoreComments] [-idt ignoreDocType] [-itn ignoreText]
 [-ins ignoreNS] [-ind ignoreNSDecl] [-inp ignorePrefixes] [-iao ignoreAttrOrder] [-iee ignoreExpStateForEmptyElems]
 [-enx XPathExprToExcludeNodes] [-out outputFormat] [-outfile outputFile] [-merge mergeOperation] [-mergeout
 outputDirPathForMerge]] [-help | --help | -h | --h]
```

-argsFile argsFilePath

Mandatory argument (it must be the first one provided) that specifies the path of the file to retrieve the script arguments from. The file path can also be provided as a URL.

-fp firstDirPath

Optional argument that specifies the first directory path (it can also be provided as a URL using the 'file://' protocol).

-fpa dirPathAlias

Optional argument used as first directory path alias. Displayed in the HTML-formatted operation report instead of a long directory path.

-sp secondDirPath

Optional argument that specifies the second directory path (it can also be provided as a URL using the 'file://' protocol).

-spa dirPathAlias

Optional argument used as second directory path alias. Displayed in the HTML-formatted operation report instead of a long directory path.

-bp 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 using the 'file://' protocol).

-bpa dirPathAlias

Optional argument used as base directory path alias. Displayed in the HTML formatted operation report instead of a long directory path.

Common Arguments for Both Syntaxes

-sl suppressLinks

Set this to **true** to suppress links to the compared directories and files shown in the HTML-formatted operation report. Default value = **false**.

-if includeFilesFilter

Use this argument to only include files that match the specified pattern in the comparison (e.g. `.xml`, `.json`). Default value = ***.

-id includeSubdirsFilter

Use this argument to only include sub-directories that match the specified pattern in the comparison (e.g. `recent,important/latest,reports*/json-rep*/new*`). 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**.

-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 `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

Specifies the format of the output. Possible values: **yaml/grouped**, **yaml/raw**, **json/grouped**, **json/raw**, **xml/grouped**, **xml/raw**, **html**, **html/ifcr**, **htm**, or **htm/ifcr**. Default value = **yaml/grouped**.

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.
- When using a configuration file, the only mandatory argument is **-argsFile argsFilePath**. All others are optional, but if used on the command line, they will take precedence and will overwrite the corresponding values in the config file. The arguments required to run the script can therefore be specified as a combination of what is provided in the config file and what is provided on the command line. For any missing options (not specified either in the file or on the command line), default values are used. However, at least 2 valid directory paths must always be provided in one way or another, otherwise the script execution will be aborted. For details about the structure of the configuration file (XML, JSON, or YAML), see: Providing Arguments to Comparison Scripts via a Configuration File *(on page 2293)*.
- 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:

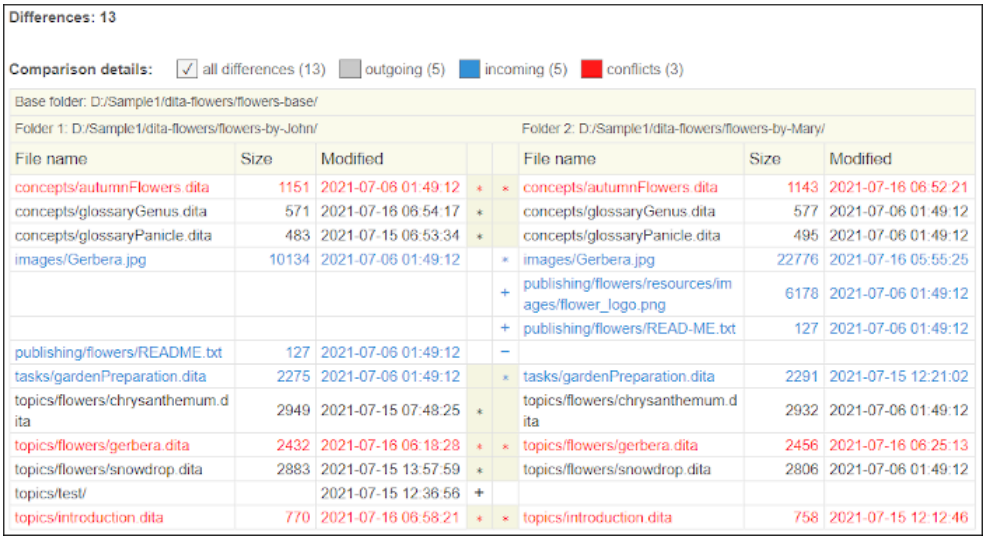
```
sh scripts/compareDirs.sh dir1 dir2 -out json/raw > results.json
```

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 583. Example of an HTML Report for Directory Comparison



Differences: 13

Comparison details: ☑ all differences (13) ☐ outgoing (5) ☐ incoming (5) ☐ conflicts (3)

Base folder: D:/Sample1/dita-flowers/flowers-base/

Folder 1: D:/Sample1/dita-flowers/flowers-by-John/					Folder 2: D:/Sample1/dita-flowers/flowers-by-Mary/		
File name	Size	Modified			File name	Size	Modified
concepts/autumnFlowers.dita	1151	2021-07-06 01:49:12	*	*	concepts/autumnFlowers.dita	1143	2021-07-16 06:52:21
concepts/glossaryGenus.dita	571	2021-07-16 06:54:17	*		concepts/glossaryGenus.dita	577	2021-07-06 01:49:12
concepts/glossaryPanicle.dita	483	2021-07-15 06:53:34	*		concepts/glossaryPanicle.dita	495	2021-07-06 01:49:12
images/Gerbera.jpg	10134	2021-07-06 01:49:12		*	images/Gerbera.jpg	22776	2021-07-16 05:55:25
				+	publishing/flowers/resources/images/flower_logo.png	6178	2021-07-06 01:49:12
				+	publishing/flowers/READ-ME.txt	127	2021-07-06 01:49:12
publishing/flowers/README.txt	127	2021-07-06 01:49:12		−			
tasks/gardenPreparation.dita	2275	2021-07-06 01:49:12		*	tasks/gardenPreparation.dita	2291	2021-07-15 12:21:02
topics/flowers/chrysanthemum.dita	2949	2021-07-15 07:48:25	*		topics/flowers/chrysanthemum.dita	2932	2021-07-06 01:49:12
topics/flowers/gerbera.dita	2432	2021-07-16 06:18:28	*	*	topics/flowers/gerbera.dita	2456	2021-07-16 06:25:13
topics/flowers/snowdrop.dita	2883	2021-07-15 13:57:59	*		topics/flowers/snowdrop.dita	2806	2021-07-06 01:49:12
topics/test/		2021-07-15 12:36:56	+				
topics/introduction.dita	770	2021-07-16 06:58:21	*	*	topics/introduction.dita	758	2021-07-15 12:12:46

Example 5: Compare Directories by Getting Arguments from a Configuration File

The arguments are read from the **config.xml** file, including the mandatory **path** arguments. However, the values that correspond to the arguments with the IDs **-out** and **-outfile** are overwritten by the **yaml** and **anotherResults.yaml** values provided on the command line (even if specified in the **config.xml** file).

```
sh scripts/compareDirs.sh -argsFile config.xml -out yaml -outfile anotherResults.yaml
```

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**.

Tip:
For some GitHub use case samples of this script, see Oxygen Comparison Template and Oxygen Comparison Action.

Related information
Compare Directories Tool *(on page 421)*
Compare Files Script *(on page 2288)*

Compare Files

Attention:
This script is bundled with the all platforms distribution of Oxygen XML Developer. To run the script, you are required to purchase a special scripting commercial license.

The **Compare Files** script (`compareFiles.sh`/`compareFiles.bat`, found in the `scripts` subfolder inside **Oxygen's** installation directory) can be used to compare files and get the comparison results in various formats.

Arguments for the Compare Files Script

There are 2 ways that the script's arguments/options can be retrieved (with slightly different command-line syntax), depending on whether or not a configuration file is specified for retrieving the arguments.

Syntax 1 (Using the Command Line to Specify Arguments)

```
sh scripts/compareFiles.sh firstFilePath secondFilePath [[baseFilePath] [-ct contentType] [-alg comparisonAlg]
 [-als algStrength] [-iws 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).

Syntax 2 (Using a Configuration File to Provide Arguments)

```
sh scripts/compareFiles.sh -argsFile argsFilePath [[-fp firstFilePath] [-sp secondFilePath] [-bp baseFilePath] [-ct
 contentType] [-alg comparisonAlg] [-als algStrength] [-iws 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] [-outfile
 outputFile] [-merge mergeOperation] [-mergeout outputDirPathForMerge]] [-help | --help | -h | --h]
```

-argsFile argsFilePath

Mandatory argument (it must be the first one provided) that specifies the path of the file to retrieve the script arguments from. The file path can also be provided as a URL.

-fp firstFilePath

Optional argument that specifies the first file path (it can also be provided as a URL).

-sp secondFilePath

Optional argument that specifies the second file path (it can also be provided as a URL).

-bp 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).

Common Arguments for Both Syntaxes

-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**.

-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**.

-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.



Tip:
When using a configuration file, the only mandatory argument is **-argsFile argsFilePath**. All others are optional, but if used on the command line, they will take precedence and will overwrite the corresponding values in the config file. The arguments required to run the script can therefore be specified as a combination of what is provided in the config file and what is provided on the command line. For any missing options (not specified either in the file or on the command line) default values

are used. However, at least 2 valid paths must always be provided in one way or another, otherwise the script execution will be aborted. For details about the structure of the configuration file (XML, JSON, or YAML), see: Providing Arguments to Comparison Scripts via a Configuration File *(on page 2293)*.

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:

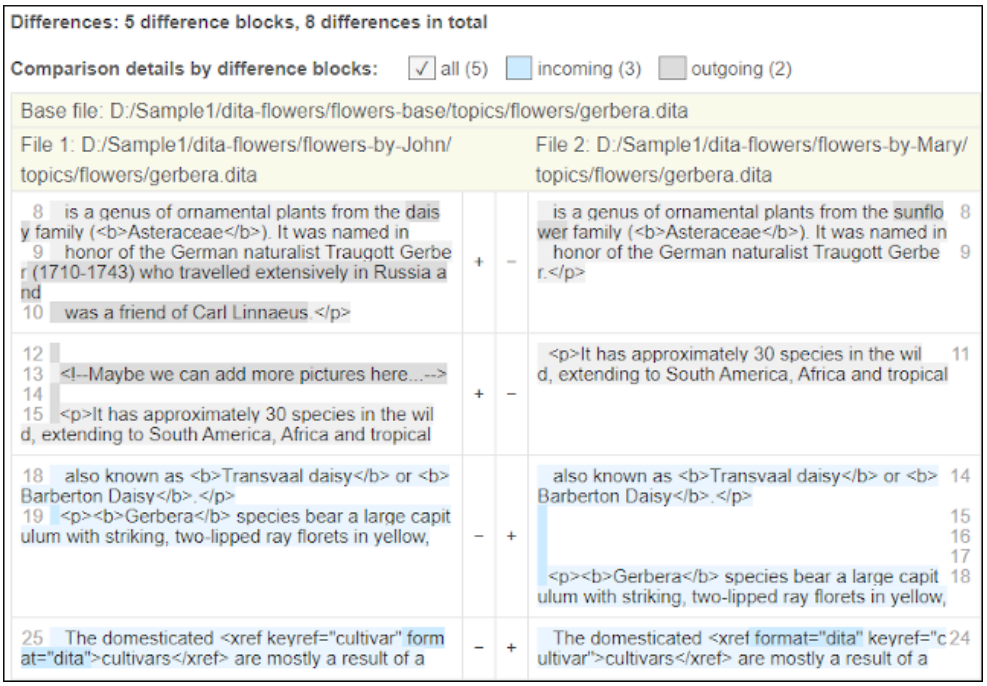
```
sh scripts/compareFiles.sh file1 file2 -alg lines -iws
```

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 584. Example of File Comparison Report in HTML Format



Example 4: Compare Files by Getting Arguments from a Configuration File

The arguments are read from the **config.xml** file, including the mandatory **file path** arguments. However, the values that correspond to the arguments with the IDs **-out** and **-outfile** are overwritten by the **yaml** and **anotherResults.yaml** values provided on the command line (even if specified in the **config.xml** file).

```
sh scripts/compareFiles.sh -argsFile config.xml -out yaml -outfile anotherResults.yaml
```

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 2281)*

Providing Arguments to Comparison Scripts via a Configuration File

The arguments/options required to run comparison scripts (Compare Directories *(on page 2281)* and Compare Files *(on page 2288)*) can be provided through configuration files (argument files) if their structure follows a few simple rules, as detailed below:

- The basic format of the configuration file is XML, but JSON or YAML formatted files are also supported, being converted to XML.
- The name of the root does not matter.
- Any child of the root (also, the name of the child does not matter) is a candidate element for describing a script argument, if it meets the following 2 mandatory conditions:
  - It has a sub-element `<id>argument_id</id>` (`argument_id` must be present for it to be mapped to the `argument_value`).
  - It has a sub-element `<value>argument_value</value>` (if `argument_value` is missing, the mapping is completed programmatically to a default value based on `argument_id`).
- Additionally:
  - If `<id>` denotes an argument ID of the type 'file path' and an alias is also desired, then a sub-element `<alias>path_alias</alias>` must be added. However, an alias also makes sense as a distinct argument ID, meaning for example, the sub-element structure `<id>spa</`

`id><value>directory-2</value>` will correctly associate the alias directory-2 with `sp` ('*sp*' stands for 'second path', '*spa*' stands for 'second path alias').
    - An optional sub-element `<descr>argument_description</descr>` can be useful when editing/modifying the XML file, providing explanations about the ID of the current argument (which is most often quite cryptic).
- If there are non-mandatory script arguments without counterparts among the child elements of the root of the parsed XML file, default values associated with the IDs of the respective arguments are provided later programmatically.

Below are some examples of configuration files that can be used to read the script arguments from:

XML File

```
<?xml version="1.0" encoding="UTF-8"?>
<args>
    <arg>
        <id>fp</id>
        <value>D:/workspace/oxygen-blog/blog-branch-01</value>
        <alias>DIR-1</alias>
        <descr>1st path</descr>
    </arg>
    <arg>
        <id>sp</id>
        <value>D:/workspace/oxygen-blog/blog-branch-02</value>
        <alias>DIR-2</alias>
        <descr>2nd path</descr>
    </arg>
    <arg>
        <id>bp</id>
        <value>D:/workspace/oxygen-blog/blog-main</value>
        <descr>base path</descr>
    </arg>
    <arg>
 <descr>base path alias</descr>
        <id>bpa</id>
        <value>BASE</value>
    </arg>
    <arg>
        <id>ia</id>
        <value>false</value>
        <descr>look inside archives</descr>
    </arg>
    <arg>
        <id>ed</id>
```

```
        <value>git*</value>
        <descr>exclude folders filter</descr>
    </arg>

 ..................................

<args>
```

JSON File

```
{
    "options": {
        "option": [
            {
                "id": "fp",
                "value": "D:/workspace/oxygen-blog/blog-branch-01",
                "alias": "DIR-1",
  "descr": "first path"
            },
            {
                "id": "sp",
                "value": "D:/workspace/oxygen-blog/blog-branch-02",
                "alias": "DIR-2",
  "descr": "second path"
            },
     {
                "id": "ed",
                "descr": "exclude folders filter",
                "value": "git*"
            }
     {
                "id": "ia",
                "descr": "look inside archives",
                "value": false
            },


   ........................


 ]
    }
}
```

YAML File

```
---
arguments:
  argument:
  - id: fp
    descr: 1st path
    value: D:/workspace/oxygen-blog/blog-branch-01
    alias: DIR-1
  - id: sp
    descr: 2nd path
    value: D:/workspace/oxygen-blog/blog-branch-02
    alias: DIR-2
  - id: bp
    descr: base path
    value: D:/workspace/oxygen-blog/blog-main
    alias: BASE
  - id: ia
    value: false
    descr: include archives
  - id: ed
    value: git*
    descr: exclude folders filter


.....................................
```

Related information

Compare Directories Script *(on page 2281)*
Compare Files Script *(on page 2288)*

Merge Files with Change Tracking Highlights

Attention:
This script is bundled with the all platforms distribution of Oxygen XML Developer. To run the script, you are required to purchase a special scripting commercial license.

The **Merge Files with Change Tracking Highlights** script (`mergeFilesTrackChanges.sh`, found in the `scripts` subfolder inside **Oxygen's** installation directory) can be used to merge 2 XML files (based on a 2-way comparison). The **Author** mode comparison results are saved as documents with highlighted tracked changes that can later be reviewed and accepted or rejected.

Notice:
This script is intended to be used for merging XML documents (Oxygen XML Developer creates change tracking markers only for XML file types. Using this script for document types other than

XML, or for documents that are not XML well-formed, causes document parsing errors and the merge operation fails.

Arguments for the Merge Files with Change Tracking Highlights Script

```
sh mergeFilesTrackChanges.sh pathOfBaseFile pathOfFileToMergeWith [[pathOfOutFile] [-nb noBackupOfBaseFile]] [-help | --help | -h | --h]
```

pathOfBaseFile

Mandatory argument that specifies the path of the base file (it can also be provided as a URL).

pathOfFileToMergeWith

Mandatory argument that specifies the file to merge with (it can also be provided as a URL).

pathOfOutFile

Optional argument that specifies the path of the file where the merge operation results are saved to (it can also be provided as a URL). If present, it must appear immediately after the first two mandatory arguments. If absent, the merge results are saved to the base file, by overwriting it. You cannot choose the same file specified as the file to merge with as the output file (the merge process is aborted in this case). Also, if the output is a remote resource, its entire parent directory structure must already exist. Otherwise, an I/O exception is thrown and the merge results cannot be saved.

-nb noBackupOfBaseFile

Set to **true** if you do not want a backup copy of the base file on the hard disk. There are 2 situations when a backup of the base file is performed automatically and the backup operation must succeed to proceed with the merge. Otherwise, the merge process is aborted if the output file is not specified (i.e. the `pathOfOutFile` argument is not present) or the specified output file is the base file itself.

The backup copy will have the same parent directory as the base directory and its name will be the name of the base file suffixed by ".OXY.BAK". The default value is **false**, which means that for either of the 2 previously mentioned situations, a backup copy of the base file will be kept on the hard disk.

Note:
The backup copy can be deleted only if the base file and, implicitly, its backup copy are local resources (not remote).

-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 results in merging `file1` and `file2` into `outfile` with changes highlighted:

```
sh scripts/mergeFilesTrackChanges.sh file1 file2 outfile
```

Example 2: Compare Files with Line by Line Algorithm

The following command results in merging `file1` and `file2` by overwriting `file1`. However, the `file1` is backed up first:

```
sh scripts/mergeFilesTrackChanges.sh file1 file2
```

Example 3: Compare Files and Generate Comparison Report

The following command results in merging `file1` and `file2` by overwriting `file1`. Although `file1` is initially backed up, the backup is eventually removed:

```
sh scripts/mergeFilesTrackChanges.sh file1 file2 -nb
```

Merge Directories with Change Tracking Highlights

Attention:
This script is bundled with the all platforms distribution of Oxygen XML Developer. To run the script, you are required to purchase a special scripting commercial license.

The **Merge Directories with Change Tracking Highlights** script (`mergeDirsTrackChanges.sh`, found in the `scripts` subfolder inside **Oxygen's** installation directory) can be used to merge 2 directories (based on a 2-way comparison). All pairs of modified XML files involved in the process are merged by saving the **Author** mode comparison results as documents with highlighted tracked changes that can be later reviewed and accepted or rejected.

Arguments for the Merge Directories with Change Tracking Highlights Script

```
sh mergeDirsTrackChanges.sh pathOfBaseDir pathOfDirToMergeWith [[pathOfOutDir] [-nb noBackupOfBaseDir] [-nu
 noUpdateOfModifNonXMLFiles] [-na noAddingFilesOnlyPresentInDirToMergeWith] [-nd noDeletionOfFilesOnlyPresentInBaseDir] [-cm
 createChangeTrackingMarkersForAddedXMLFiles]] [-help | --help | -h | --h]
```

pathOfBaseDir

Mandatory argument that specifies the path of the base directory (it can also be provided as a URL using `file://` protocol).

pathOfDirToMergeWith

Mandatory argument that specifies the path of the directory to merge with (it can also be provided as a URL using `file://` protocol).

pathOfOutDir

Optional argument that specifies the path of the directory where the merge operation results are saved to (it can also be provided as a URL using `file://` protocol). If present, it must appear immediately after the first two mandatory arguments. If absent, the merge results are saved to the base directory, by overwriting it. You cannot choose the same directory specified as the directory to merge with as the output directory (the merge process is aborted in this case).

-nb noBackupOfBaseDir

Set to **true** if you do not want a backup copy of the base directory on the hard disk. There are 2 situations when a backup of the base directory is performed automatically and the backup operation must succeed to proceed with the merge. Otherwise, the merge process is aborted if the output directory is not specified (i.e. the `pathOfOutDir` argument is not present) or the specified output directory is the base directory itself.

The backup copy will have the same parent directory as the base directory and its name will be the name of the base directory suffixed by ".OXY.BAK". The default value is **false**, which means that for either of the 2 previously mentioned situations, a backup copy of the base directory will be kept on the hard disk.

-nu noUpdateOfModifNonXMLFiles

Set to **true** if you want to keep the non-XML files at their versions from the base directory. The default value is **false**, which means that all files in the output directory that are copies of non-XML files in the base directory will be replaced by their corresponding files in the directory to merge with.

-na noAddingFilesOnlyPresentInDirToMergeWith

Set to **true** if you want to skip adding the files that are only present in the directory to merge with to the output directory as well. The default value is **false**, which means that all files that are only present in the directory to merge with are also added to the output directory.

-nd noDeletionOfFilesOnlyPresentInBaseDir

Set to **true** if you want to preserve the files that are only present in the base directory. The default value is **false**, which means that all files that are only present in the base directory and initially copied to the output directory are deleted.

-cm createChangeTrackingMarkersForAddedXMLFiles

Set to **true** if you want to create change tracking markers for the XML files only present in the directory to merge with (that will be added to the output directory). Although these files have no counterparts in the base directory, change tracking markers of the type "entire content added/inserted" will be created. The option is not necessarily intended for the merge process itself, but it might prove a useful addition when you want to apply various *Oxygen* transformation scenarios to the resulting output directory. For example, if you merge 2 versions of a DITA project and then want a PDF to highlight the changes between those versions, you can apply a transformation on the resulting `ditamap` file. The `-am` option presents the new DITA files as "added content" in the resulting PDF. Note that the option is only considered if the `-na` argument is absent or is explicitly set to **false** (default value).

-help | --help | -h | --h

Displays help text.



Notes:

- The merge process has a preliminary phase where the entire structure and content of the base directory is copied to the output directory.
- 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.
- Once the merge operation is complete, a report file is created and saved in the output directory (in a separate subdirectory named "._OXY_MERGE_REPORT"). Loading the report file in Oxygen XML Developer provides additional functionality. Aside from the fact that the report provides an overview of the merge process, it also provides links to all the files in the resulting output directory. You can use the respective links to load the XML files in the editor, then switch to **Author** mode to review the tracked changes and accept or reject them.

Examples of Compare Directories Script

Example 1: Compare Directories Without Updating non-XML Files

The following command results in merging `dir1` and `dir2` into `outdir`, but without updating the non-XML files in `dir1` detected with changes to their version in `dir2`:

```
sh scripts/mergeDirsTrackChanges.sh dir1 dir2 outdir -nu
```

Example 2: Compare Two Directories and Overwrite the First One

The following command results in merging `dir1` and `dir2` by overwriting `dir1`. However, the `dir1` is backed up first:

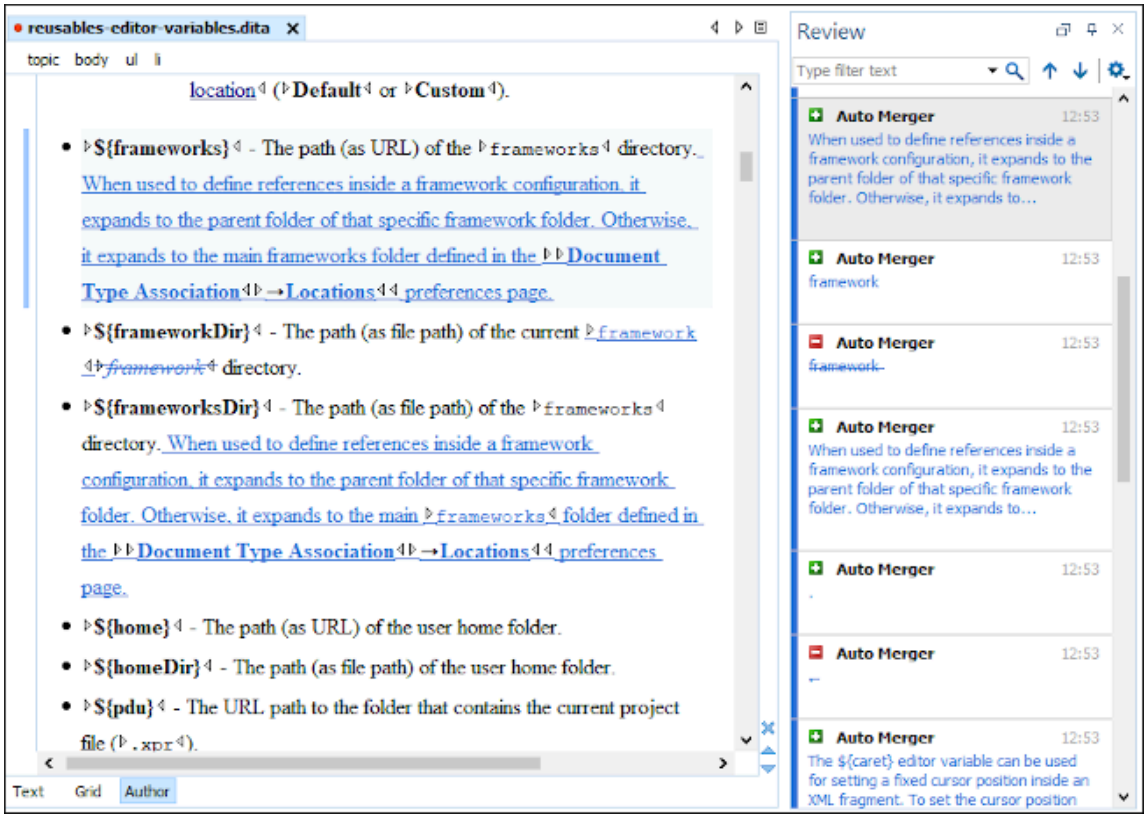
```
sh scripts/mergeDirsTrackChanges.sh dir1 dir2
```

Example 3: Compare Directories and Create Change Tracking Markers

The following command results in merging `dir1` and `dir2` into `outdir` and creating change tracking markers (of the type "entire content added/inserted") for all XML files that are only present in `dir2` (to be added to the `outdir`):

```
sh scripts/mergeDirsTrackChanges.sh dir1 dir2 outdir -cm
```

Figure 585. Example of a Merged File with Change Tracking Markers Opened in Author Mode



Format and Indent Files

Attention:
This script is bundled with the all platforms distribution of Oxygen XML Developer. To run the script, you are required to purchase a special scripting commercial license.

The **Format and Indent Files** script (`batchFormatAndIndent.bat`/`batchFormatAndIndent.sh`, found in the `scripts` subfolder inside **Oxygen's** installation directory) 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.

23.
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.

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.

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 Developer 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 Text Mode *(on page 457)*
- Content Completion Assistant in Grid Mode *(on page 512)*
- Content Completion in XSLT Stylesheets *(on page 632)*
- Content Completion in Ant Build Files *(on page 678)*
- Content Completion in XML Schema *(on page 733)*
- Content Completion in XQuery *(on page 775)*
- Content Completion Assistance in WSDL Documents *(on page 793)*
- Content Completion in CSS Stylesheets *(on page 818)*
- Content Completion in LESS Stylesheets *(on page 822)*
- Content Completion in Relax NG Schemas *(on page 829)*
- Content Completion in NVDL Schemas *(on page 845)*
- Content Completion in JavaScript Documents *(on page 943)*
- Content Completion in Schematron Documents *(on page 965)*
- Content Completion in SQF *(on page 1003)*

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 Type Association

In general terms, a ***Document Type Association*** is a set of rules that associate a document type with a *framework (on page 2305)*. In Oxygen XML Developer, ***Document Type Association*** also specifically refers to a preferences page *(on page 93)* where you can create new custom *frameworks* or edit existing ones. Note that *frameworks* (document types) that come built-in with Oxygen XML Developer are read-only, but you can **Extend** *(on page 94)* or **Duplicate** *(on page 94)* them to configure them as custom *frameworks*.

DITA Map

A ***DITA map*** is a component of the DITA *framework (on page 2305)* 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 2303)* 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 200)*.

For example, if you are using DITA-OT 4.4.0 that comes bundled with Oxygen XML Developer, the default directory is: *[OXYGEN_INSTALL_DIR]*`/frameworks/dita/`*DITA-OT*. You can also specify a custom directory.

Foldable Element

A ***foldable element*** refers to elements that can be collapsed and expanded in Oxygen XML Developer. *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 a vocabulary or document type. A *framework* is associated to a 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 Developer includes a **Document Type Configuration Dialog Box** *(on page 95)* that allows you to define the set of rules and customize various authoring mechanisms for new or existing *frameworks*.

Global Options

Global Options refers to the storage option *(on page 240)* in the Oxygen XML Developer preference pages (**Options > Preferences**). If you select **Global Options** *(on page 241)*, 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 241)* that can then be shared).

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 2309)* defines a set of effective key bindings.

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 Developer:

- **Java Key Store (JKS)**
- **Public-Key Cryptography Standards version 12 (PKCS-12)**

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 350)* section.

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 t 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 1271)*

Perspective

In Oxygen XML Developer, 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 () in the top-right corner of Oxygen XML Developer or by selecting the *perspective* from the **Window > Open Perspective** menu.

The *perspectives* that are available in Oxygen XML Developer are:

- **Editor** *(on page 272)* - The most commonly used *perspective* and it is used to edit XML documents.
- **XSLT Debugger** *(on page 274)* - Used to detect problems in an XSLT transformation by executing the process step by step in a controlled environment.
- **XQuery Debugger** *(on page 275)* - Used to detect problems in an XQuery transformation process by executing the process step by step in a controlled environment
- **Database** *(on page 276)* - Used to browse and manage databases.

Plugin

In Oxygen XML Developer, 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 Developer, see Oxygen XML Add-on Repositories.

For more information, see the following topics:

- Installing and Updating Add-ons *(on page 73)*
- Automatic and Manual Methods for Installing Plugins *(on page 1860)*
- Packing and Deploying Plugins as Add-ons *(on page 1891)*
- Add-ons Preferences *(on page 92)*
- Extending Oxygen XML Developer with Plugins *(on page 1857)*
- General Configuration of an Oxygen XML Developer Plugin *(on page 1857)*

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 Developer 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 334)* for one or more selected files.

Project Options

Project Options refers to the storage option *(on page 240)* in the Oxygen XML Developer preference pages (**Options > Preferences**). If you select **Project Options** *(on page 241)*, 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 241)*.

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 Fix/Assist

The ***Quick Fix/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 Developer 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 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.
- 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 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 + Option + 1 on macOS)** on your keyboard.
- The *Quick Fix* menu also contains the option to automatically fix all validation errors in the document where quick fix proposals were generated. By selecting **Apply all default quick fix proposals**, each default quick fix proposal for each validation problem reported in the document will be applied (in bulk). The effect is similar to invoking the **Apply all default quick fix proposals** action from the contextual menu in Text mode *(on page 503)*.

Oxygen XML Developer also provides support for defining and customizing a library of *Quick Fixes* using the Schematron language *(on page 991)*.

Root Map

A ***Root Map*** (or main map) specifies a *DITA map (on page 2304)* 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 2306)*).

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 Developer includes default global catalogs as well as default catalogs for each of the built-in *frameworks (on page 2305)*, and you can also create your own. Oxygen XML Developer uses these *XML Catalogs* to resolve references for document validation and transformations. For more information, see Working with XML Catalogs *(on page 564)*.

Index

A

Add fonts to built-in FO processor
1174, 1176
Add hyphenation libraries to built-in DITA-OT FO processor
1723
Add hyphenation libraries to built-in FO processor
1177
Add PDF image support to built-in DITA-OT FO processor
1723
Add PDF image support to built-in FO processor
1177
Add Schematron Quick Fix
991
Add-on
2307
Add-on preferences
92
Add-ons
1921, 2307
Check for add-on updates
73
Install new add-ons
73
Manage add-ons
73
Adding custom views
1864
Additional Framework plugin extension
1869
Additional languages plugin extension
1872
Additional XProc engine plugin extension
1869
Addons
1921
Annotation preferences
150
Ant build files
675
Component Dependencies view
685
Content completion
678
Editing in Main Files context
676
Highlight occurrences
686
Outline view
680
Quick Assist feature
688
Quick fix support
678
Refactoring actions
687
Referenced/Dependent Resources view
683
Search declarations
686
Search references
686
Syntax highlighting
679
Transforming
677
Validation
676
Ant preferences
197
Ant transformation scenario
1140
Options tab
1141
Output tab
1144
Parameters tab
1143
Appearance preferences

84
Apply all default quick fix proposals
615, 1960
Archive preferences
219
Archives
1735
Browse
1735
Edit files
1740
EPUB
1738
File browser
1735
Migrate OOXML to DITA
1741
Migrate OOXML to TEI
1741
Modify
1735
ODF
1738
OOXML
1738
Associate JSON schema in a framework configuration
869
Associate schema directly in JAML documents
928
Associate schema directly in JSON documents
868
Associate schema directly in XML documents
561
Associate schema in a framework configuration
563
Associate schema in a validation scenario defined in framework
559
Associate schema through a validation scenario
555, 865
Associate schema to a JSON document
865
Associate schema to an XML document
554
AsyncAPI documents
Content completion
1067
Editing features
1067
Validation
1067
Attributes view in Design mode
737
Attributes view in Text mode
468
Author Action dialog box
103
Author editing mode
Entities view
472
MathML equations in HTML output
1403
Model view
470
Auto recovery
316
Automated tests
1892
Automatic Spell Check
388
Automatic validation
517
Automatic validation in JSON
856
Available memory
271

B

Backup file
316
BaseX database connection
1780
BaseX database contextual menu actions
1780

BaseX XQJ connection
1783
Batch Documents Converter Addon - Convert various document tpes like HTML, Markdown, Word, Excel to XML formats like DITA XML, DocBook, XHTML
1921
Batch transformation
346, 1215
Batch validation
346
Bidirectional text in Grid mode
512
Bidirectional text in Text mode
489
Browsing for remote files with SharePoint
1812
Built-in frameworks
DITA Map
1055
DITA Topic
1054
DocBook 4
1042
DocBook 5
1047
DocBook 5.1
1047
DocBook Assembly
1051
DocBook Targetset Map
1053
DocBook Topic
1052
EPUB
1064
JATS
1063
jTEI
1062
S1000D
1056
TEI ODD
1061
TEI P5
1059
XHTML
1058
Built-in XML refactoring operations
582, 1925

C

Canonicalize files tool
620, 1999
Certificates preferences
199
CGM Image Support Addon - Experimental support for rendering CGM 1.0 images in the Author visual editing mode
1921
Change language for interface
266
Changing the font size in Text mode
446
Character Map dialog box
394
Check for add-on updates
73
Check Spelling
380
Check Spelling in multiple Files
389
Check Well-Formedness action
515
Checking Well-Formedness in JSON
855
CHM Error HHC5003: Compilation failed while compiling file
2220, 2220
CHM Error HHC5010: Cannot open file
2220, 2220
Class Loader
1890
Class Loader issues
1889

Close file
317
Code template preferences
151
Code templates
462
Color preferences
86
Common problems and solutions
2207
Compare Directories Against a Base tool
427, 2046
Compare Directories tool
421, 2040
Compare images
427, 2046
Menus
425, 2045
Toolbar
423, 2043
Compare documents with change tracking highlights
2032
Compare Files tool
402, 2012
Command-line arguments
409, 2020
Integrate with Git
410, 2021
Integrate with Sourcetree
411, 2022
Menus
416, 2027
Toolbar
413, 2024
Comparing files and directories
401
Compile LESS to CSS
823
Compile XSL Stylesheet for Saxon tool
672, 1968
Components Validation plugin extension
1870
Configure Calabash with XEP
1189
Configure transformation scenario
1207
Configure Transformation Scenario dialog box
1210
Configuring options
237
Configuring Oxygen
79
Configuring Toolbars
249, 291
Content Completion Assistant in Grid Mode
512
Content completion helper views
462
Content completion in HTML
1012
Content completion in JSON
863
Content completion in Text Mode
457
Content completion in YAML
929
Content completion preferences
144
Content Fusion Connector Addon - Create review tasks and collaborate using the Oxygen Content Fusion server
1921
Contextual actions in YAML documents
934
Contextual menu actions in JSON
880
Contextual menu actions in Text Mode
492
Contextual menu of current editor tab
327
Convert between JSON and XML from XSLT
674
Convert database to XML Schema

766, 1965
Convert JSON to XML
873, 1975
Convert JSON to YAML
879, 933, 1974
Convert schema to another schema language
763, 1962
Convert XML to JSON
876, 1978
Convert XSD to JSON Schema
919, 1982
Convert YAML to JSON
879, 932, 1974
Copy/Paste in Grid Mode
510
Create JSON schema from learned document structure
869
Create new transformation scenario
1101
Create new validation scenario
526
Create Schematron Quick Fix
991
Creating Markdown documents
1018
Creating new documents
299
Creating publishing templates
1309, 1464
CSS processors preferences
196
CSS stylesheets
Content completion
818
Custom CSS properties
817
Editing features
817
Folding
820
Format and indent (pretty print)
820
Minifying
820
Outline view
819
Syntax highlighting
819
Validation
817
CSS validation preferences
168
Current license was already activated
70, 2227
Custom editor variables
229
Custom Protocol plugin extension
1873
Custom system properties
261
Custom validation engine preferences
161
Custom XML refactoring operations
594, 1938
Custom XSLT/XQuery transformation engine preferences
191
Customize document templates
309
Customizing default options
238

D

Dark color theme
84
Dark mode
84
Data Source Explorer view
1742
Data Sources preferences page
205
Database connection preferences
205
Database drivers

210
Database perspective
276
Databases
1742
Connections
1747
BaseX
1780
Contextual menu actions
1780
XQJ
1783
eXist
1761
Connection wizard
1761
Contextual menu actions
1763
Manual configuration
1762
Generic JDBC
1777
IBM DB2
1784
Contextual menu actions
1788
JDBC-ODBC
1778
MarkLogic
1766
Contextual menu actions
1773
Debugging
1770, 1799
MarkLogic development
1769
Microsoft SQL Server
1748
Contextual menu actions
1751
MySQL
1775
Oracle
1752
Contextual menu actions
1755
PostgreSQL
1757
Contextual menu actions
1760
SharePoint
1802
Browsing for remote files
1812
Contextual menu actions
1809
MS Azure authentication
1813
SharePoint Browser view
1807
WebDAV
1789
Contextual menu actions
1789
Data Source Explorer view
1742
SQL support
1792
Table Explorer view
1744
XQuery
1795
Debugging
1799
MarkLogic
1770, 1799
Drag and drop from Data Source Explorer
1795
Transformations
1796
Validation
1796
Debug PDF transformation

1723
Debugger preferences
188
Debugging XQuery
1826
Breakpoints
1849
Debugging Java extensions
1855
Identify expressions
1846
Information views
1832
Breakpoints view
1833
Context view
1834
Messages view
1836
Nodes/Values Set view
1842
Output Mapping Stack view
1838
Stack view
1837
Templates view
1841
Trace view
1840
Variables view
1843
XPath Watch view
1835
Layout
1827
Performance profiling
1850
Profiling
Hotspots view
1853
Invocation Tree view
1852
Steps in a typical debugging process
1845
Toolbar
1828
Debugging XSLT
1826
Breakpoints
1849
Debugging Java extensions
1855
Debugging XSLT that call Java extensions
1855
Identify expressions
1846
Information views
1832
Breakpoints view
1833
Context view
1834
Messages view
1836
Nodes/Values Set view
1842
Output Mapping Stack view
1838
Stack view
1837
Templates view
1841
Trace view
1840
Variables view
1843
XPath Watch view
1835
Layout
1827
Performance profiling
1850
Profiling
Hotspots view

1853
Invocation Tree view
1852
Steps in a typical debugging process
1845
Supported processors
1856
Toolbar
1828
Debugging XSLT that call Java extensions
1855
Design editing mode
Attributes view
737
Components
703
Grouping components
728
xs:all
720
xs:alternative
716
xs:any
721
xs:anyAttribute
723
xs:assert
726
xs:attribute
709
xs:attributeGroup
711
xs:choice
720
xs:complexType
711
xs:element
705
xs:field
726
xs:group
717
xs:import
719
xs:include
718
xs:key
724
xs:keyRef
725
xs:notation
720
xs:openContent
727
xs:override
719
xs:redefine
719
xs:schema
704
xs:selector
725
xs:sequence
720
xs:simpleType
714
xs:unique
723
Contextual menu actions
695
Editing actions
694
Facets view
692
Navigation
690, 896
Outline view
735
Palette view
691
Design mode preferences
131
Detect Main Files
352

Detect Main Files from Project
352
Diagram preferences
128
Diff Directories tool
421, 2040
Diff Files tool
402, 2012
Diff tool
402, 2012
Digital Signature
Canonicalize files
620, 1999
Certificates
620
Sign files
622, 2000
Verify signature
625, 2003
Digital Signatures
617
Example of how to digitally sign XML content
625
Overview
617
Directory comparison appearance preferences
219
Directory comparison preferences
218
DITA
Output
Syntax highlights in codeblocks
1341
DITA Logging preferences
203
DITA Map document type
1055
DITA Map Metrics Report transformation
1093
DITA Map PDF - based on HTML5 & CSS
1087
Abbreviated-form element
1659
Accessibility
1599, 1599
Add
Strings
1704
Appendices
1587
Page breaks
1588
Archiving
1600, 1601
Back matter
1543
Style topics
1544
Bookmap
1662
Bookmarks
1575
Change labels
1575
Depth
1576
Initial state
1577
Remove numbering
1577
Sections display
1576
Changing the cover page
1662
Changing the page size
1662
Changing the title
1661
Changing the TOC depth
1661
Command Line
1455
Comments
1606

Styling
1610
Cover page
1511
Add empty pages
1520
Add second cover
1518
Add text
1517
Background image
1513
Copyright page
1521
Place cover on left side
1518
Place cover on right side
1518
Title styling
1516
Custom transformation parameters
1660
Customization CSS
1469
Customize
Strings
1703
Debugging
1471, 1483
XPath Expressions
Debug
1475
Write
1475
Deep numbering
1550
Double sided pagination
1571
Force even number of pages
1573
Force odd number of pages
1573
Start chapters on odd page
1572
Style blank pages
1572
Style first page
1573
Draft watermarks
1612
Conditional draft watermark
1613, 1613
Figures
Figure numbering
1634
Flagging content
1614
Fonts
1601
Asian languages
1604
Content
1602
Music
1605
Symbols
1603
Titles
1602
Footer
1485
Add copyright
1498
Add topics
1499
Chapter Number
1507
Page Number
1507
Footnotes
1588
Reset Counter
1590
Style Markers and Calls

1589
Front matter
1543
Page breaks
1544
Style topics
1544
Hazard
1655
Header
1485
Add background image
1492
Add links
1500
Change header at each chapter
1495, 1496
Change separators
1489
Changing the heading
1505
Language
1507
Only keep chapter title
1490
Style text
1491
Styling
1501
Underlined header
1504, 1570, 1571
XPath
1502
Header and Footer
Chapter first page
1487
Font
1486
Position
1488
Size
1486
Hyphenation
1594
Alter a dictionary
1596
Avoid line breaks at hyphens
1598
Define for a specific word
1598
Dictionaries
1595
Disable for an element
1597
Enable for an element
1597
Entire map
1597
Force line breaks at hyphens
1598
Install a dictionary
1595
i18n
1702
Images
1628
Borders
1633
Centering
1632
Control image size
1631, 1631
Resolution
1629
Rotate
1630
Side by side with text
1630, 1630
Size
1629
Wide
1630
Index
1578

Add a leader
1583
Change number format
1584
Change style and letters
1581
Inline Terms
1584
Style labels
1582
Table style layout
1585
Integration Server
Jenkins
1456
Links
1626
List of Figures
1569
List of Tables
1569
Localization
1702
Logging
1454
MathML equation customization
1619
MathML equation numbering
1620
Metadata
1529
Changing the keywords
1539
Changing the title property
1540
Cover page
1534
Custom
1533
Footer
1537
Header
1537
Index terms
1532
Key values
1541
Keywords
1532, 1539
Removing the title property
1540
Title property
1540, 1540
Use key value in CSS
1541
Modify
Strings
1703
Notes
1654
Numbering
1545, 1550
Reset page numbering
1555
Page breaks
1508
Add a blank page after a topic
1510
Avoid in lists and tables
1508
Enforce number of lines
1511
Force before or after topic
1509
Page size
Change setting for an element
1484
Changing
1483
Orientation
1484
Permissions
1705
Programming Elements

1649
Publication content
1662
Security
1705
Sign PDF
1706
Styling
1616
Table of contents
1556
Change header
1561
Display short description in TOC
1564
Display subtopics
1563
Display topic before TOC
1563
Increase depth
1559
Remove TOC entries
1563
Start on odd page
1562
Style
1560
Tables
1639
Centering
1642
Customize
Cells
1643
Columns
1643
Rows
1643
Layout
1639
Rotate
1640
Small images
1642
Split Cell
Borders
1645
Stripes
1645
Text bleeding
1641
Wide
1640
Zebra stripes
1645
Tasks
1658
Titles
Change prefix
1617
Layout
1616
Remove prefixes
1617
Separate page
1618
Tracked changes
1606
Style footnotes
1611
Styling
1610
Trademarks
1660
Translation
1702
Troubleshooting
Abbreviated-form
1710
Access Denied
1707
Cell borders missing
1709
Date formatting issues

1710
Disappearing lines
1709
Error Chunk Copy-To
1713
Error parsing
1708
Failed to run pipeline error
1709
Glossentry
1710
Glossgroup
1710
Glyph not available in font
1707
Highlights span off page
1712
I/O exception
1708
PDF file is damaged
1707
Unexpected Page Break
1712
Unknown host
1708
Videos
1638
Key Reference
1638
DITA Map to CHM (Compiled Help) transformation
1092
DITA Map to MS Office Word transformation
1091
DITA Map to PDF transformation
1090
DITA Map to WebHelp Responsive transformation
1073
DITA Map to Zendesk transformation
1094
DITA Map transformations
1072
DITA Maps
Transformation scenarios
1072
DITA new topics preferences
202
DITA PDF - based on HTML5 & CSS
Add
Strings
1704
Changing the cover page
1662
Changing the page size
1662
Changing the title
1661
Changing the TOC depth
1661
Command Line
1455
Comments
Styling
1610
Cover page
SVG templates
1523
Custom transformation parameters
1660
Customization CSS
1469
Customize
Strings
1703
Debugging
1471
Figures
Figure numbering
1634
i18n
1702
Integration Server
Jenkins
1456
Localization

1702
Modify
Strings
1703
Permissions
1705
Security
1705
Single topic
Cover page
1523
Tracked changes
Styling
1610
Translation
1702
DITA preferences
200
DITA Prolog Updater Addon - Automatically update the created/modification dates and author name in the DITA map and topic prologs
1921
DITA publishing preferences
203
DITA References View Addon - See all outgoing and incoming references for the current DITA topic
1921
DITA to Word
1091
DITA topic
Transformation scenarios
1098
DITA Topic document type
1054
DITA topic transformations
1098
DITA Translation Package Builder Addon - Build translation packages containing only the modified DITA content to send to translation agencies
1921
DITA-OT preferences
200
DITA-OT transformation scenario
1127
Advanced tab
1136
Feedback tab
1135
Filters tab
1136
FO Processor tab
1133
Output tab
1139
Parameters tab
1134
Templates tab
1128
DocBook 4 document type
1042
DocBook 5 document type
1047
DocBook 5.1 document type
1047
DocBook Assembly
1051
DocBook Checker Addon - Reports issues such as broken internal and external links, problems with images, or profiling conditions that conflict with profiling preferences in Docbook 4 or 5 XML documents
1921
DocBook olink
1043, 1048
DocBook Targetset Map document type
1053
DocBook to DITA transformation
1099
DocBook to EPUB transformation
1100
DocBook to PDF output customization
1724
DocBook to PDF transformation
1100

DocBook Topic document type
1052
DocBook transformations
1099
Document plugin extension
1884
Document template preferences
122
Document templates
Creating
307
Customizing
309
Sharing
313
Document type association preferences
93
Document type configuration dialog box
95
Association rules tab
97
Author tab
100
Actions subtab
102
Content Completion subtab
115
Contextual Menu subtab
112
CSS subtab
101
Menu subtab
112
Toolbar subtab
115
Catalogs tab
119
Classpath tab
99
Extensions tab
122
Schema tab
99
Templates tab
118
Transformation tab
120
Validation tab
121
Document Types
1042, 1069
Documents with long lines
399
Drag and Drop in Grid Mode
509
Drag and Drop in Text Mode
455
DTD Entities
573, 573
Duplicate transformation scenario
1209

E

Edit mode preferences
126
Edit validation scenario
536
Editing actions in Grid Mode
507
Editing Modes
279
Design
280, 689
Grid
279
Text
279
Editing publishing templates
1324, 1467
Editing XML markup in Text Mode
449
Editing XSLT
Compile XSL Stylesheet for Saxon tool
672, 1968
Convert between JSON and XML from XSLT

674
Editor perspective
272
Editor preferences
124
Editor variables
252
Custom editor variables
261
Elements view in Text mode
472
Emmet Abbreviations Addon - Provides the means for high-speed coding and editing in Text mode or Author mode via a content assistance mechanism.
1921
Encoding preferences
123
Entities view
472
EPUB document type
1064
Error: 1248 WARN org.apache.fop.apps.FOUserAgent
2218
Error: Anttask - Error rendering fo file
2218
Error: Cannot find files.log file
2213
Error: java.io.FileNotFoundException
2218
Error: Navigation to the web page was canceled
2228
Error: OutOfMemory
2205, 2206
Error: OutofMemoryError
2228
Error: stack overflow
2206
Error: Startup crash - Fault Module Name nvoglv32.dll
2213
Executing SQF in Other Documents
999
eXist database
Connection wizard
1761
Manual configuration
1762
eXist database connection
1761
eXist database contextual menu actions
1763
Export actions
102
Export color themes
84
Export Global Options
242
Export Global Transformation Scenarios
251
Export Global Validation Scenarios
251
Export Layout
243, 285
Export to Excel
513
Exporting Markdown documents
1017
Extending Oxygen with plugins
1857
External DITA-OT plugin extension
1872
External tool configuration
221
External tool preferences
221
External Tools
2197

F

Facets view in Design mode
692
Feedback Connector Addon - Connect to the Oxygen Feedback server to see all the comments added in your WebHelp output

1921
File comparison appearance preferences
217
File comparison preferences
215
File properties
328
File related actions in Text Mode
490
File type preferences
226
Find actions
376
Find All Elements action
374
Find All Elements dialog box
374
Find and invoke action
376
Find/Replace action
364
Find/Replace dialog box
364
Find/Replace in Files action
368
Find/Replace in multiple files
368
Finding/Replacing text
Find All Elements dialog box
374
Find/Replace dialog box
364
Find/Replace in multiple files
368
Navigating Find/Replace matches
378
Quick Find toolbar
378
Flatten Schema tool
767, 1966
Floating license servers
Server signature does not match
70, 2227
FO processor preferences
192
Folding in Text Mode
454
Folding in YAML documents
930
Font preferences
88
Font size configuration in Text mode
446
FOP Error
2218, 2218
Format and Indent Files tool
486, 1984
Format and Indent in Text Mode
481
Format and indent in YAML documents
930
Format/Indent multiple files
486, 1984
Formatting preferences
136
Formatting Schematron Quick Fix Content
998
Framework directory locations
94
Frameworks
1042, 1069

G

General plugin extension
1881
Generate Documentation tools
1985
Generate HTML report for directory comparison tool
434, 2053
Generate Java Classes from XML Schema Addon - Generate Java classes from an XML Schema (XSD) file
1921
Generate Java classes from XSD

769, 1967
Generate Java classes from XSD tool
769, 1967
Generate JSON Schema documentation
917, 1998
Generate JSON Schema tool
915, 1972
Generate OpenAPI documentation
1998
Generate OpenAPI Documentation Addon - Generate documentation for OpenAPI components in HTML format, including annotations and cross references
1921
Generate Sample JSON Files tool
917, 1970
Generate Sample XML Files tool
748, 1954
Generate WSDL Documentation tool
808, 1995
Generate XML Schema Documentation tool
753, 1986
Customize PDF output for DITA
763
Customize PDF output for DocBook
762
Output formats
757
Generate XSLT Stylesheet Documentation tool
665, 1989
Custom format
671
HTML format
668
Generating JSON Schema Documentation Addon - Generate documentation for a JSON Schema file in HTML format
1921
Generic JDBC database connection
1777
Getting familiar with the interface
24, 283
Getting Started
23
Getting Started with Oxygen
Help
2200
Interface
24, 283
Resources to help you with Oxygen
26
Shortcut keys
293
Git Client Addon- Checkout and Work with a Git Repository
1921
Global Options
240
Global preferences
81
Graphics and media
1350
Grid editing mode
279, 505
Bidirectional text
512
Content Completion Assistant
512
Copy/Paste
510
Drag and Drop
509
Editing actions
507
Export to Excel
513
Special characters
512
Grid editing modes
Layout
505
Navigation
506
Grid mode preferences

130

H

Help
Randomize XML text content
2203
Support related resources
2200
Using the Help menu
2200
Help resources
1439
Hex Viewer tool
2009
Hide file tabs
325
Highlight ID occurrences in Text Mode
492
Highlights
437
HTML documents
Content completion
1012
Editing features
1010
Folding
1014
Minification
1014
Minifying
1014
Outline view
1014
Syntax highlighting
1013
Validation
1011
HTTP authentication schemes
324
HTTP connection settings
232
HTTP floating license server
Allowed users list
70
Replacing
68
Upgrading
69
HTTP license server
58
Management and Statistics
65
HTTPS troubleshooting
322
Huge file editor
399
Huge files
399

I

IBM DB2 database connection
1784
IBM DB2 database contextual menu actions
1788
ICU4J Library Add-on - Adds provides the entire ICU4J JAR library used for sorting and collations with the Saxon XSLT processor
1921
Ignore validation problems
549
Ignored validation problems preferences
163
Image maps
1635
Image not found
1635
Image preview pane
397
Import color themes
84
Import data dynamically
1822
Import Global Options
242
Import Global Transformation Scenarios
251

Import Global Validation Scenarios
251
Import preferences
197
Import text files
1815
Importing data
1815
Importing data from a database
1820
Importing data from Excel
1817
Importing data from HTML files
1822
Importing Data from text files
1815
Increase memory
271, 1443
Indenting Schematron Quick Fix Content
998
Information view
437
Information view preferences
235
Insert File in Text mode
490
Install DITA-OT plugins
1096
Install new add-ons
73
Installed fonts do not appear in Fonts preferences
2221
Installing license server all-platform distribution
61, 62
Installing license server on Windows
61
Installing multiple license server instances
64
Installing Oxygen
33
Group deployment
49
Linux Installation
40
Linux/Unix Server Installation
47
macOS Installation
38
Transfer license key
53
Upgrading
71
Windows Installation
34
Windows Server Installation
45
Integrate DITA-OT plugins
1096
Integrate Schematron Quick Fixes in a framework
1000
Integrate Schematron rules in a framework
963
Integrating external tools
2197
Introduction
22

J

JATS document type
1063
Java system properties
261
Java VM parameters
267
JavaScript documents
940
Content Completion
943
Editing features
940
Outline view
945
Syntax highlighting
944
Validation

943
JDBC-ODBC database connection
1778
JSON documents
850
Associate JSON schema in a framework configuration
869
Associate schema
865
Associate schema directly in JSON documents
868
Associate schema through a validation scenario
865
Content completion
863
Contextual menu actions
880
Flatten schema
922
Folding
870
JSON to XML converter
873, 1975
JSON to YAML converter
879, 933, 1974
Navigating references
852
Outline view
870
Presenting validation errors
857
Schema annotations
864
Syntax highlighting
870, 921
Validating JSON schema
920
Validation
855
Against a schema
856
Automatic validation
856
Check Well-formedness
855
Manual validation
857
Resolving references to remote schemas
863
Sharing scenarios
862
Validation scenarios
859
XML to JSON converter
876, 1978
XSD to JSON Schema converter
919, 1982
YAML to JSON converter
879, 932, 1974
JSON Lines documents
Content completion
922
Editing features
922
Validation
922
JSON schema
856, 920
JSON Schema
893
JSON Schema Converter tool
919, 1982
JSON Schema Design mode
280, 894
Components
903, 903
Contextual menu actions
899
Validation
914
JSON Schema Design Mode
Editing actions
898

JSON Schema diagram editor
894
Components
903, 903
Contextual menu actions
899
Editing actions
898
Validation
914
JSON Schema Diagram Editor
Navigation
896
Palette view
896
JSON Schemas
Design mode editing
Navigation
896
Palette view
896
JSON to XML tool
873, 1975
JSON to YAML tool
879, 933, 1974
JSON transformation scenario
FO Processor tab
1148, 1182
Output tab
1149, 1183
XQuery tab
1151, 1199
XSLT tab
1145, 1179
JSON transformation with XQuery
1151
JSON transformation with XSLT
1144
JSON validation scenarios
859
JSON-LD documents
Content completion
1068
Editing features
1068
Validation
1068
JSON5 documents
Editing features
922
jTEI
document type
1062

L

Large documents
397
Large File Viewer tool
2007
Large files
398
Layout configuration
243, 285
Layout preferences
90
Learn document structure
869
LESS stylesheets
Compile to CSS
823
Content completion
822
Editing features
821
Syntax highlighting
822
Validation
822
Licenses
51
Licensing
50, 51
Licensing Oxygen
Floating license
Reserve floating license

57
Floating licenses
55
Register floating licenses for multiple users
57
Release a floating license
56
Request a floating license
license server
55
Live Tutorials Addon - Create and run tutorials for learning how to edit a certain XML vocabulary with Oxygen. The application will notify you when each mission in a tutorial is successfully fulfilled
1921
Load Layout
243, 285
Localizing SQF messages
1000
Localizing the user interface
266
Localizing XML refactoring operations
614, 1953
Lock Handler plugin extension
1874
Lock/Unlock XML tags in Text Mode
489
Locking a floating license
57

M

Mac Function Keys
2226
Mac Touch Bar
2226
Main Files
350
Add files
353
Detecting
352
Enabling
351
Overview
351
Transformation
354
Validation
354
Manage add-ons
73
Manage highlighted content in Text Mode
487
Managing license server
65
Manual validation actions
517
Manual validation in JSON
857
Markdown documents
1016
Actions in Markdown Editor
1019
Creating Markdown documents
1018
Markdown Editor
1017
Rules and specifications
1027
Syntax highlighting
1026
Validation
1026
Markdown Editor
1017
Markdown preferences
204
MarkLogic database connection
1766
MarkLogic database contextual menu actions
1773
MarkLogic debugging
1770, 1799
MarkLogic for the developer
1769

Markup transparency in Text Mode
488
MathML equations in HTML output in Author mode
1403
MathML equations in WebHelp output
1354
Memory availability
271
Memory issues
1443
Menu Shortcut Keys
223
Merge directories with change tracking tool
2056
Merge documents with change tracking tool
2039
Message preferences
235
Microsoft SQL Server database connection
1748
Microsoft SQL Server database contextual menu actions
1751
Model view
470
Modular XML files
567
Move file tabs
325
Move XML resources
573
MS Azure authentication
1813
MySQL database connection
1775

N

Named-User licenses
51
Navigating in Text Mode
442
Network connection preferences
230
New document from templates
307
New Document Wizard
299
New from Templates Wizard
307
New project
28, 330
NISO Journal Article Tag Suite document type
1063
Non-XML files
329
NTLM
324
NVDL schemas
842
- Component Dependencies view
847
- Content completion
845
- Diagram Editor
842
  - Contextual menu actions
844
  - Full Model view
843
  - Introduction
842
  - Logical Model view
844
- Outline view
847
- Refactoring actions
848
- Search actions
848
- Syntax highlighting
846
- Validation
845

O

Olink element
1043, 1048
Open File at Cursor in Text mode
490
Open file at specific location
315
Open file at start-up
314
Open file in system application
314
Open preferences
133
Open Redirect plugin extension
1875
Open remote document
318
Open URL dialog box
318
Open/Find Resource
In content
360
In file paths
363
In reviews
363
Open/Find Resource dialog box
358
Open/Find resource preferences
227
Open/Find Resource view
354
OpenAPI document type
1065
OpenAPI test scenario documents
Content completion
1066
Editing features
1066
Validation
1066
OpenAPI tester
1983
OpenAPI Tester Addon - Testing tool for OpenAPI files
1921
Opening file
313
Option Page Group plugin extension
1876
Option Page plugin extension
1875
Options Menu
Preferences
79
Oracle database connection
1752
Oracle database contextual menu actions
1755
Out of memory
1443
Out Of Memory error
192, 2007
Outline view
465
Outline view in Design mode
735
Outline view preferences
235
OutOfMemory
1443
oxy:allows-child-element function
107
oxy:allows-global-element function
109
oxy:current-selected-element function
110
oxy:is-required-element function
111, 111
oxy:platform function
111
oxy:selected-elements function
110

Oxygen AI Positron Addon - Provides support for helping writers generate content by using the Oxygen AI Positron service
1921
Oxygen SDK
1857
Automated tests
1892
CMS integration
1885
Configuration
1857
Custom Protocol
1890
Debugging a plugin
1895
IntelliJ
1895
Debugging an SDK extension
1896
Deploying plugins as add-ons
1891
Disable a plugin
1897
Installation
1860
Testing plugins and Java extensions
1892
Types of Extensions
1861
Oxygen Styles Basket
1309, 1464

P

Palette view in Design mode
691
Palette view in JSON Design mode
896
PDF
CSS
2220
Troubleshooting
2220
PDF output
1438
PDF Output Customization
DocBook to PDF output
1724
PDF output using Calabash XProc processor
1189
PDF Processors
1440
Performance preferences
133
Perspectives
272
Database
276
Editor
272
XQuery Debugger
275
XSLT Debugger
274
Plugin extensions
1861
Additional Framework
1869
Additional XProc engine
1869
Components Validation
1870
Contribute additional languages
1872
Contribute external DITA-OT
1872
Custom Protocol
1873
Document
1884
General
1881
Lock Handler
1874
Open Redirect

1875
Option Page
1875
Option Page Group
1876
Refactoring Operations
1879
Resource Locking
1876
Saxon XQuery Transformer
1881
Saxon XSLT Transformer
1880
Selection
1882
Trusted Hosts
1868
URL Stream Handler
1877
validator
1880
Workspace Access
1861
Adding custom views
1864
Workspace Access (JavaScript-based)
1864
XQuery Transformer
1880
XSLT Transformer
1879
Plugin preferences
221
Plugins
Automated tests
1892
CMS integration
1885
Configuration
1857
Custom Protocol
1890
Debugging
1895
IntelliJ
1895
Debugging an SDK extension
1896
Deploying plugins as add-ons
1891
Disable a plugin
1897
Installation
1860
Testing
1892
Types of Extensions
1861
PostgreSQL database connection
1757
PostgreSQL database contextual menu actions
1760
Preferences
79
Add-ons
92
Ant
197
Appearance
84
Application Layout
90
Archive
219
Code Templates
151
Colors
86
Content Completion
144
Content Completion annotations
150
CSS formatting
143

CSS Processors
196
CSS Validator
168
Custom Editor Variables
229
Custom Engines
191
Custom Validation Engines
161
Data Sources
205
Debugger
188
Diagram
128
Directories Comparison
218
Directories Comparison Appearance
219
DITA
200
DITA Logging
203
DITA New topics
202
DITA publishing
203
Document Templates
122
Document Type Association
93
Document type configuration dialog box
95
Document Type locations
94
Document Validation
160
Edit Modes
126
Editor
124
Encoding
123
External Tools
221
File Types
226
Files Comparison
215
Files Comparison Appearance
217
FO Processor
192
Fonts
88
Format
136
Global
81
Grid mode
130
HTTP(S)/WebDAV
232
Ignored validation problems
163
Import
197
JavaScript Content Completion
148
JavaScript formatting
143
JSON Content Completion
149
JSON formatting
144
Mark Occurrences
159
Markdown
204
Menu Shortcut Keys
223
Messages
235

Network Connection Settings
230
Open
133
Open/Find Resource
227
Plugins
221
Print
167
Project Level Settings
92
Proxy
230
Relax NG parser
173
Sample XML Files Generator
176
Save
135
Saxon-HE/PE/EE XQuery
186
Saxon-HE/PE/EE XQuery Advanced
187
Saxon-HE/PE/EE XSLT
181
Saxon-HE/PE/EE XSLT Advanced
183
Saxon6 XSLT
180
Schema Design mode
131
Schematron parser
174
SFTP
233
Spell Check
163
Spell Check Dictionaries
166
SSH
234
SVN
210
SVN Diff
213
SVN Messages
214
SVN Working Copy
212
Syntax Highlight
158
Text mode
127
Trusted Hosts
233
Views
235
Whitespaces
142
XML
168
XML Catalog
168
XML formatting
139
XML Parser
171
XML Refactoring
200
XML Schema parser
172
XML Signing Certificates
199
XML Structure Outline
235
XPath
190
XPath Content Completion
147
XPath formatting
143
XProc
178

XQuery
185
XQuery formatting
142
XQuery Profiler
189
XSD Content Completion
148
XSLT
180
XSLT Content Completion
146
XSLT Profiler
189
XSLT/XQuery
179
XSLTProc
184
YAML Content Completion
149
Presenting validation errors in JSON
857
Print documents
438
Print preferences
167
Print Preview dialog box
438
Printing
438
Problems and solutions
2207
Process overview
1440
Project Level Settings preferences
92
Project Options
240
Project view
334
Project view preferences
235
Projects
330
Adding items to projects
28, 330
Batch transformations
346
Batch validation
346
Creating new projects
28, 330
Main files
350
Add files
353
Detecting
352
Enabling
351
Overview
351
Transformation
354
Validation
354
Managing resources
334
Move resources
344
Project view
334
Rename resources
344
Sharing
347
Protect PDF
1705
Proxy preferences
230
Publishing
1070

Q

Quick Assist feature in Text Mode

491
Quick Find toolbar
378
Quick fix support in XML documents
551

R

RDP performance issues
2207
Read-only files
400
Rectangular Selection feature
455
Refactoring Operations plugin extension
1879
Refactoring XML Documents
578, 1922
Referenced/Dependent Resources view for XML documents
570
Registering floating licenses for multiple users
57
Registering named-user license
52
Registering subscription license
53
Regular expressions syntax
379
Relax NG schemas
823
Component Dependencies View
837
Content completion
829
Custom datatype library
842
Diagram Editor
824
Contextual menu actions
827
Full Model view
825
Introduction
824
Logical Model view
826
Symbols
826
Editing in Main Files context
824
Moving resources
836
Outline view
831
Quick Assist feature
840
Refactoring actions
838
Referenced/Dependent Resources view
834
Renaming resources
836
Search actions
838
Syntax highlighting
830
Validation
828
Release a floating license
56
Remote Desktop performance issues
2207
Rename XML resources
573
Request a floating license from a license server
55
Reserving a floating license
57
Reset Global Options
242
Reset Layout
243, 285
Resolve schemas through XML catalogs
566
Resource Locking plugin extension

1876
Restrict access to PDF content
1705
Restricting Schematron Quick Fix operations
997
Results view
474
Make persistent copy of results
479
Run OpenAPI test scenario tool
1983

S

S1000D document type
1056
Sample XML File Generator preferences
176
Save file
316
Save preferences
135
Save remote document
318
Save template as button
1131
Saxon 6 XSLT preferences
180
Saxon HE/PE/EE advanced XQuery preferences
187
Saxon HE/PE/EE advanced XSLT preferences
183
Saxon HE/PE/EE XQuery preferences
186
Saxon HE/PE/EE XSLT preferences
181
Saxon XQuery Transformer plugin extension
1881
Saxon XSLT and XQuery Transformer Addon - Install various versions of the Saxon XSLT/XQuery processor for use when transforming
1921
Saxon XSLT Transformer plugin extension
1880
Schema annotations in JSON
864
Schema annotations in Text Mode
460
Schema Design mode
280, 689
Schema Diagram Editor
280, 689
Schematron
Editing features
948
Editing in Main Files context
962
Examples
950, 978
Integrating in a framework
963
Sharing
963
Unit test (XSpec)
976
Validation
Against Schematron
962
Schematron examples
950, 978
Schematron Quick Fix examples
950, 978
Schematron Quick Fix Operations
Add
992
Delete
992
Replace
992
String Replace
992
Schematron Quick Fixes
553, 978
Content Completion
1003
Customizing

991
Defining
991
Embedded
1005
Examples
950, 978
Executing SQF in Other Documents
999
Formatting and Indenting
998
Highlight occurrences
1003
Integrating in a framework
1000
Localization
1000
Multiple similar quick fixes
999
Refactoring actions
1003
Restricting operations
997
Search actions
1003
Sharing
1000
SQF Operations
992
Use-When Condition
997
User Entry operation
997
Validation
1002
Schematron schemas
Content completion
965
Syntax highlighting
966
Schematron Schemas
Embedded
548, 967
Highlight occurrences
972
Moving resources
972
Outline view
968
Quick Assist feature
975
Refactoring actions
973
Referenced/Dependent Resources view
970
Renaming resources
972
Search actions
973
Validation
964
Scratch Buffer
400
Search actions for IDs
567
Search dialog box
364
Search multiple files
368
Search preferences
227
Searching documents
354
Open/Find Resource
In content
360
In file paths
363
In reviews
363
Open/Find Resource dialog box
358
Open/Find Resource view
354

Security
1868
Security permissions
1705
Selecting content in Text Mode
455
Selection plugin extension
1882
Server signature does not match
70, 2227
Set indent to zero in Text Mode
485
SFTP connection settings
233
Share transformation scenarios
1216
Share validation scenarios
546
SharePoint Browser view
1807
SharePoint contextual menu actions
1809
SharePoint database connection
1802
Sharing document templates
313
Sharing JSON validation scenarios
862
Sharing project level options
241
Sharing project options file
347
Minimize differences between versions
350
Sharing publishing templates
1326, 1469
Sharing Schematron Quick Fixes
1000
Sharing Schematron rules
963
Sharing settings
241
Sharing transformation scenarios
347
Sharing validation scenarios
347
Sharing XML refactoring operations
613, 1952
Shortcut actions in Text Mode
446
Shortcut key configuration
223
Shortcut Keys
293
Show change tracking/comments in DocBook PDF
1101
Sign files tool
622, 2000
Signing XML document preferences
199
Simple text editor
329
Single sign-on
324
Smart Editing in Text Mode
445
Special character preferences
133
Special characters
390
Character Map
394
Fallback Font Support
393
Inserting symbols
394
Unrecognized characters
392
Unsupported characters
392
Special characters in Grid mode
512
Special characters in Text mode
489

Spell check preferences
163
Spell checking
380
Add dictionaries
383
Add term lists
385
Custom list of terms
385
Customizing dictionaries and term lists
382
Dictionaries and term lists
381
Forbidden words
385
Ignored words
388
Learned words
387
Multiple files
389
Replace dictionaries
386
Spell Checking
Automatic spell check
388
SQF
553, 978
Content Completion
1003
Customizing
991
Defining
991
Embedded
1005
Examples
950, 978
Executing SQF in Other Documents
999
Formatting and Indenting
998
Highlight occurrences
1003
Integrating in a framework
1000
Localization
1000
Multiple similar quick fixes
999
Operations
992
Refactoring actions
1003
Restricting operations
997
Search actions
1003
Sharing
1000
Use-When Condition
997
User Entry operation
997
Validation
1002
SQF attribute: use-for-each
999
SQF examples
950, 978
SQF multi-lingual support
1000
SQL transformation scenario
1206
SSH connection settings
234
StackOverflowException error
162
Startup parameter
Application launcher parameters
267
Command-line script parameters
270

Custom startup parameters file
270
Startup parameters
267
Status information
437
Storing global options
240
Storing project level options
240
Storing XML refactoring operations
613, 1952
Subscription licenses
53
Supported document types
1040
Supported frameworks
OpenAPI
1065
Supported processors for XSLT/XQuery debugging
1856
SVG
Syntax Diagrams
1637
SVG files
1006
SVG Viewer
1007, 2010
SVG viewer in Results panel
1009
SVG Viewer in Results panel
1009
SVG Viewer tool
1007, 2010
SVN Client
2060
Authentication
2078
Branches/Tags
2111
Create branch/tag
2111
Exporting resources
2147
Importing resources
2146
Manage resources
2150
Merging
2113
Patching
2132
Relocate working copy
2132
Switch repository location
2130
Checking out a working copy
2082
Define a working copy
2081
Entering local paths and URLs
2192
History dialog box
2084
Manage repository locations
2077
Menus
2061
Preferences
2192
Properties
2110
Request history for a resource
2110
Request status information for a resource
2109
Resource History view
Directory Change Set view
2177
Revision Graph
2188
Share projects
2080
Sparse checkouts

2151
Status bar
2076
Synchronize with the SVN repository
2093
Conflicts
2096
Integrating bug tracking tools
2108
Send changes to repository
2104
Update working copy
2103
View differences
2094
Technical issues
2194
Toolbar
2074
Use an existing working copy
2085
Views
2061
Annotations view
2179, 2187
Compare Images view
2185
Compare view
2181
Dynamic Help view
2187
Editor view
2178
History view
2172
Image Preview panel
2185
Properties view
2185
Repositories view
2152
Working Copy view
2157
Working copy resource management
2086
Add resources
2086
Copy resources
2090
Delete resources
2089
Edit files
2086
Ignore resources
2088
Lock/Unlock resources
2091
Move resources
2091
Rename resources
2091
SVN message preferences
214
SVN preferences
210
Swagger
1065
Switch file tabs
325
Syntax highlight preferences
158
Syntax highlighting in Text Mode
480
Syntax highlights in codeblocks
1341
Syntax highlights in Text mode
480
System properties
261

T

Table Explorer view
1744
Tags transparency selector
488

TEI ODD document type
1061
TEI P5 document type
1059
Templates tab
1128
Terminology Checker Addon - Report terminology problems using both Vale and custom rule files.
1921
Testing plugins and Java extensions
1892
Text editing mode
279, 442
Attributes view
468
Bidirectional text
489
Content completion
457
Contextual menu actions
492
Drag and Drop
455
Editing XML markup
449
Elements view
472
Entities view
472
File related actions
490
Folding
454
Format and indent
481
Format and indent multiple files
486, 1984
Highlight ID occurrences
492
Lock/Unlock XML tags
489
Manage highlighted content
487
Markup transparency
488
Model view
470
Navigation
442
Rectangular selection
455
Schema annotations
460
Selecting content
455
Set indent to zero
485
Shortcut actions
446
Smart Editing
445
Special characters
489
Syntax highlighting
480
Syntax highlights
480
Validation errors
519
Views
464
Text invisible issue
2231
Text mode preferences
127
Three-way directory comparison and merge tool
427, 2046
Three-way file comparison
402, 2012
Toolbar configuration
249, 291
Touch Bar on Mac
2226

Trackpad scroll function doesn't work on Lenovo Thinkpad
2230
Transform DITA document to MS Word
1091
Transformation parameters
1444
Transformation Scenarios
1070
Ant
1140
Apply transformation scenarios
1210
Batch transformation
1215
Built-in transformation scenarios
1071
Configure
1207
Configure Transformation Scenario dialog box
1210
Configure XSLT processor extension paths
1117, 1174
Custom XSLT processor
1116, 1173
Debugging PDF transformations
1723
DITA map
1072
DITA Map Metrics Report
1093
DITA Map PDF - based on HTML5 & CSS
1087
DITA Map to CHM (Compiled Help
1092
DITA Map to MS Office Word
1091
DITA Map to PDF
1090
DITA Map to WebHelp Responsive
1073
DITA Map to Zendesk
1094
DITA topic
1098
DITA-OT
1127
DocBook
1099
DocBook to DITA
1099
DocBook to EPUB
1100
DocBook to PDF
1100
Duplicate
1209
Integrate DITA-OT plugins
1096
Integrate external XProc engine
1189
JSON with XQuery
1151
JSON with XSLT
1144
New
1101
Sharing
1216
Show Change Tracking and Comments
1101
SQL
1206
Supported XSLT processors
1113, 1171
XML to PDF with CSS
1125
XML with XQuery
1117
XML with XSLT
1102
XProc
1184
XQuery

1190, 1198
XSLT
1159, 1178
Transformation Scenarios view
1217
Transformation types
1207
Transforming Documents
1070
Translator Helper Addon - Use Google Translate or the DeepL Translator to translate content selected in the Author visual editing mode
1921
Tree editor preferences
235
Tree Editor tool
2012
Trusted hosts connection settings
233
Two-way file comparison
402, 2012

U

Unicode support
391
Uninstalling Oxygen
78
Upgrading Oxygen
71
Uppercase plugin
1882
URL Stream Handler plugin extension
1877
Use-When SQF Condition
997
User Entry SQF operation
997
Using a floating license offline
57
UTF-8 BOM handling
123

V

Vale Linter for Markdown and HTML Validation Addon - Run the Vale linter over the currently edited Markdown or HTML file
1921
Validating JSON Documents
Automatic validation
856
Check Well-formedness
855
Manual validation
857
Resolving references to remote schemas
863
Sharing scenarios
862
Validating JSON Documents against a schema
856
Validating JSON schema
920
Validating XML Documents
514
Against a schema
516
Against a schema with embedded Schematron
548, 967
Against Schematron
962
Automatic validation
517
Check Well-formedness
515
Create new validation scenarios
526
Custom validators
522
Customizing error messages
521
Edit validation scenarios
536
Manual validation
517
Resolving references to remote schemas

546
Sharing scenarios
546
Text Mode
519
Validating XSLT that call Java extensions
629
Validation engine 'StackOverflowException' error
162
Validation errors in Text mode
519
Validation preferences
160
validator plugin extension
1880
Verify Signature tool
625, 2003

W

WebDAV connection
1789
WebDAV connection settings
232
WebDAV contextual menu actions
1789
WebDAV over HTTPS
322
WebHelp feature matrix
1221
WebHelp Responsive
1222
Add link to PDF
1404
Adding audio objects
1353
Adding favicon
1352
Adding logo
1351
Adding publishing templates to gallery
1324
Adding videos
1353
Adding welcome message
1346
Built-in templates
1269
Changing scoring values in search results
1355
Comments section
1306
Context-sensitive help
1236
Converting old publishing templates
1327
Converting publishing templates to version 22
1330
Converting publishing templates to version 24
1330
Converting publishing templates to version 24.1
1329
Converting publishing templates to version 25
1328
Converting publishing templates to version 27.1
1328
Converting publishing templates to version 28
1327
Converting version 20 publishing templates to version 21
1331
Copy resources to output directory
1398
Creating publishing templates
1309, 1464
CSS styling
1312, 1332
Custom search engine
1368
Custom search filter
1357
Custom templates
1269
Customizing main page layout
1344
Customizing output with CSS

1312, 1332
Customizing output with HTML content
1334
Customizing the footer
1347
Customizing the menu
1344
Customizing the tiles
1348
Dark mode
1312
Disable caching
1403
Edit link to launch Web Author
1400
Editing publishing templates
1324, 1467
Excluding topics from search results
1358
Facebook Like button
1376
Flagging DITA content
1401
Google Analytics
1380
Google Search
1363
Index terms layout page
1232
Inserting HTML
1334
Local fonts
1415
Localizing interface
1373
Main layout page
1223
Navigation links
1343
Optimizing Japanese content indexing
1356
Optimizing search results
1371
Ordered lists
layout
1339
styles
1339
PDF link
1404
Previous/Next arrows
1343
Publishing template descriptor file
1271
Publishing template HTML fragments
1278
Publishing template HTML page layout files
1287
Publishing template package
1271
Publishing template resources
1274
Publishing template transformation parameters
1276
Publishing template XSLT extensions
1277
Publishing templates could not be loaded
1326
Publishing templates troubleshooting
1326
Right-to-left languages
1375
Search features
1233
Search layout page
1229
Search rules
1233
searchQuery parameter
1371
Sharing publishing templates
1326, 1469
Sharing templates
1269

Tabs
1418
Topic layout page
1226
Transformation parameters
1420
Tweet button
1378
Using publishing templates from a command line
1325
XSLT-Import extension point
1435
XSLT-import extension points
1384
XSLT-Parameter extension point
1435
WebHelp Responsive Templates tab
Save template as button
1131
Workspace Access (JavaScript-based) plugin extension
1864
Workspace Access plugin extension
1861
Writer Helper Addon - Helps technical writers review, find similar content and provides productivity tips
1921
WSDL documents
791
Component Dependencies view
802
Content completion
793
Editing in Main Files context
792
Generate documentation for WSDL documents
808, 1995
Custom format
813
HTML format
812
Highlight occurrences
804
Moving resources
802
Outline view
795
Quick Assist feature
807
Refactoring actions
804
Referenced/Dependent Resources view
800
Renaming resources
802
Search actions
804
SOAP analyzer
813, 2003
Test remote files
816, 2005
Syntax highlighting
794
Validation
793
WSDL SOAP Analyzer tool
813, 2003
Test remote files
816, 2005

X

XHTML document type
1058
XHTML documents
Validation
1059
XInclude
573, 575
XInclude 1.1 features
575

XLIFF documents
Editing features
939
XML catalog preferences
168
XML Catalogs
564
XML documents
442
Associate schema
554
Associate schema directly in XML documents
561
Associate schema in a framework configuration
563
Associate schema in a validation scenario defined in framework
559
Associate schema through a validation scenario
555
Author Mode editing
Entities view
472
MathML equations in HTML output
1403
Model view
470
Code templates
462
DTD Entities
573, 573
Editing in Main Files context
567
Grid Mode editing
505
Content Completion Assistant
512
Copy/Paste
510
Drag and Drop
509
Editing actions
507
Export to Excel
513
Learn document structure
563
Moving resources
573
Outline view
465
Quick Assist feature
491
Quick fix support
551
Refactoring
578, 1922
Built-in operations
582, 1925
Custom operations
594, 1938
Localizing operations
614, 1953
Sharing operations
613, 1952
Storing operations
613, 1952
Referenced/Dependent Resources view
570
Renaming resources
573
Resolve schemas through XML catalogs
566
Search actions for IDs
567
Text Mode editing
442
Attributes view
468
Content completion
457
Contextual menu actions
492
Drag and Drop

455
Editing XML markup
449
Elements view
472
Entities view
472
File related actions
490
Folding
454
Format and indent
481
Highlight ID occurrences
492
Lock/Unlock XML tags
489
Manage highlighted content
487
Markup transparency
488
Model view
470
Navigation
442
Rectangular selection
455
Schema annotations
460
Selecting content
455
Set indent to zero
485
Shortcut actions
446
Smart Editing
445
Syntax highlighting
480
Validation
514
Against a schema
516
Against a schema with embedded Schematron
548, 967
Automatic validation
517
Check Well-formedness
515
Create new validation scenario
526
Custom validators
522
Customizing error messages
521
Edit validation scenario
536
Manual validation
517
Resolving references to remote schemas
546
Sharing scenarios
546
Text Mode
519
XInclude
573, 575
XML catalogs
564
XML documents without a schema
563
XML parser preferences
171
XML refactoring preferences
200
XML Refactoring tool
578, 1922
Built-in operations
582, 1925
Custom operations
594, 1938
Localizing operations
614, 1953

Sharing operations
613, 1952
Storing operations
613, 1952
XML Schema Design mode
280, 689
XML Schema Diagram Editor
280, 689
Attributes view
737
Components
703
Contextual menu actions
695
Edit namespaces
731
Editing actions
694
Facets view
692
Navigation
690
Outline view
735
Palette view
691
Validation
730
XML Schema Regular Expression Builder tool
769, 2006
XML Schemas
689
Attributes view
737
Component Dependencies view
742
Content completion
733
Convert DB Structure to XML Schema tool
766, 1965
Design mode editing
280, 689
Components
703
Grouping components
728
xs:all
720
xs:alternative
716
xs:any
721
xs:anyAttribute
723
xs:assert
726
xs:attribute
709
xs:attributeGroup
711
xs:choice
720
xs:complexType
711
xs:element
705
xs:field
726
xs:group
717
xs:import
719
xs:include
718
xs:key
724
xs:keyRef
725
xs:notation
720
xs:openContent
727
xs:override
719

xs:redefine
719
xs:schema
704
xs:selector
725
xs:sequence
720
xs:simpleType
714
xs:unique
723
Contextual menu actions
695
Edit namespaces
731
Editing actions
694
Facets view
692
Navigation
690
Palette view
691
Validation
730
Editing in Main Files context
731
Flatten Schema tool
767, 1966
Generate documentation for XML Schema
753, 1986
Customize PDF output for DITA
763
Customize PDF output for DocBook
762
Output formats
757
Generate Sample XML Files tool
748, 1954
Generate/Convert Schema tool
763, 1962
Highlight occurrences
744
Moving resources
741
Outline view
735
Quick Assist feature
746
Refactoring actions
744
Referenced/Dependent Resources view
739
Regular Expressions Builder
769, 2006
Renaming resources
741
Search actions
744
Set version
772
Syntax highlighting
734
Text mode editing
731
Validation
732
XML Schema 1.1 support
771
XML to JSON tool
876, 1978
XML to PDF with CSS transformation scenario
1125
CSS tab
1125
Output tab
1126
XML transformation with XQuery
1117
XML transformation with XSLT
1102
XPath Expressions
1726

Catalogs
1733
Prefix mapping
1734
Toolbar
1727
XPath Builder view
1729
XPath Results view
1732
XProc Scripts
Editing features
946
XProc transformation scenario
1184
Inputs tab
1186
Options tab
1188
Outputs tab
1187
Parameters tab
1186
XProc tab
1185
XQuery
774
Content completion
775
Folding
777
Format and Indent
777
Generate HTML documentation
785, 1993
Input view
783
Outline view
778
Sequence view
786
Syntax highlighting
776
Transformations
786
Sequence view
786
Updating XML documents
790
XQJ
1798
Unit test (XSpec)
790
Updating XML documents
790
Validation
774
XQuery Builder view
779, 889
XQuery Debugger perspective
275, 1826
Breakpoints
1849
Debugging Java extensions
1855
Identify expressions
1846
Information views
1832
Breakpoints view
1833
Context view
1834
Messages view
1836
Nodes/Values Set view
1842
Output Mapping Stack view
1838
Stack view
1837
Templates view
1841
Trace view

1840
Variables view
1843
XPath Watch view
1835
Layout
1827
Performance profiling
1850
Profiling
Hotspots view
1853
Invocation Tree view
1852
Steps in a typical debugging process
1845
Toolbar
1828
XQuery Documentation tool
785, 1993
XQuery preferences
185
XQuery Profiler preferences
189
XQuery transformation on JSON scenario
1198
XQuery transformation scenario
1190
FO Processor tab
1123, 1156, 1196, 1204
Output tab
1123, 1157, 1197, 1205
XQuery tab
1118, 1191
XQuery Transformer plugin extension
1880
XSD to JSON Schema Converter Addon - Converting an XML Schema file (XSD) to a JSON Schema file
1921
XSD to JSON Schema tool
919, 1982
XSL-FO processor
1174
XSLT
Component Dependencies view
649
Component documentation
652
Content completion
632
Content completion in XPath expressions
634
Editing features
626
Editing in Main Files context
627
Generate documentation for XSLT stylesheets
665, 1989
Custom format
671
HTML format
668
Highlight occurrences
651
Input view
644
Moving resources
648
Outline view
639
Quick Assist feature
660
Quick fix support
630
Refactoring actions
655
Referenced/Dependent Resources view
646
Renaming resources
648
Search declarations
651
Search references

651
Syntax highlighting
639
Text Value Templates
654, 654
Unit test (XSpec)
662
Validating XSLT that call Java extensions
629
Validating XSLT with custom engines
629
Validation
627
XPath Tooltip Helper
638
XSLT Debugger perspective
274, 1826
Breakpoints
1849
Debugging Java extensions
1855
Debugging XSLT that call Java extensions
1855
Identify expressions
1846
Information views
1832
Breakpoints view
1833
Context view
1834
Messages view
1836
Nodes/Values Set view
1842
Output Mapping Stack view
1838
Stack view
1837
Templates view
1841
Trace view
1840
Variables view
1843
XPath Watch view
1835
Layout
1827
Performance profiling
1850
Profiling
Hotspots view
1853
Invocation Tree view
1852
Steps in a typical debugging process
1845
Supported processors
1856
Toolbar
1828
XSLT preferences
180
XSLT Profiler preferences
189
XSLT transformation on JSON scenario
1178
XSLT transformation on XML scenario
1159
XSLT transformation scenario
FO Processor tab
1111, 1168
Output tab
1112, 1169
XSLT tab
1102, 1159
XSLT Transformer plugin extension
1879
XSLTProc preferences
184
XSpec Helper View Addon - Run XSpec test scenarios and view the results directly in the application

1921

Y

YAML documents
923
Associate schema directly in YAML documents
928
Content completion
929
Outline view
930
Syntax highlighting
930
Validation
924
Validation scenarios
925
YAML editor
923
YAML to JSON tool
879, 932, 1974
YAML validation scenarios
925

Copyright

Oxygen XML Developer User Manual

Syncro Soft SRL.

Copyright © 2002-2026 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 Developer End-User License Agreement, as well as information regarding support for this product, while under warranty, is available through the Oxygen XML Developer 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 Developer, see Oxygen XML Developer End-User License Agreement.

Documentation: For the most current versions of documentation, see the Oxygen XML Developer 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.