Contents

Chapter 1. Overview................................................................................................................. 10
  Resources.............................................................................................................................. 10
  Supported Processors........................................................................................................... 11
  Technical Details.................................................................................................................. 11
  Increasing Memory Allocation for Java ............................................................................... 14
  Transformation Parameters............................................................................................... 15
  Console Logging.................................................................................................................. 24
  License Key.......................................................................................................................... 24

Chapter 2. Generating PDF Output....................................................................................... 25
  Generating PDF from a Command Line ............................................................................. 25
  Generating PDF from an Integration Server....................................................................... 25

Chapter 3. Publishing Templates.......................................................................................... 28
  Template Package Contents for PDF.................................................................................. 29
  How to Create a Publishing Template............................................................................... 35
  How to Edit a Packed Publishing Template....................................................................... 38
  How to Use a Publishing Template in a PDF Transformation........................................... 39
  How to Share a Publishing Template.................................................................................. 40

Chapter 4. Customizing PDF Output Using CSS................................................................. 41
  Debugging the CSS.............................................................................................................. 42
    How to Speed up CSS Development and Debugging....................................................... 45
    How to Write XPath Expressions................................................................................... 46
    How to Debug XPath Expressions................................................................................ 46
  Default Page Definitions..................................................................................................... 48
  Page Size............................................................................................................................ 52
    Page Size - Built-in CSS rules....................................................................................... 53
    How to Change the Page Size....................................................................................... 53
    How to Change the Page Orientation.......................................................................... 54
    How to Change the Page Settings for a Specific Element............................................ 54
  Page Headers and Footers................................................................................................. 54
    Page Headers and Footers - Built-in CSS.................................................................... 55
    How to Position Text in the Headers and Footers....................................................... 55
How to Display Chapter's Headers on First Page.................................................................56
How to Change the Size of Headers and Footers.......................................................................57
How to Change the Font of the Headers and Footers...............................................................58
How to Change Header Text for Each Topic................................................................................58
How to Change Header Images for Each Chapter....................................................................60
How to Add a Multi-line Copyright Notice to the Footer..........................................................62
How to Add a Group of Topics to the Footer............................................................................63
How to Add a Background Image to the Header........................................................................64
How to Add a Link in Headers and Footers................................................................................65
How to Decorate the Header by Using a Background Image on the Entire Page......................66
How to Style a Part of the Text from the Header.........................................................................66
How to Simplify the Header (Keep Only the Chapter Title)..........................................................68
How to Change the Header Separators......................................................................................68
How to Change the Header Styling Depending on Page Side....................................................69
How to Use XPath Computed Data or Images in the Header or Footer....................................70
How to Add a Line Under the Header.......................................................................................72
How to Change the Headings Using a Parameter......................................................................73
How to Change the Headings depending on the Language.......................................................75
How to Display the Chapter and the Page Number in the Footer..............................................75

Page Breaks..............................................................................................................................76
Page Breaks - Built-in CSS.........................................................................................................76
How to Avoid Page Breaks in Lists and Tables..........................................................................76
How to Force a Page Break Before or After a Topic or Another Element.................................77
How to Add a Blank Page After a Topic....................................................................................78
How to Enforce a Number of Lines from Paragraphs that Continue in Next Page........................79
How to Avoid Page Breaks Between Top-Level Topics (Chapters)..........................................79

Cover (Title) Page......................................................................................................................79
Cover Page - XML Fragment.....................................................................................................79
Cover Page - Built-in CSS rules................................................................................................80
How to Add a Background Image for the Cover ........................................................................81
How to Change Styling of the Cover Page Title..........................................................................84
How to Add Text to the Cover Page...........................................................................................84
How to Place Cover on the Right or Left Side............................................................................85
Contents | iv

How to Add a Second Cover Page and Back Cover Page........................................... 86
How to Add a Specific Number of Empty Pages After the Cover Page.............................. 88
How to Add a Copyright Page after the Map Cover (Not for Bookmaps)... 89
How to Remove the Cover Page and TOC.................................................................... 90
How to Add a Cover in Single-Topic Publishing.......................................................... 91

Metadata....................................................................................................................... 91
Metadata - XML Fragment......................................................................................... 92
Metadata - Built-in CSS.......................................................................................... 94
How to Create a Searchable PDF.............................................................................. 94
How to Add the Publication Audience to the Custom PDF Metadata............................ 96
How to Show Metadata in the Cover Page................................................................. 97
How to Show Metadata in the Header or Footer......................................................... 100
How to Remove or Change the PDF Keywords......................................................... 100
How to Remove the PDF Publication Title Property.................................................. 101
How to Change the PDF Publication Title Property................................................... 101
How to Use Data Elements from the Map to Create Custom PDF Metadata.................. 102
How to Control the PDF Viewer................................................................................ 103

Front Matter and Back Matter....................................................................................... 104
Front Matter and Back Matter - XML Fragment........................................................ 104
Front Matter and Back Matter - Built-in CSS............................................................ 105
How to Remove Page Breaks Between Front Matter Child Topics............................... 105
How to Style the Front Matter and Back Matter Topics.............................................. 105

Numbering................................................................................................................... 107
Numbering - Built-in CSS......................................................................................... 107
Numbering - Input XML Fragments......................................................................... 107
Numbering Types...................................................................................................... 111
How to Reset Page Numbering at First Chapter/Part................................................. 116
How to Use Part, Chapter, and Subtopics Numbers in Links...................................... 117
How to Include Topic Sections in TOC..................................................................... 117

Table of Contents...................................................................................................... 118
Table of Contents - XML Fragment.......................................................................... 118
Table of Contents - Built-in CSS............................................................................ 120
How to Increase TOC Depth.................................................................................... 120
How to Style the Table of Contents Entries ................................................................. 121
How to Change the Header of the Table of Contents ................................................... 122
How to Make the Table of Contents Start on an Odd Page ......................................... 124
How to Display a Topic Before the Table of Contents ................................................ 124
How to Display Short Descriptions in the TOC ......................................................... 124
How to Remove Entries from the TOC ......................................................................... 125
How to Hide the TOC ................................................................................................. 125
Table of Contents on a Page (Mini TOC) .................................................................... 125
   Table of Contents for Chapters (Mini TOC) - XML Fragment ................................... 127
   Table of Contents for Chapters (Mini TOC) - Built-in CSS ...................................... 129
   How to Style the Table of Contents for Chapters (Mini TOC) ................................. 129
List of Tables/Figures ............................................................................................... 130
   How to Set a Header for a List of Tables/Figures .................................................... 131
   How to Remove the Numbers Before a List of Tables or Figures ........................... 132
Double Side Pagination ............................................................................................. 132
   How to Start Chapters on Odd Pages ..................................................................... 133
   How to Style the Empty (Blank) Pages .................................................................. 133
   How to Force an Odd or Even Number of Pages in a Chapter ............................... 134
   How to Style the First page of a Chapter ................................................................. 134
Multiple Column Pages ............................................................................................. 134
   How to Use a Two Column Layout ......................................................................... 134
PDF Bookmarks ......................................................................................................... 136
   PDF Bookmarks - Built-in CSS ............................................................................... 136
   How to Change the Bookmark Labels using the Navigation Title ............................ 137
   How to Control Bookmarks Depth and Sections Display in PDF ............................. 137
   How to Specify the Open/Closed PDF Bookmark State .......................................... 138
   How to Remove the Numbering From the PDF Bookmarks .................................... 138
Index .......................................................................................................................... 139
   Index - XML Fragment ........................................................................................... 140
   Index - Built-in CSS ............................................................................................... 142
   How to Style the Index Page Title and the Grouping Letters ................................. 143
   How to Style the Index Terms Labels ...................................................................... 143
   How to Add Filling Dots Between the Index Labels and the Page Numbers ............ 144
Contents | vi

How to Change the Index Page Number Format and Reset its Value ........................................ 145
How to Impose a Table-like Index Layout ............................................................................. 145

Appendices ......................................................................................................................... 148
How To Control Page Break Within Appendices ................................................................. 149

Footnotes ............................................................................................................................. 149
Footnotes - Built-in CSS ...................................................................................................... 149
How to Change Style of the Footnote Markers and Footnote Calls ................................ 149
How to Add a Separator Above the Footnotes ................................................................. 150
How to Reset the Footnotes Counter ................................................................................ 150
How to Display Footnotes Below Tables .......................................................................... 151

Hyphenation ....................................................................................................................... 155
Hyphenation Dictionaries ................................................................................................. 155
Installing New Hyphenation Dictionaries ....................................................................... 156
How to Alter a Hyphenation Dictionary ........................................................................... 156
How to Enable Hyphenation for Entire Map ..................................................................... 157
How to Enable/Disable Hyphenation for an Element ....................................................... 158
How to Define Hyphenation for a Specific Word ............................................................... 159
How to Force or Avoid Line Breaks at Hyphens .............................................................. 159

Accessibility ....................................................................................................................... 160
Accessibility - Built-in CSS ............................................................................................... 160
How to Create Fully Accessible Documents ................................................................... 160

Archiving ............................................................................................................................. 161
How to Allow Document Archiving ................................................................................ 161

Fonts ..................................................................................................................................... 162
How to Avoid Characters Being Rendered as # ................................................................ 162
How to Set Fonts in Titles and Content ............................................................................. 164
How to Use Fonts for Asian Languages ............................................................................ 165
How to Set Fonts for Displaying Music ............................................................................ 166

Comments, Highlights, and Tracked Changes ..................................................................... 167
Comments and Tracked Changes - Built-in CSS ............................................................... 168
Comments and Tracked Changes - HTML Fragment ........................................................ 168
How to Style Tracked Changes or Comments .................................................................. 171
How to Style Tracked Changes Shown as Footnotes ......................................................... 171
How to Show Only Change Bars on Tracked Changes................................................................. 172
Draft Watermarks......................................................................................................................... 173
How to Remove the Related Links Section.................................................................................. 173
How to Add a Draft Watermark on All Pages.............................................................................. 174
How to Add a Draft Watermark in the Foreground................................................................. 174
How to Add a Draft Watermark Depending on Metadata..................................................... 174
Flagging Content.......................................................................................................................... 175
How to Flag Content Using Change Bars ............................................................................. 176
How to Flag Content Using Images......................................................................................... 176
Styling the Content.................................................................................................................... 176
Reusing the Styling for WebHelp and PDF Output................................................................. 177
Titles........................................................................................................................................ 177
Equations.................................................................................................................................. 180
Lists........................................................................................................................................ 181
Links........................................................................................................................................ 185
Images and Figures.................................................................................................................... 187
Videos....................................................................................................................................... 196
Tables....................................................................................................................................... 197
Programming Elements............................................................................................................ 207
Notes.......................................................................................................................................... 212
Tasks.......................................................................................................................................... 213
Abbreviated Forms.................................................................................................................... 214
Trademarks................................................................................................................................. 215
Styling Through Custom Parameters....................................................................................... 215
How to Limit the Depth of the TOC Using a Parameter......................................................... 216
How to Change the Page Size Using a Parameter................................................................. 216
How to Change the Cover Page Using a Parameter.............................................................. 217
Chapter 5. Controlling the Publication Content........................................................................ 218
How to Omit the Front Page, TOC, Glossary, Index for a Plain DITA Map............................ 219
How to Make Chapters Look Like Individual Publications .................................................. 220
Chapter 6. XSLT Extensions for PDF Transformations.......................................................... 221
How to Use XSLT Extension Points for PDF Output from a Publishing Template................ 221
How to Style Codeblocks with a Zebra Effect......................................................................... 222
How to Remove the Related Links Section............................................................................. 224
<table>
<thead>
<tr>
<th>Contents</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>How to Wrap Words in Markup</td>
<td>225</td>
</tr>
<tr>
<td>How to Convert Definition Lists into Tables</td>
<td>226</td>
</tr>
<tr>
<td>How to Display Footnotes Below Tables</td>
<td>228</td>
</tr>
<tr>
<td>How to Wrap Scientific Numbers in Tables Cells</td>
<td>232</td>
</tr>
<tr>
<td>How to Use a Bullet for Tasks that Contain a Single Step</td>
<td>233</td>
</tr>
<tr>
<td>How to Change the Critical Dates Format</td>
<td>235</td>
</tr>
<tr>
<td>How to Remove Links from Terms</td>
<td>236</td>
</tr>
<tr>
<td>How to Display Glossary as a Table</td>
<td>238</td>
</tr>
<tr>
<td>How to Include Sections in the Mini TOC</td>
<td>243</td>
</tr>
<tr>
<td>How to Add a Link to the TOC</td>
<td>246</td>
</tr>
<tr>
<td>How to Use XSLT Extension Points for PDF Output from a DITA-OT Plugin</td>
<td>248</td>
</tr>
<tr>
<td>How to Style Codeblocks with a Zebra Effect</td>
<td>248</td>
</tr>
<tr>
<td>How to Remove the Related Links Section</td>
<td>250</td>
</tr>
<tr>
<td>How to Use Custom Parameters in XSLT Stylesheets</td>
<td>250</td>
</tr>
<tr>
<td>Chapter 7. DITA-OT Extension Points</td>
<td>252</td>
</tr>
<tr>
<td>How to Contribute a Custom CSS to the Transformation from a DITA-OT Plugin</td>
<td>252</td>
</tr>
<tr>
<td>Chapter 8. Localization</td>
<td>254</td>
</tr>
<tr>
<td>How to Customize CSS Strings</td>
<td>254</td>
</tr>
<tr>
<td>How to Modify Existing Strings</td>
<td>255</td>
</tr>
<tr>
<td>How to Add New Strings</td>
<td>256</td>
</tr>
<tr>
<td>Chapter 9. Security</td>
<td>258</td>
</tr>
<tr>
<td>How to Protect PDF Files by Setting Security Permissions</td>
<td>258</td>
</tr>
<tr>
<td>Chapter 10. Troubleshooting</td>
<td>259</td>
</tr>
<tr>
<td>Damaged PDF File</td>
<td>259</td>
</tr>
<tr>
<td>Error Parsing CSS File - Caused by a Networking Problem</td>
<td>259</td>
</tr>
<tr>
<td>Failed to Run Pipeline: The Entity Cannot Be Resolved Through Catalogs</td>
<td>260</td>
</tr>
<tr>
<td>Disappearing Thin Lines or Cell Borders</td>
<td>260</td>
</tr>
<tr>
<td>Glossary Entries Referenced Using 'glossref' are not Displayed</td>
<td>261</td>
</tr>
<tr>
<td>The format-date() XPath Function Does Not Respect the Specified Locale</td>
<td>261</td>
</tr>
<tr>
<td>Missing Title when Publishing Single Topics with PDF/UA-1</td>
<td>263</td>
</tr>
<tr>
<td>Highlights Span Unexpectedly to the End of the Page</td>
<td>263</td>
</tr>
<tr>
<td>Unexpected Page Break Before or After an Element</td>
<td>264</td>
</tr>
<tr>
<td>Chapter 11. Glossary</td>
<td>265</td>
</tr>
</tbody>
</table>
Contents | ix

Bookmap.............................................................................................................265
DITA Map..............................................................................................................265
DITA Open Toolkit...............................................................................................265
DITA-OT-DIR.........................................................................................................265
Framework............................................................................................................265
Index......................................................................................................................a
Copyright...............................................................................................................g
1. 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 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 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](http://docs.oasis-open.org/dita/v1.2/os/spec/DITA1.2-spec.html).


### Related Information:

**DITA-OT DAY 2017: Using CSS to Style PDF Output**

### Supported Processors

The **DITA-OT CSS-based PDF Publishing Plugin** supports the following CSS processors:

- **Oxygen PDF Chemistry** - This is recommended processor because the built-in CSS files were fine-tuned for this processor. For example, *metadata extraction (on page 91)* only functions with this processor. If the plugin is started from an **Oxygen XML Editor/Author** distribution, a Chemistry installation is not needed.


### Technical Details

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-OT3.x/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 221) 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 177). During this phase, the `com.oxygenxml.pdf.css.xsl.merged2html5` (on page 221) 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 29).
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

  \(\text{set ANT_OPTS=}\%\text{ANT_OPTS}\% \text{ } -Xmx1024M\)

- Linux/macOS

  \(\text{export ANT_OPTS=}\$\text{ANT_OPTS} -Xmx1024M\)

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.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>args.allow.external.coderefs</code></td>
<td>Enables the inclusion of code files that are located outside the DITA map folder hierarchy, referenced using the DITA <code>&lt;coderef&gt;</code> element. Allowed values are <code>yes</code> or <code>no</code> (default).</td>
</tr>
<tr>
<td><code>args.chapter.layout</code></td>
<td>Specifies whether chapter-level TOCs are generated for bookmaps. When set to <strong>MINITOC</strong>, a small section with links is added at the beginning of each chapter. The default is <strong>BASIC</strong>. For details, see: Table of Contents on a Page (Mini TOC) <em>(on page 125).</em> Allowed values:</td>
</tr>
<tr>
<td></td>
<td>• <strong>BASIC</strong> - No chapter TOC is created.</td>
</tr>
<tr>
<td></td>
<td>• <strong>MINITOC</strong> - A chapter-level TOC is generated.</td>
</tr>
<tr>
<td></td>
<td>• <strong>MINITOC-BOTTOM-LINKS</strong> - A chapter-level TOC is generated, with the links under the chapter description.</td>
</tr>
<tr>
<td><code>args.css</code></td>
<td>You can use this to specify a list of CSS URLs to be used in addition to those specified in the <code>dita.css.list</code> parameter or publishing template. The files must have URL syntax and be separated using semicolons.</td>
</tr>
<tr>
<td><code>args.css.param.*</code></td>
<td>You can use this parameter pattern to set attributes on the root of the merged map. This means you can activate specific CSS rules from your custom CSS using custom attributes. For examples, see: Styling Through Custom Parameters <em>(on page 215).</em></td>
</tr>
<tr>
<td><code>args.css.param.clone-referenced-footnotes</code></td>
<td>You can use this parameter to control the footnotes behavior:</td>
</tr>
</tbody>
</table>
• 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.

<table>
<thead>
<tr>
<th>args.css.param.numbering</th>
<th>You can use this parameter to change the numbering of the first-level topics (chapters) and nested topics. Allowed values:</th>
</tr>
</thead>
<tbody>
<tr>
<td>shallow</td>
<td>Only the topics from the first level are numbered (chapters). This is the default.</td>
</tr>
<tr>
<td>deep</td>
<td>All the topics from the map are numbered (nested topics up to level 3).</td>
</tr>
<tr>
<td>deep-chapter-scope</td>
<td>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 &amp; CSS transformation scenario.</td>
</tr>
<tr>
<td>deep-chapter-scope-no-page-reset</td>
<td>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 &amp; CSS transformation scenario.</td>
</tr>
</tbody>
</table>

For more details, see Numbering Types *(on page 111).*

<table>
<thead>
<tr>
<th>args.css.param.numbering-sections</th>
<th>Controls whether or not the sections are included in the table of contents. When set to yes (sections are included), they are numbered according the numbering scheme set by the args.css.param.numbering parameter.</th>
</tr>
</thead>
</table>

<p>| 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. |</p>
<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>args.css.param.show-profiling-attributes</code></td>
<td>Controls whether or not the profiling attributes are displayed in the output.</td>
</tr>
<tr>
<td></td>
<td>Allowed values:</td>
</tr>
<tr>
<td></td>
<td>• yes</td>
</tr>
<tr>
<td></td>
<td>• no (default)</td>
</tr>
<tr>
<td><code>args.css.param.title.layout</code></td>
<td>Changes the structure of the title element. In the output, the title area consists of two parts: one is the number of the chapter (and optionally, the sections number), and one is the title text. This parameter allows a switch between normal text flow (in-line flow) and a table layout where the number is placed in one cell and the text in the other (to avoid wrapping text under the chapter number).</td>
</tr>
<tr>
<td></td>
<td>• normal</td>
</tr>
<tr>
<td></td>
<td>• table (avoid wrapping text under counter)</td>
</tr>
<tr>
<td><code>args.draft</code></td>
<td>Specifies whether or not the content of <code>&lt;draft-comment&gt;</code> and <code>&lt;required-cleanup&gt;</code> elements is included in the output.</td>
</tr>
<tr>
<td></td>
<td>Allowed values:</td>
</tr>
<tr>
<td></td>
<td>• no (default) - No draft information is shown in the output.</td>
</tr>
<tr>
<td></td>
<td>• yes - The draft information is shown in the output.</td>
</tr>
<tr>
<td><code>args.figurelink.style</code></td>
<td>Specifies how cross references to figures are styled in output. Allowed values:</td>
</tr>
<tr>
<td></td>
<td>• NUMBER - Only the number of the figures are shown in links.</td>
</tr>
<tr>
<td></td>
<td>• TITLE - Only the title of the figures are shown in links.</td>
</tr>
<tr>
<td></td>
<td>• NUMTITLE (default) - Both the title and number of the figures are shown in links.</td>
</tr>
<tr>
<td><code>args.gen.task.lbl</code></td>
<td>Specifies whether or not to generate headings for sections within task topics. Allowed values: YES or NO (default). When set to YES, headings such as &quot;About this task&quot;, &quot;Before you begin&quot;, &quot;Procedure&quot;, or &quot;What to do next&quot;, are shown in the task contents.</td>
</tr>
<tr>
<td><code>args.hyph.dir</code></td>
<td>Specifies the directory that contains custom hyphenation dictionaries. Further details see: Hyphenation (on page 155).</td>
</tr>
<tr>
<td><code>args.input</code></td>
<td>Specifies the main DITA map file for your documentation project.</td>
</tr>
<tr>
<td>Parameter</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------------------------</td>
<td>--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>args.keep.output.debug.files</td>
<td>Specifies whether or not the debug files generated during the transformation should be kept in the output folder. Allowed values: <strong>YES</strong> (default) or <strong>NO</strong>.</td>
</tr>
<tr>
<td>args.output.base</td>
<td>Specifies the name of the output file without a file extension. By default, the name of the PDF file is derived from the name of the DITA map file. This parameter allows you to override it.</td>
</tr>
<tr>
<td>args.rellinks.group.mode</td>
<td>Specifies the related links grouping mode. All links can be grouped into a single &quot;Related Information&quot; group or links grouped by their target type (topic, task, or concept). Allowed values: <strong>single-group</strong> (default) or <strong>group-by-type</strong>. For more details see: [How to Group Related Links by Type](on page 187).</td>
</tr>
<tr>
<td>args.tablelink.style</td>
<td>Specifies how cross references to tables are styled in output. Allowed values:</td>
</tr>
<tr>
<td>clean.temp</td>
<td>Specifies whether or not the DITA-OT deletes the files in the temporary directory after it finishes a build. Allowed values: <strong>yes</strong> (default) or <strong>no</strong>.</td>
</tr>
<tr>
<td>chemistry.security.policy</td>
<td>Specifies a Java policy file that applies to the Chemistry process. A template can be found here: plugins/com.oxygenxml.pdf.css/lib/oxygen-pdf-chemistry/config/chemistry.policy.</td>
</tr>
<tr>
<td>chemistry.security.workspace</td>
<td>Specifies a directory where the temporary files and font cache created by the Chemistry process need to be stored. This becomes required when the chemistry.security.policy parameter is specified.</td>
</tr>
<tr>
<td>chemistry.security.resources.dir</td>
<td>Path to an additional folder that Chemistry will use to read its resources (CSS, images). The process already has read access to the input map folder, the publishing templates folder, and the OPE install folder. This optional parameter should only be used when the chemistry.security.policy parameter is set.</td>
</tr>
<tr>
<td>chemistry.security.resources.host</td>
<td>The host, specified as <strong>name:port</strong>, 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.</td>
</tr>
<tr>
<td>css.processor.path.antenna-house</td>
<td>Path to the Antenna House executable file that needs to be run to generate the PDF (for example, C:\path\to\AHFCmd.exe on Windows).</td>
</tr>
</tbody>
</table>
| 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).
<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>css.processor.path.prince</td>
<td>Path to the Prince executable file that needs to be run to generate the PDF (for example, C:\path\to\prince.exe on Windows).</td>
</tr>
<tr>
<td>css.processor.type</td>
<td>Specifies the processor to use for the transformation. Allowed values: chemistry (default), antenna-house, or prince.</td>
</tr>
<tr>
<td>default.language</td>
<td>Specifies the default language for source documents. Examples: fr, de, zh, etc. Depending on the transformation type, the actual number of supported languages can vary, see: Localization (on page 254).</td>
</tr>
<tr>
<td>drop.block.margins.at.page-boundary</td>
<td>Specifies that the top and bottom margins associated with a block element should be discarded when the block is at the top or bottom of the page. Allowed values: YES (default) or NO.</td>
</tr>
<tr>
<td>editlink.ditamap.edit.url</td>
<td>Use this parameter to add an Edit link next to the topic title in the WebHelp output. When a user clicks the link, the topic is opened in Oxygen XML Web Author or Content Fusion where they can make changes that can be saved to a file server. The value should be set as the edit URL of the main DITA map used for publishing your output. The easiest way to obtain the URL is to open the map in Web Author or Content Fusion and copy the URL from the browser's address bar.</td>
</tr>
<tr>
<td>editlink.additional.query.parameters</td>
<td>You can use this optional parameter to add additional parameters to be appended to each generated edit link. Each parameter must start with &amp; (for example: &amp;tags-mode=no-tags).</td>
</tr>
<tr>
<td>editlink.remote.ditamap.url (deprecated)</td>
<td>Use this parameter in conjunction with 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: <a href="https://getFileContent/oxyengxml/userguide/master/UserGuide.ditamap">https://getFileContent/oxyengxml/userguide/master/UserGuide.ditamap</a>.</td>
</tr>
<tr>
<td>editlink.web.author.url (deprecated)</td>
<td>This parameter needs to be used in conjunction with editlink.remote.ditamap.url to add an Edit link next to the topic title in the PDF output. When a user clicks the link, the topic is opened in Oxygen XML Web Author where they can make changes that can be saved to a file server. The value should be set as the URL of the Web Author installation. For example: <a href="https://www.oxygenxml.com/oxygen-xml-web-author/">https://www.oxygenxml.com/oxygen-xml-web-author/</a>.</td>
</tr>
<tr>
<td>enable.latin.glyph.substitutions</td>
<td>When set to yes (default), glyph substitution is enabled (if the particular font supports it). This applies to Latin-based scripts only (the substitutions are always enabled in other types of scripts). If you encounter problems rendering or copying accented glyphs (e.g. umlauts or other dia-</td>
</tr>
</tbody>
</table>
critics), 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 can­
ot be mapped to a compound glyph, resulting in the glyph not being ren­
dered in the PDF output.

**Warnings:**

- Disabling substitutions also disables Latin ligatures.
- Disabling substitutions is not recommended unless ab­
solutely necessary. It is better practice to use another
font if you can find one that does not have the rendering
issues.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>figure.title.placement</td>
<td>Controls the title placement of the figures, relative to the image. Possible values include top (default) and bottom.</td>
</tr>
</tbody>
</table>
| filter.unused.glossentries                   | When set to no (default), all glossary entries are displayed in the gloss­
- 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 188). |
| 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): |

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pdf.version</td>
<td>Use this parameter to specify the version of the produced PDF. It has no impact on the set of PDF features used by the engine, but may be used to signal a compatibility level to the PDF readers. The default is 1.5.</td>
</tr>
<tr>
<td>pdf.security.restrict.printhq</td>
<td>Restricts high quality printing. Used for protecting the PDF Document. The restriction is off by default. Accepted values: yes or no.</td>
</tr>
<tr>
<td>pdf.security.restrict.assembledoc</td>
<td>Restricts assembling document (e.g. adding pages). Used for protecting the PDF Document. The restriction is off by default. Accepted values: yes or no.</td>
</tr>
<tr>
<td>pdf.security.restrict.accesscontent</td>
<td>Restricts extracting text and graphics. Used for protecting the PDF Document. The restriction is off by default. Accepted values: yes or no.</td>
</tr>
<tr>
<td>pdf.security.restrict.fillinforms</td>
<td>Restricts filling in existing interactive forms. Used for protecting the PDF Document. The restriction is off by default. Accepted values: yes or no.</td>
</tr>
<tr>
<td>pdf.security.restrict.annotations</td>
<td>Restricts filling in existing interactive forms. Used for protecting the PDF Document. The restriction is off by default. Accepted values: yes or no.</td>
</tr>
<tr>
<td>pdf.security.restrict.print</td>
<td>Restricts printing. Used for protecting the PDF Document. The restriction is off by default. Accepted values: yes or no.</td>
</tr>
<tr>
<td>pdf.security.restrict.copy</td>
<td>Restricts copying content. Used for protecting the PDF Document. The restriction is off by default. Accepted values: yes or no.</td>
</tr>
<tr>
<td>pdf.security.restrict.edit</td>
<td>Restricts copying content. Used for protecting the PDF Document. The restriction is off by default. Accepted values: yes or no.</td>
</tr>
<tr>
<td>pdf.security.user.password</td>
<td>User password. The document can be opened using this password. When the owner password parameter is not specified, the user password gives full rights to the people using it. When the owner password parameter is specified, the people can open the document using the user password but restrictions will apply. Missing by default.</td>
</tr>
<tr>
<td>pdf.security.owner.password</td>
<td>Owner password. There are no restrictions for people using this password.</td>
</tr>
<tr>
<td>pdf.security.encrypt.metadata</td>
<td>Encrypts the metadata. By default active when other security parameters are set. Accepted values: yes or no.</td>
</tr>
<tr>
<td>Parameter</td>
<td>Description</td>
</tr>
<tr>
<td>-----------</td>
<td>-------------</td>
</tr>
<tr>
<td>show.changes.and.comments</td>
<td>When set to <strong>yes</strong>, the user comments, colored highlights and tracked changes are shown in the output.</td>
</tr>
<tr>
<td>show.changes.and.comments.as.changebars</td>
<td>When set to <strong>yes</strong> (default) and the show.changes.and.comments parameter is also set to <strong>yes</strong>, the user comments and tracked changes are shown as change bars in the PDF output. This parameter can be used in conjunction with the 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 <em>(on page 41).</em></td>
</tr>
<tr>
<td>show.changes.and.comments.as.pdf.sticky.notes</td>
<td>When set to <strong>yes</strong> (default) and the show.changes.and.comments parameter is also set to <strong>yes</strong>, the user comments and tracked changes are shown in the PDF output as sticky note annotations. When set to <strong>no</strong>, the comments and tracked changes are left in the document model and are styled by the default CSS rules as footnotes. You can override this from your customization CSS <em>(on page 41).</em></td>
</tr>
<tr>
<td>show.changed.text.in.pdf.sticky.notes.content</td>
<td>When set to <strong>yes</strong> (default) and both the show.changes.and.comments and show.changes.and.comments.as.pdf.sticky.notes parameters are also set to <strong>yes</strong>, the inserted and deleted text is shown in the sticky note annotations. When set to <strong>no</strong>, only the inserted and deleted labels are shown in the annotations (this is useful for search scope).</td>
</tr>
<tr>
<td>show.image.map.area.numbers</td>
<td>When set to <strong>yes</strong>, a counter for each area from the image map is displayed over the image, near the defined shape. The default is <strong>no</strong>.</td>
</tr>
<tr>
<td>show.image.map.area.shapes</td>
<td>When set to <strong>yes</strong>, each of the image map area shapes is displayed with a translucent fill over the image. You can use this to debug your image maps. The default is <strong>no</strong>.</td>
</tr>
<tr>
<td>show.media.as.link</td>
<td>When set to <strong>yes</strong>, media objects will not appear and an external link is generated for each one instead.</td>
</tr>
<tr>
<td>sort.and.group.glossentries</td>
<td>When set to <strong>no</strong> (default), elements in the glossary are sorted based upon the document order. If set to <strong>yes</strong>, elements in the glossary are sorted alphabetically and grouped by their first letter.</td>
</tr>
<tr>
<td>store-type</td>
<td>Setting this parameter to <strong>memory</strong> will increase the processing speed and thus, could help decrease the publishing time.</td>
</tr>
<tr>
<td>table.title.placement</td>
<td>Controls the placement of the title for tables. Possible values include <strong>top</strong> (default) and <strong>bottom</strong>.</td>
</tr>
<tr>
<td>table.title.repeat</td>
<td>Specifies whether or not a table caption should repeat on other pages when the table spans onto multiple pages. The caption is not repeated.</td>
</tr>
</tbody>
</table>
for tables nested in lists or other tables. Allowed values are **yes** (default) or **no**.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>use.css.for.embedded.svg</td>
<td>When set to <strong>yes</strong> (default), the CSS files specified in the publishing template or by the <code>args.css</code> parameter are also applied on embedded SVG elements. Allowed values are <strong>yes</strong> and <strong>no</strong>.</td>
</tr>
<tr>
<td>use.navtitles.in.all.links</td>
<td>Specifies whether a <code>&lt;navtitle&gt;</code> 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 <strong>yes</strong> and <strong>no</strong> (default).</td>
</tr>
<tr>
<td>parallel</td>
<td>Specifies whether or not certain pre-processing tasks should be run in parallel. Setting this parameter to <strong>true</strong> may add a small increase to the publishing speed. Allowed values are: <strong>true</strong> and <strong>false</strong> (default).</td>
</tr>
</tbody>
</table>

The following parameters can be used to specify a publishing template:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pdf.publishing.template</td>
<td>Specifies the path to the folder containing the custom PDF template.</td>
</tr>
<tr>
<td>pdf.publishing.template.descriptor</td>
<td>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.</td>
</tr>
</tbody>
</table>

The following parameter is available on all DITA transformations when using the **Oxygen Publishing Engine**:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>args.disable.security.checks</td>
<td>Specifies whether or not to load external entities that are not solved through catalogs. For security reasons, the default is <strong>no</strong>. Allowed values:</td>
</tr>
<tr>
<td></td>
<td>• <strong>yes</strong></td>
</tr>
<tr>
<td></td>
<td>• <strong>no</strong> (default)</td>
</tr>
</tbody>
</table>

The following parameters are only available for the **DITA PDF - based on HTML5 & CSS** single DITA topic transformation scenario (**pdf-css-html5-single-topic** trans type):

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>args.root.map</td>
<td>Specifies the path of the root map file used to expand the key references in the published topic.</td>
</tr>
<tr>
<td>args.enable.root.map.key.processing</td>
<td>Indicates whether or not the keys should be processed using the root map parameter. Allowed values:</td>
</tr>
</tbody>
</table>
### Console Logging

To activate the logging of the last processing stage, involving the usage of the Chemistry processor to generate the PDF from the merged HTML, use the `--verbose` (or `-v`) DITA-OT parameter from the command line.

**Note:** When the transformation is started from an Oxygen application, this parameter is automatically set.

### License Key

#### Chemistry License

If you have an Oxygen PDF Chemistry license key, you will be able to generate PDF output that is not stamped with the Chemistry logo image 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.
2. 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 15). 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). For more information about the build parameters, see DITA-OT: Building Output Using the dita Command.

Generating PDF from an Integration Server

PDF output can be automatically generated from a Continuous Integration/Continuous Delivery system, such as Jenkins.

To integrate PDF output with the Jenkins CI tool, follow these steps:

1. Create a Maven project to incorporate the DITA-OT that already integrates Oxygen DITA-OT CSS-based PDF Publishing plugin.
2. Go to the root of your Maven project and edit the pom.xml file to include the following fragment:

```xml
<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>
</project>
```
<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>
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
3. 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](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:
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.

**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:

```xml
<publishing-template>
  <name>Flowers</name>

  <pdf>
    <tags>
      <tag>purple</tag>
      <tag>light</tag>
    </tags>

    <preview-image file="flowers-preview.png"/>

    <resources>
      <css file="flowers.css"/>
    </resources>

    <parameters>
      <parameter name="figure.title.placement" value="top"/>
    </parameters>
  </pdf>
</publishing-template>
```

Tip:
It is recommended to edit the template descriptor in Oxygen XML Editor/Author because it provides content completion and validation support.

Template Name and Description

Each template descriptor file requires a `<name>` element. This information is displayed as the name of the template in the transformation scenario dialog box.

Optionally, you can include a `<description>` and it displayed when the user hovers over the template in the transformation scenario dialog box.

```xml
<publishing-template>
  <name>Flowers</name>

  <description>Flowers themed light colored template</description>

  ...
</publishing-template>
```
Template Author

Optionally, you can include author information in the descriptor file and it displayed when the user hovers over the template in the transformation scenario dialog box. This information might be useful if users run into an issue or have questions about a certain template.

If you include the `<author>` element, a `<name>` is required and optionally you can include `<email>`, `<organization>`, and `<organizationUrl>`.

```
<publishing-template>
  ...
  <author>
    <name>John Doe</name>
    <email>jdoe@example.com</email>
    <organization>ACME</organization>
    <organizationUrl>http://www.example.com/jdoe</organizationUrl>
  </author>
  ...
</publishing-template>
```

PDF Element

The `<pdf>` element contains various details about the template and its resources that define the PDF output. It is a required element if you intend on using a DITA Map to PDF transformation scenario. The elements that are allowed in this `<pdf>` section specify the template tags (on page 32), template preview image (on page 32), resources (on page 32) (such as CSS files), transformation parameters (on page 33), or XSLT extensions (on page 34).

```
<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>
  </pdf>
</publishing-template>
```

Template Preview Image

The `<preview-image>` element is used to specify an image that will be displayed in the transformation scenario dialog box. It provides a visual representation of the template to help the user select the right template. The image dimensions should be 200 x 115 pixels and the supported image formats are: JPEG, PNG, or GIF.

You can also include an `<online-preview-url>` element to specify the URL of a published sample of your template. This will display an Online preview icon in the bottom-right corner 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>
  </pdf>
</publishing-template>
```

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>
  </pdf>
</publishing-template>
```
Transformation Parameters

You can also set one or more transformation parameters in the descriptor file.

```xml
<publishing-template>
  ...
  <pdf>
    ...
    <parameters>
      <parameter name="show.changes.and.comments" value="yes"/>
    </parameters>
  </pdf>
</publishing-template>
```

The following information can be specified in the `<parameters>` element:

**Parameter name**

The name of the parameter. It may be one of the transformation parameters listed in the `Parameters` tab of the DITA Map PDF - based on HTML5 & CSS transformation scenario or a DITA-OT PDF-based output parameter.

**Note:**

It is not recommended to specify an input/output parameter in the descriptor file (such as the input Map, DITAVAL file, or temporary directory).

**Attention:**

JVM arguments like `-Xmx` cannot be specified as a transformation parameter.

**Parameter Value**

The value of the parameter. It should be a relative path to the template root folder for file paths parameters.

**Parameter Type**

The type of the parameter: `string` or `filepath`. The `string` value is default.

After creating a publishing template and creating a publishing template, 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>
  </pdf>
</publishing-template>
```

For more information about the available extension points, see: XSLT Extensions for PDF Transformations (on page 221).

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>
  </webhelp>

  ...  
</publishing-template>
```
How to Create a Publishing Template

To create a customization, you can start from scratch or from an existing template, and then adapt it according to your needs.

Creating a Publishing Template Starting from Scratch

To create a new *Oxygen Publishing Template (on page 266)*, follow these steps:

1. Create a folder that will contain all the template files.

2. In *Oxygen XML Editor/Author*, open the new document wizard (use File > New or the New toolbar button), then choose the *Publishing Template Descriptor* template.
3. Save the .opt file into your customization directory.
4. Open the .opt file in the editor and customize it to suit your needs. See: Publishing Template Package Contents for PDF Customizations (on page 29).

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 266) 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 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. See: Publishing Template Package Contents for PDF Customizations (on page 29).

For more information about creating and customizing publishing templates, watch our video demonstration:

https://www.youtube.com/embed/zNmXfKWx08

Creating a Publishing Template Using the Oxygen Styles Basket

Another way to create an Oxygen Publishing Template (on page 266) is to use the Oxygen Styles Basket. This tool is a handy free-to-use web-based visual tool that helps you pick and mix aspects from galleries and generate an archive that can directly be used into DITA Map WebHelp Responsive or DITA Map PDF - based on HTML5 & CSS transformation scenarios.

It is based on galleries that you can use to pick and mix styling aspects to create a custom look and feel. Various different types of aspects can be selected to be integrated in a CSS stylesheet (such as fonts, tables, lists, spacing, code).
You can see the results of your changes in the Preview pane or you can click the See Results toolbar button to see the results in a generated preview version of either PDF or WebHelp output. When you are finished with your customization, you can Download the results as a Publishing Template Package that you can use in your transformation process.

Of course, it is also possible to re-upload a previously generated package for further customization.

Resources

For more information about the Oxygen Styles Basket, see the following resources:

- Video: Introducing the New Oxygen Styles Basket
- Webinar: Using Oxygen Styles Basket to Create CSS Customization from Scratch

### How to Edit a Packed Publishing Template

To edit an existing Oxygen Publishing Template (on page 266) 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.

## 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 15)

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 at the bottom of that preferences page to Project Options.
4. Share your project file (.xpr).
4. 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 and the DITA PDF - based on HTML5 & CSS transformation scenarios:

1. Create the CSS file that will contain all of your customizations. It is recommended to create this file in your project directory so you can edit it easily.

   **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.
   - Use the Oxygen Styles Basket (on page 37) to generate basic selectors for common content.

3. For linking the CSS file, you have two options:
   - Create a publishing template, create the customization CSS file inside the template folder, and link it to the publishing template descriptor. For assistance, see Publishing Templates.
   - Choose an existing publishing template, then edit the scenario and set the full path to the custom CSS file as the value of the `args.css` parameter. The rules from custom CSS will override the rules from the template CSS files.

4. Run the transformation scenario.

   **Tip:**
   For more information and some tips in regard to publishing DITA documents to PDF using CSS, watch our Webinars:

   - Transforming DITA documents to PDF using CSS, Part 1 – Page Definitions, Cover Page and PDF Metadata:
     https://www.youtube.com/embed/5NsVEOvxbas
   - Transforming DITA documents to PDF using CSS, Part 2 – Book Design, Pagination, Page Layout, and Bookmarks:
Debugging the CSS

If you notice that some of the CSS properties were not applied as expected, some of the tips offered in this topic might help you with the debugging process.

Merged Map File

Depending on the type of transformation, one or more merged map files are created at some point during the transformation stages. These files could be used to help debug unexpected results.

1. The first thing you should try is to check the file structure of the merged map file. This can be found in the out/pdf-css directory and it has the .merged.html file extension (you will also find a .merged.xml file that aggregates the entire ditamap structure). You can open the HTML files in Oxygen XML Editor/Author to examine the structure. Optionally, you can use the pretty print feature (Format and Indent) to make the structure easier to read.
2. Check that the CSS selectors are written correctly against the document structure.
3. If you still cannot identify the problem, then inspect how the styles are applied (you can try any of the methods listed below).

Inspecting the Applied Styles Using the Chrome Browser

To inspect the applied CSS styles using Chrome:

1. Open the file ending in .merged.html.
2. Click on the element you want to inspect.
3. Activate the Chrome Developer Tools by using > More Tools > Developer Tools, or press CTRL +SHIFT+I.
4. Activate the Rendering pane by using > More Tools > Rendering:
5. In the **Rendering** pane, select **print** from the **Emulate CSS media** section. This will activate the CSS selectors enclosed in `@media print {...}`.

**Note:**
This allows you to debug the styling of elements, table of contents, and index, but not the styles of the page margin boxes (headers, footers) or page breaks.
Tip:
In the left pane of the Developer Tools interface, you can inspect elements and their styles in the Elements tab. You can click on any of the links to display the applied CSS files in the Styles tab in the right pane. Editing the styles in that pane results in a live preview of how the change will affect the output.

CAUTION:
Do not modify the built-in rules directly in the CSS files from the Oxygen XML Editor/Author installation. Instead, copy the rules to your own customization CSS.

Inspecting the Applied Styles Using Oxygen XML Editor/Author
To inspect styles:

1. In Oxygen XML Editor/Author, open the file ending in .merged.html.
2. [Optional] From the Styles toolbar, choose the + Print Ready entry. This will activate certain CSS selectors enclosed in @media print {...}.
3. Click on the element you want to style. Use the Inspect Styles action from the Contextual Menu. A specialized CSS Inspector view will show the built-in CSS rules.

Tip:
With this file open in Author mode, it might be helpful to switch the Tags Display Mode to Full Tags with Attributes. You might be able to identify the selector you need to style without using the CSS Inspector view.

Note:
This allows you to debug styling of elements, but not of the page margin boxes (headers, footers) or page breaks.

CAUTION:
Do not modify the built-in rules directly in the CSS files from the Oxygen XML Editor/Author installation. Instead, copy the rules to your own customization CSS.

Other techniques
These are some other techniques you may find useful:

- Add background and borders properties to the specific CSS rule. If they do not appear in the output then there is a problem with the rule selector.
- Try to use the !important notation to the property that is not applied, or make the selector more specific (you can add more parent selectors).
• To figure out how the elements are mapped to PDF, you can use this fragment in the customization CSS:

```css
* {
    border: 1pt solid blue !important;
}

*:before(1000) {
    content: oxy_name() !important;
    color: orange;
}

*:before(999) {
    content: "[ class= " attr(class) "] " !important;
    color: orange;
}
```

This will show the element name, its class attribute, and will paint a blue border around each of the elements in the output. It will not show the page margin boxes or some content elements that were hidden.

**How to Speed up CSS Development and Debugging**

You can speed up your CSS development considerably by not invoking the entire pipeline of transforming your DITA maps to PDF. Instead, you can use the merged map (*on page 42*) and transform it directly to PDF.

1. Transform your DITA Map to PDF using the DITA Map PDF - based on HTML5 & CSS transformation scenario.
2. Open the merged file (*on page 42*) (*.merged.html) that is located in the output directory in the editor.
3. Configure an XML to PDF transformation with CSS scenario. Do not set CSS files here since the merged file already contains pointers to the stylesheets. This scenario uses the Chemistry CSS processor.
4. Optional: Enable the output of the CSS processor using the following preferences page: Options > Preferences > XML > PDF Processors > CSS Processor.

Now you can make incremental changes to the CSS stylesheet and quickly see the results by transforming the merged file directly.

**Fastpath:**

If your changes involve only element styling, with no specific paged media CSS rules and properties, you can simply open the merged file in a browser (such as Chrome or Firefox) and refresh at each CSS change, as shown in: Debugging the CSS (*on page 42*).
How to Write XPath Expressions

This topic contains some guidelines for writing XPath expressions. They are used to extract the content from the merged DITA map document.

This is an example where the product name meta-information is placed before the front page title:

```html
/*[class~="front-page/front-page-title"]:before {
    text-align: left;
    content: oxy_xpath("(//*[contains(@class, 'topic/prodname')]/text())[1]");
    display:block;
}
```

Notes:

- Do not use the DITA element names directly. You must use the DITA `@class` attribute instead, as these attributes are propagated to the merged elements (including HTML `<div>` elements) while the element names can be lost. By using the class selectors, you also cover DITA specializations.
- Use the `[1]` XPath predicate to select the first value from the document. For example, `oxy_xpath("(/*[contains(@class, 'topic/prodname')]/text())[1]").` The meta-information might be copied multiple times in the output, inherited by the `<topicref>` elements, so you can get many more values than expected.
- Do not use strings as values for the pseudo-elements content, as the string values are not supported for pseudo-elements. Instead, use the XPath directly.
- Use the Oxygen XPath Builder view to test the XPath expressions.

Related Information:

How to Debug XPath Expressions (on page 46)

How to Debug XPath Expressions

You can use the content extracted from the document using the `oxy_xpath` function in your pseudo-elements (:before, :after) or in string-set variables.

For example, the following XPath finds the publication author that is set in the DITA Map and displays it in the bottom-left part of the cover page:

```xml
<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 46).
2. Begin by transforming your document using your customization CSS.
3. In the output folder, you will find a [MAP_NAME].merged.html file (or if you are using the DITA Map PDF - based on HTML5 & CSS transformation, a [MAP_NAME].merged.html file).
4. Open the merged file in the Oxygen XML Editor/Author.
5. Activate the XPath Builder view (Window > Show View > XPath/XQuery Builder).
6. Paste your XPath expression and click the Execute XPath button. Check if it returns the expected results.

The XPath builder has a function that allows it to display the document path of the current element from the editor (⚙️. Settings drop-down menu > 🔄 Update on cursor move). Alternatively, you can right-click the element in the merged document and select the Copy XPath action, then paste it in the XPath builder.

Related Information:
XPath Builder Documentation
XPath Examples (w3schools.com)
Default Page Definitions

All page definitions are found in: `[PLUGIN_DIR]css/print/p-pages-and-headers.css`.

**Note:**
This is listed solely for illustration purposes, as the plugin might use something different.

There are page definitions for the default page, chapter page, table of contents page, front matter page, back matter page, index page, large tables page, and blank page.

**Default Page**

The default page imposes a header that contains the publication title, chapter, and section title. They alternate on the left or right side of the page:

```css
@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:

```css
@page :left, table-of-contents:left, chapter:left { 
  @top-left { 
    content: "..."; 
  } 
} 

@page :right, table-of-contents:right, chapter:right { 
  @top-right { 
    content: "..."; 
  } 
} 
```
Chapter Page

This is inherited from the default page. The chapter page is associated with the topics marked as chapters, usually direct children of the map. It clears the header from the first page of each chapter. If you need to add other information to the chapter headers, make sure you override these rules in your CSS:

```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;
   }
}
```

Table of Contents Page

The table of content page. It clears the headers and uses a lower roman page number in the header.

```css
@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;
    }
}

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:lef t {
    @top-left           { content: counter(page, lower-roman); }
}

@page matter-page:rig ht {
    @top-right          { content: counter(page, lower-roman); }
}

Index Page

The page that contains the index terms (appears only if there are such items in your topics). It uses a lower 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 }
}
When transformed, the page layout is spread on two columns.

**Large Tables Page**

The big tables are placed on a rotated page, with orientation landscape:

```css
@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;
  }
}
```
Blank Page

The following example clears the header for the blank pages that may be created by a page-break-before, page-break-after, or by using double side pagination (on page 132):

```css
@page :blank{
  @top-left {
    content: none;
  }
  @top-right {
    content: none;
  }
}
```

Page Size

This is where you can find information on how the page sizes are defined.
Page Size - Built-in CSS rules

The [PLUGIN_DIR]/css/print/p-page-size.css file contains the default page rules. It uses the US-LETTER size (8.5 X 11 inches). The content of this file is:

```css
@page {
    padding-top: 0.2em;
    padding-bottom: 0.2em;
    size: us-letter;
    margin: 1in;
}
```

**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 41), use:

```css
@page {
    size: A4;
    margin: 2cm;
}
```

If you need different margins depending on the page side:

```css
@page {
    size: A4;
    margin: 2cm;
}
@page :left{
    margin-right: 4cm;
}
@page :right{
    margin-left: 4cm;
}
```

This would only increase the gutter margins or the inside margins needed for binding of the final book. The other margins would remain 2cm.
How to Change the Page Orientation

Suppose you want to publish on a landscape page orientation. The default is portrait, so you need to change it by using the size property. This will contain both the physical measurements and the orientation. In your customization CSS (on page 41), use:

```css
@page {
  size: us-letter landscape;
}
```

How to Change the Page Settings for a Specific Element

Suppose your publication mainly uses a portrait page orientation, but there are some topics that have wide images. To avoid having the images bleed outside of the page, you could use a wider page setting (landscape).

1. Mark the topic with an `@outputclass` attribute and give it a distinct value (for example, `wide`), you can set the attribute on the root element of the topic or on the `<topicref>` element from the map.

   ```plaintext
   Note:
   The `@outputclass` values from the `<topicref>` automatically propagate to the root of the topic from the merged map (on page 42).
   ```

2. In your customization CSS (on page 41), match the output class and associate it with a named page. In the following example, the page has a landscape orientation and small margins. This technique works for any element (e.g. a table or list) not just for a topic.

   ```css
   @page wide-page {
     size: letter landscape;
     margin: 0.5in;
   }
   *[outputclass = 'wide'] {
     page: wide-page !important;
   }
   ```

   ```plaintext
   Note:
   The `!important` rule is necessary to override the default page settings.
   ```

Page Headers and Footers

The page headers and footers use the string sets defined for publication, chapter, and section titles. These string-sets are defined in the numbering CSS (on page 107):

```plaintext
parttitle
```
Set to the title of the current part (only for DITA bookmaps that use parts).

**chaptertitle**

Set to the title of the current chapter (Shallow and Deep numbering).

**sectiontitle**

Set to the title of each section (Deep numbering only).

To see where the default page rules are defined, see: Default Page Definitions *(on page 48)*.

Although you may define string sets in your customization CSS, you need to take into account the fact that the string-set CSS property is not additive, and matching the same elements will end up breaking the current definitions. A very common use-case is to match the title element that is also used in the default CSS. The best approach, in this case, is to take a look at the rules from the numbering CSS *(on page 107)*, copy the ones dealing with string sets to your customization, then alter the property definition by adding your definition to the existing ones (and not removing the existing ones).

**Related Information:**

Numbering *(on page 107)*

**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 48)*.

**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:

```css
@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:
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 66).

How to Display Chapter's Headers on First Page

By default, the header is not displayed on the first page of each chapter:

```css
/* No headers on the chapter first page. */
@page chapter:first: left {  
  @top-left {  
    content: none;  
  }
}
@page chapter:first: right {  
  @top-right {  
    content: string(maptitle) string(parttitle) string(chaptertitle) string(sectiontitle) " | " counter(page);  
    white-space: nowrap;  
    text-align: left;  
  }
}
```
If you want to display them on the first page, you just need to override the above default rules with the following default content:

```css
@page chapter:first:
  @top-left {
    content: string(maptitle) string(parttitle) string(chaptertitle) string(sectiontitle) " | " counter(page);
  }
}
@page chapter:first:
  @top-right {
    content: string(maptitle) string(parttitle) string(chaptertitle) string(sectiontitle) " | " counter(page);
  }
}
```

How to Change the Size of Headers and Footers

This is directly related to the page margins and size.

The headers and footers are placed in the so-called page margin boxes, a series of rectangular areas residing in the page margins.

To affect the margins of all page definitions, you may use the following rule:

```css
@page {
  margin-top:3cm !important;
  margin-bottom:3cm !important;
  margin-left:2cm !important;
  margin-right:2cm !important;
}
```

If you want to affect only a specific page, like the first page from chapters for instance, you must use more specific page selectors. See the Default Page Definitions (on page 48) for details.

Note that the page margin boxes fill the entire page margin. This means the margin-top, for example, dictates the height of the @top-left-corner, @top-left, @top-center, @top-right, @top-right-corner margin boxes. These cannot have margins on themselves, so to change the position of the content inside them, you must use padding properties:

```css
@page {
  @top-left {
    content:"..."
  }
}
```
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 41), add a CSS rule similar to this:

```css
@page {
  font-size: 12pt;
  font-family: "Arial";
}
```

**Important:**

These settings apply to all page margin boxes, but not to the text inside the page.

If you want to change the settings only for a specific page type (for example, the table of contents), use the name of the page:

```css
@page table-of-contents {
  font-size: 12pt;
  font-family: "Arial";
}
```

Related Information:

How to Change the Header of the Table of Contents (on page 122)

How to Change Header Text for Each Topic

It is possible to dynamically change the header depending on the content in a topic. The following example assumes that the data to be presented in the header is located in the metadata section of each topic. One way is to specify it in the DITA map by using the `<topicmeta>` element for the `<topicref>` topic reference:

```xml
...  
  <topicref href="topics/installing.dita">
    <topicmeta>
      <data name="header-data" value="ID778-3211"/>
    </topicmeta>
  
...```

In the above example, there is 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 42), like this:
<title class="- topic/title ">Installing</title>

You install components to make them available for your solution.

This information can be extracted from the CSS:

```css
/* Define the string set variable that contains the text extracted from the data element */
*[class ~="topic/topic"] *[class ~="topic/data"] [name="header-data"] {
    string-set: hdrstr attr(value);
}

/* Using the value='none' stops applying the image. */
*[class ~="topic/topic"] *[class ~="topic/data"] [name="header-data"] [value="none"] {
    string-set: hdrstr "";
}

/* Use the string set variable in one of the page margin boxes. */
@page chapter {
    @top-left-corner {
        content: string(hdrstr);
    }
}
```

Notes:
The string set is applied to all pages that follow the data element, until another data element changes it:
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 42), 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 ">
    ...
  </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. */

/* 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 -->
```
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 41), add the following CSS rule:

```css
@page :right {
    @bottom-center {
        content: "© 2017 - My Company Ltd \A All rights reserved";
        font-size: 0.5em;
        color: silver;
    }
}
```

Note:
Other page rules (such as the table-of-contents) override the contents of the @bottom-center because they are more specific. If you need to also print the copyright in the TOC pages, then use this as the selector:

```css
@page :right, table-of-contents:right {
    ...
}
```

Note:
To use new lines (\n characters) in your headers or footers, use the \A notation, as in the example above.
How to Add a Group of Topics to the Footer

To create a footer that contains the content of several topic files, but only on the last page, there are two possible approaches:

**Method 1: Using the position:fixed CSS Property**

1. Group all the footer topics under a single parent topic, under the last topic from your DITA map. For example, you can have the following map structure:

   ...  
   End topic  
   Footer container topic  
   Footer content topic 1  
   Footer content topic 2

2. Add an `@outputclass=footer` on the `<topic>` root element of the footer container topic, or on its `<topicref>` in the map.

3. Use the CSS `position: fixed` property to position this topic to the bottom of the page:

   ```css
   *[outputclass =~ "footer"] {  
     position: fixed;  
     bottom: 0.5in; 
     left: 0.5in; 
     width:5in; 
     height:200pt; 
   }
   ```

   **Note:**
   Make sure the width and height are enough for the content of the footer to fit. Be careful because the content might bleed out of the page. Use bottom and left values to position the block in the page.

**Method 2: Using the float:footnote CSS Property**

The second approach would be to declare the footer block as a footnote. Assuming the same DITA Map structure as above, you can use the following CSS fragment:

```css
*[outputclass =~ "footer"] {  
float:footnote;  
}

*[outputclass =~ "footer"]:footnote-call{  
color:transparent;  
font-size:0; 
}
```
Oxygen DITA-OT CSS-based PDF Publishing plugin 25.1 | 4 - Customizing PDF Output Using CSS | 64

---

```css
*[outputclass =~ "footer"] .footnote-marker {
  color: transparent;
  font-size: 0;
}
```

**Note:**
Use transparent colors and/or zero size font to avoid the display of the footnote counters.

---

**How to Add a Background Image to the Header**

A common use-case is to add a background image to one of the page corners.

```css
@page :left {
  @bottom-left-corner {
    content: " ";
    background-image: url('https://www.oxygenxml.com/resellers/resources/OxygenXMLEditor_icon.svg');
    background-repeat: no-repeat;
    background-position: 50% 50%;
  }
}
```

**Important:**
Always specify a `content` property. If not, the page margin box will not be generated.

Another use-case is to use the `@top-left` or `@top-right` page margin boxes. These boxes have an automatic layout and they can be very small if they have no content. If there is no text to be placed over the image, use a series of non-breaking spaces (`\A0`) to increase the box width as in the following example (alternatively, you can use the technique described in How to Decorate the Header by Using a Background Image on the Entire Page (on page 66)):

```css
@page :left {
  @top-left {
    content: '\A0\A0\A0\A0\A0\A0\A0\A0\A0\A0\A0\A0\A0\A0\A0\A0\A0\A0\A0\A0\A0';
    background-image: url('https://www.oxygenxml.com/resellers/resources/OxygenXMLEditor_icon.svg');
    background-repeat: no-repeat;
    background-position: 50% 50%;
  }
}
```
Note:
You can use raster image formats (such as PNG or JPEG), but it is best to use vector images (such as SVG or PDF). They scale very well and produce better results when printed. In addition, the text from these images is searchable and can be selected (if the glyphs have not been converted to shapes) in the PDF viewer.

Related Information:
Images and Figures (on page 187)
How to Add a Background Image for the Cover (on page 81)
How to Add a Link in Headers and Footers (on page 65)

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`:

```html
<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">
        <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 **PDF Chemistry** displayed as its text (the content of the `<text>` element).

Note:
If you just want to add a link without text, you can define a rectangle that contains the link instead of text.

To display the link, you just need to set your SVG file as the content of one of the page margin boxes:

```css
@page {
    @top-left {
        content: url("custom.svg");
    }
}
```
**Method 2: Using the CSS -oxy-link Property**

It is also possible to add a link inside the document header (or footer) by using the -oxy-link property on the @page margin box declaration. The entire page margin box will behave as a link and will be clickable.

```
@page {
  @top-left {
    content: "Link";
    -oxy-link: "https://www.oxygenxml.com/";
    color: blue;
  }
}
```

**How to Decorate the Header by Using a Background Image on the Entire Page**

If you want to precisely position artwork and the page margin boxes are not sufficient, it is possible to use a background image for the entire page.

This technique consists of creating an image (SVG is the best since it is a vector image) as wide as the page that would contain the logo and placing other decorations at the desired locations. This offers the best results and the position of the artwork does not depend on the page margin contents.

**Example:**

```
@page :left, chapter: left, chapter:first: left {
  background-image: url('img/page_background_image_with_logos_and_artwork_for_left_page.svg');
  background-repeat: no-repeat;
  background-position: 50% 50%;
  background-size: 8.5in 11.5in; /* Optional: Adapt to your page size. */
}
```

For a list of all the possible page names, see: Default Page Definitions (on page 48).

**Related Information:**

- How to Add a Background Image for the Cover (on page 81)

**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 64).
- Use the oxy_label constructor. This is a function that creates a text label with a set of styles.

```
You can combine the `oxy_label` with `oxy_xpath`, to extract and style a piece of text from the document:

```plaintext
content: oxy_label(text, oxy_xpath("/some/xpath"), styles, "color:blue; ");
```

**Note:**

These functions work only with the Chemistry CSS processor.

**Note:**

You cannot use `string()` inside an `oxy_label()`. As a workaround, to apply styling on the dynamic text retrieved by a `string()` function you can define some overall styles for the entire page margin box and then use the `oxy_label` to style differently the static text.

```plaintext
@page {
    @top-right {
        color: red;
        content: oxy_label(text, "My Company", styles, "color:black")
            ,
        string(chaptertitle); /* This inherits the styling from @top-right*/
    }
}
```

- Use two adjacent page margin boxes, and style them differently:

```plaintext
@page {
    @top-center {
        content: "First part";
        color: red;
        text-align:right;
    }
    @top-left {
        content: "- Second part";
        color: blue;
        text-align:left;
    }
}
```
How to Simplify the Header (Keep Only the Chapter Title)

The headers display information such as map title, part title, chapter title, and section title, ending in the page number.

```css
.content: string(maptitle) string(parttitle) string(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 41), add the following CSS rule:

```css
@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:
  ```css
  @page :left, table-of-contents:left, chapter:left{
    ...
  }
  @page :right, table-of-contents:right, chapter:right{
    ...
  }
  ```

- Add an !important classifier just before the semi-colon.
  ```css
  @top-right {
    content: string(chaptertitle) " | " counter(page) !important;
  }
  ```

How to Change the Header Separators

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 a separator. In your customization CSS (on page 41), add the following CSS rule:

```css
*[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 ""; /* 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();
}

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 41):
If you intend to modify just the headers of the table of contents, use the `table-of-contents` page rule selector:

```xml
@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:

```xml
@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:

```xml
@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:

```xml
<bookmeta>
  <metadata>
    ...
    <data name="cover">
      <image href="product-cover.png" outputclass="cover-image"/>
    </data>
    ...
  </metadata>
</bookmeta>
```

```css
@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:

```css
@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:
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.

```css
: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:

```css
@page chapter, chapter:first: left: right, front-page{
  padding-top: 1em;
  @top-left {
    content: "Custom header";
    color: gray;
    border-bottom: 1px solid black;
  }
}
vertical-align: bottom;
}

@top-center{
    content: " ";
    border-bottom: 1px solid black;
    vertical-align: bottom;
}

@top-right{
    content: " ";
    border-bottom: 1px solid black;
    vertical-align: bottom;
}

@top-right-corner{
    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 64)*.

**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 215).

Next, alter your customization CSS to make use of the parameter value. In the example below, the text is placed in the central part of the header:

```css
@page front-page, table-of-contents, chapter {
  @top-center{
    content: oxy_xpath("//*/heading.text");
  }
}
```

**Note:**
You can use any XPath 2.0 here. It will be executed in the context of the merged map document, so you can collect data from it. You can use `if/then/else` expressions if your parameter is a switch.

The text does not affect the first pages from the page sequences because the built-in CSS page rules (on page 48) clear the content from the headers. If you need the text content on all pages, you might consider adding an `!important` keyword after the `content` property value, or increase the specificity of the page selectors, like this:

```css
@page front-page, 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 70). You can use this technique to alter the chapter titles as in the following example:

```css
*[class =~ "map/map"]*[numbering='deep']
  *[class =~ "topic/topic"]*[is-chapter]:not([is-part]) >
    *[class =~ "topic/title"] {
      string-set:
        chapter_title * " " counters(chapter-and-sections, ".") " "
      oxy_xpath("//*/heading.text") content(),
      section_title ""
    }
```
How to Change the Headings depending on the Language

It is possible to customize the text displayed in the headings depending on the language of the publication.

In this case, you can simply use the @lang attribute in your customization CSS. In the following example, the page counter displayed in the bottom part of the page is preceded by the word "Page", according to the selected language:

```css
@page chapter {
  @bottom-center {
    content: oxy_xpath("if (@lang='es') then 'Página' \
    else if (@lang='it') then 'Pagina' \
    else 'Page'") *" counter(page);
  }
}
```

Note: Backslashes (\) are used to split the XPath into multiple lines to make it easier to read.

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:

```css
*[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:

1. By creating an `@page` and assigning it to an element will create a page break between this element and the sibling elements that have a different page.


3. In your DITA topic, set the `@outputclass` attribute on the topic root (or any element) to contain one of the `page-break-before`, `page-break-after`, or `page-break-avoid` values. If you want to control the page breaking from the DITA map, use the `@outputclass` attribute on the `<topicref>`, with any of the values mentioned above.

Related Information:
- Double Side Pagination (on page 132)
- Oxygen PDF Chemistry: Controlling Page Breaks

Page Breaks - Built-in CSS

Page break properties are used in: `[/PLUGIN_DIR/css/print/p-page-breaks.css]`.

How to Avoid Page Breaks in Lists and Tables

To avoid splitting elements over two pages, you can use the `page-break-inside` CSS property. For example, if you want to impose this on tables and lists, then add the following rules to your customization CSS (on page 41):

```
*[class ~="topic/table"] {  
  page-break-inside:avoid;
}
*[class ~="topic/ol"] {  
  page-break-inside:avoid;
}
*[class ~="topic/ul"] {  
  page-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:
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 41):

```
*[class ~="map/map"] > *[class ~="topic/topic"] > *[class ~="topic/topic"] {  
    page-break-before:always;
}
```

If you need to break at third or fourth level topics, add more `.. > *[class ~="topic/topic"]` selectors to the expression.

If you want to force a page break for a specific topic, mark the topic (or any other element you need to control page breaking for) with an `@outputclass` attribute set to one of these values:

- **page-break-before**
  Use this for a page break before the marked element.

- **page-break-after**
  Use this for a page break after the marked element.

- **page-break-avoid**
  Use this to avoid page breaks inside the marked element.

For example, to force a page break before a certain topic, use:

```
<topic outputclass="page-break-before" ...>
```

**Note:**
You can set the output class on the `<topicref>` element from the DITA map instead of the `<topic>` element. In this way you can reuse the topic in another context where the page breaking is not necessary.

You can also control page breaking for lists, paragraphs, or any other block type elements. The following example avoids page breaks inside an ordered list:

```
<ol outputclass="page-break-avoid" ...>
```
How to Add a Blank Page After a Topic

If you want to add a new blank page after a topic, add the following rules to your customization CSS (on page 41).

Style the separating blank page:

```css
@page topic-separating-page{
    @top-left {
        content: "";
    }
    @top-right {
        content: "";
    }
    @top-center {
        content: "This page is blank";
    }
}
```

Associate this page to the `:after` pseudo-element of the topic:

```css
*[class~="topic/topic"][outputclass~="add-separator-page"]:after {
    content: " ";
    display: block;
    page: topic-separating-page;
}
```

In the XML content, on the `<topic>` element, set the `@outputclass` to the `add-separator-page` value.

```
<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 41):

```
: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 42) 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>
  <bookmeta xmlns:dita-ot="http://dita-ot.sourceforge.net/ns/201007/dita-ot" ...
  </bookmeta>
```
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.

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";
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:

```xml
<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
<?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 41)*, add the following:

```css
@page front-page {
    background-image: url("us-letter.svg");
    background-position: center;
    background-repeat: no-repeat;
}
```
For smaller artworks, you can use `background-position` with percentage values to position and center the artwork (for example, a company logo):

```css
@page front-page {
    background-image: url("company-logo.svg");
    background-position: 50% 5%; /* The first is the alignment on the X axis, the second on the Y axis. */
    background-repeat: no-repeat;
}
```

**Note:**
The text from the SVG or PDF background images is searchable in the PDF reader.

**How to 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:

```css
@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;
}
```
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.

```css
@page front-page {
  @page {page: main-title-page;}
  background-image: url(oxy_xpath("\
    if(/*[@class, ' topic/prodname ']()[1] = 'gardening') then 'bg-gardening.svg' else\n    if(/*[@class, ' topic/prodname ']()[1] = 'soil') then 'bg-soil.svg'\n    else 'bg-default.svg'\n  ");
  background-position:center;
}
```

The backslash (\) is used to continue the expression string on the subsequent lines (there should be no spaces after it). For more use cases solved using XPath, see: Metadata (on page 91).

Related Information:
- Oxygen PDF Chemistry: Graphics

How to Change Styling of the Cover Page Title

Match the front page title element in your customization CSS (on page 41) based on its class attribute:

```css
*[@class =~ "front-page/front-page-title"] {
  margin-top: 1in;
  font-size: 3em;
}
```

⚠️ Important:

Make sure the sum of the top and bottom margins and paddings for this element do not exceed the physical dimension of the page. If this happens, an extra blank page may appear before the cover page. Usually, it is enough to specify only the top margin.

How to Add Text to the Cover Page

If you need to add arbitrary text to the cover page, you can use the front page title element as an anchor and add as many blocks of text as you need after it, and style them differently.

In your customization CSS (on page 41), add the following:
The result is:

![Cover Page with Metadata]

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 97).

Related Information:
How to Show Metadata in the Cover Page (on page 97)

### How to Place Cover on the Right or Left Side

In your customization CSS (on page 41), add the following CSS rules:

```css
* [class ~="front-page/front-page"]{
  page-break-before:left;
}
```
Oxygen DITA-OT CSS-based PDF Publishing plugin 25.1 | 4 - Customizing PDF Output Using CSS | 86

**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 132)](#)

**How to Add a Second Cover Page and Back Cover Page**

It is possible to add a second cover page after the front-page by defining another page-selector:

```css
@page second-cover {
    @top-left {content: none;}
    @top-right {content: none;}
    @bottom-center {content: none;}

    background-image: url("second-cover.svg");
    background-position: center;
    background-repeat: no-repeat;
    background-size: 100% 100%;
}

*[class ~= 'front-page/front-page']:after{
    page: second-cover;
    page-break-after: always;
    display: block;
    content: "\2002";
}
```

If you want to add a back cover page, you should use an `:after` pseudo element on the map itself:

```css
*[class ~= "map/map"]:after
```

and bind it to another `@page` declaration:

```css
@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;
```
Note:
For any background-image, it is recommended to use SVG instead of PNG (or JPG) because it scales it to the page size.

Tip:
To add multiple cover pages, use multiple-leveled pseudo selectors, such as :after(1), :after(2). Remember that the larger the value, the more distant the pseudo element is to the target element.

How to Dynamically Add a Second Cover Page

It is possible to dynamically set the path to the SVG image that will be displayed on the secondary cover page.

First, you need to declare a <data> element in the <bookmap>’s metadata that contains the URL to your cover image:

```xml
<bookmap>
  <booktitle>
    ...
  </booktitle>
  <bookmeta>
    <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:

```css
@page second-cover {
  ...
```
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 41), add the following CSS rules:

```css
@page my-blank-page {
/* Hide the page numbers */
@top-left {content: none;}
@top-right {content: none;}
}
*[class ~='front-page/front-page']:after(1){
    page:my-blank-page;
    display:block;
    content: '\2002';
    color:transparent;
    page-break-after:always;
}
*[class ~='front-page/front-page']:after(2){
    page:my-blank-page;
    display:block;
    content: '\2002';
    page-break-after:always;
}
*[class ~='front-page/front-page']:after(3){
    page:my-blank-page;
    display:block;
    content: '\2002';
    page-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 134)

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 41), declare a new page layout:

```
@page copyright-notice-page {
  @top-left {
    content:none; /* Clear the headers for the copyright page */
  }
  @top-right {
    content:none;
  }
}
```

The element with the class `front-page/front-page` element contains the title of the publication and generates the cover page. A synthetic :after element is created that follows this element and it is placed on a different page.

```
*[class~="front-page/front-page"]:after{
  display:block;

  page: copyright-notice-page; /* Moves the synthetic element on a new page. */

  margin-top:90%; /* use margins to position the text in the page */
  margin-left: 5em;
  margin-right: 5em;

  content: "Copyright 2018-2019 MyCorp Inc. \A All rights reserved";

  text-align:center; /* More styling */
  color:blue;
}
```
If you need to add more content as blocks, use the :after(2), :after(3) pseudo-elements:

```css
*[@class~="front-page/front-page"]:after(2) {
    display: block;
    page: copyright-notice-page; /* Continue on the same page as the first ':after'. */
    content: "Some more styled text";
    color: red;
}
```

If you want to extract information from the document, use the oxy_xpath() function. For example, if the copyright info is stored in the map like this:

```xml
<map ...>
  <topicmeta>
    <copyright>
      <copyryear year="2018"/>
      <copyrholder>MyCorp Inc.</copyrholder>
    </copyright>
  </topicmeta>
  ...
</map>
```

then use this:

```css
*[@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")
    "\n    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 46)

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 41):
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:

```xml
@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%;
}
```

```css
:root::before {
  page: topic-cover;
  display: block;
  content: "\2002";
  page-break-after: always;
}
```

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 42), 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`:

```xml
         xmlns:opentopic-index="http://www.idiominc.com/opentopic/index" cascade="merge"
         class="- map/map bookmap/bookmap "
         ditaarch:DITAArchVersion="1.3">


    <bookmeta xmlns:dita-ot="http://dita-ot.sourceforge.net/ns/201007/dita-ot"
               class="- map/topicmeta bookmap/bookmeta">
      <author class="- topic/author ">Howe Tuduit</author>
      <bookid class="- topic/data bookmap/bookid">
        <isbn class="- topic/data bookmap/isbn">071271271X</isbn>
        <booknumber class="- topic/data bookmap/booknumber">SG99-9999-00</booknumber>
        <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">
        ...
      </bookrights>
    </bookmeta>

    <oxy:front-page-title>
      ...
    </oxy:front-page-title>
  </oxy:front-page>
</bookmap>
```

For the **DITA Map PDF - based on HTML5 & CSS** transformation type, the merged map is further processed resulting in a collection of HTML5 `<div>` elements. These elements preserve the original DITA `@class` attribute values and add a new value derived from the DITA element name.
Maps

The maps have a more simple structure, they use the `<topicmeta>` element for metadata sections. This is also a simplified example, as there may be many more elements in the metadata section:

```
     xmlns:opentopic-index="http://www.idiominc.com/opentopic/index"
     cascade="merge" class="- map/map"
     ditaarch:DITAArchVersion="1.3">
     ...
</map>


<topicmeta class="- map/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.

```
<prodinfo class="- topic/prodinfo">
  <prodnamedata class="- topic/prodname">oXygen PDF CSS DITA Plugin</prodnamedata>
</prodinfo>
```

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.
In the generated PDF, keywords are displayed in the Document Properties.

**Bookmaps**

If you want your keywords to appear inside a bookmap, you need to define them inside the `<bookmeta>` element:

```xml
<bookmap>
  ...
  <bookmeta>
    <keywords>
      <keyword>web server</keyword>
      <keyword>hard disk</keyword>
    </keywords>
  </bookmeta>
</bookmap>
```

**Maps**

If you want your keywords to appear inside a map, you need to define them inside the `<topicmeta>` element:

```xml
<map>
  ...
  <topicmeta>
    <keywords>
      <keyword>flowers</keyword>
      <indexterm>care and preparation</indexterm>
      <keyword>seasons</keyword>
    </keywords>
  </topicmeta>
</map>
```

**Topics**

If you want your keywords to appear inside one or more topics, you need to define them inside the `<prolog>` element:

```xml
<topic>
  ...
  <prolog>
    <metadata>
      <keywords>
        <indexterm>iris</indexterm>
      </keywords>
    </metadata>
  </prolog>
</topic>
```
Warning:
Keywords must be at map level or at topic level, you cannot combine them.

How to Add the Publication Audience to the Custom PDF Metadata

The audience element indicates the users the publication is addressing. This can be placed inside a `<topicmeta>` element in a `<map>` as in the following example:

```xml
<map>
  ...
  <topicmeta>
  ...
  <audience type="programmer" job="programming" experiencelevel="expert"/>
</map>
```

To collect the `@type` attribute, add the following in your customization CSS (on page 41):

```css
*[[class ~="map/map"] > *[class ~="map/topicmeta"] > *[class ~="topic/audience"] {  
  -oxy-pdf-meta-custom: "Audience" attr(type);
}
```

Notice:
It is best to use the class selector (such as `*[class ~="map/topicmeta"]`) instead of `topicmeta` to cover cases where the elements are specialized (for instance, in a bookmap the `bookmeta` is a `topicmeta`, so your selector will also function for bookmaps, not only simple maps).

Note:
The selector begins with `map >` to choose the `<topicmeta>` that is a direct child of the map, not other `<topicmeta>` elements from other `<topicref>` elements.

Tip:
You can define multiple key value pairs by separating them with commas:

```css
-oxy-pdf-meta-custom: "Audience" attr(type), "Job" attr(job)
```

The metadata is displayed in the Custom tab of the Document Properties dialog box from Acrobat Reader:
How to Show Metadata in the Cover Page

The following CSS extensions are used in the subsequent examples:

- **oxy_xpath** - Executes an XPath expression and returns string content. Use this whenever you need to extract data from an element other than the one matched by the CSS rule selector.

- **:after(N)** - Creates more than one after pseudo-element. The argument value represents how far the generated content is from the real content. For example, in the second code snippet in the next section (on page 98), the content of the :after is closer to the title (upper) than the content of the :after(2).

**Note:**

The `attr()` CSS function can also be used but it is limited to extracting attribute values from the matched element.
Processing Metadata for Bookmaps

Suppose you need to present the Author and the ISBN (when it exists) just under the publication title and suppose your bookmap contains:

```xml
<bookmap id="taskbook">
  <booktitle>
    <booklibrary>Retro Tools</booklibrary>
    <mainbooktitle>Product tasks</mainbooktitle>
    <booktitlealt>Tasks and what they can do</booktitlealt>
  </booktitle>
  <bookmeta>
    <author>Howe Tuduit</author>
    <critdates>
      <created date="1/1/2015"/>
      <revised modified="3/4/2016"/>
      <revised modified="3/5/2016"/>
    </critdates>
    <bookid>
      <isbn>071271271X</isbn>
      <booknumber>SG99-9999-00</booknumber>
    </bookid>
  </bookmeta>
</bookmap>
```

The entire `<booktitle>` element content is displayed on the first page of the PDF, so if you need to add the information after it, in your customization CSS (on page 41), add the following CSS rules:

```css
*[class ~="bookmap/booktitle"]:after {
  display: block;
  content: oxy_xpath('//*/[contains(@class, " bookmap/bookmeta ")]/*[contains(@class, " topic/author ")]//text()');
  margin-top: 4em;
  text-align: center;
  color: gray;
}
*[class ~="bookmap/booktitle"]:after(2) {
  display: block;
  content: oxy_xpath('if(//*[contains(@class, " bookmap/isbn ")]) then concat("ISBN ", //*[contains(@class, " bookmap/isbn ")]//text()) else ");
  text-align: center;
  color: gray;
}
```
Processing Metadata for DITA Maps

Suppose you need to present the **Revision Date** just under the publication title and suppose your DITA map contains:

```xml
<map>
  <title>Growing Flowers</title>
  <topicmeta>
    <critdates>
      <revised modified="2021-04-26"/>
    </critdates>
  </topicmeta>
</map>
```

The entire `<title>` element content is displayed on the first page of the PDF. If you need to add the information after it, add the following CSS rules in your **customization CSS** *(on page 41)*:

```css
*[class ~="front-page/front-page-title"] > *[class ~="topic/title"]:after {
  display: block;
  content: "last revision "oxy_xpath(''//*[contains(@class, "map/topicmeta ")]|1 "
  //*[contains(@class, "topic/revised ")]//@modified');
  margin-top: 4em;
  text-align: center;
  color: gray;
}
```

**Note:**
The `[1]` predicate is used to avoid duplicated results as the `topicmeta` is included in all children topics.

Generating Synthetic Pages for Metadata

Suppose you need to show this information on a page that follows the title page, instead of on the title page. In this case, you need to prepare a named page and place the content in it. Add the following rules in your **customization CSS** *(on page 41)*:

```css
@page page-for-meta {
  background-color: yellow; /* Just to see it better*/
  @top-left-corner {
    content:""; /* Remove the default header */
  }
  @top-right-corner {
    content:""; /* Remove the default header */
  }
}

*[class ~="bookmap/booktitle"]:after {
```
How to Show Metadata in the Header or Footer

The header and footer are composed of page margin boxes that can be populated with static text by using string-sets.

If you need to add some of the map metadata to the header of the front page (for example, the creation date), add the following CSS rules in your customization CSS (on page 41):

```css
* [class ~= "front-page/front-page"] >
  *[class ~= "map/topicmets"] >
    *[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 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 41):

```css
:root {
  -oxy-pdf-meta-keywords:"";
}
```
To change them, if you have a hard-coded list, you just enumerate each of them in the property content, separating them with comma:

```css
:root {
-oxy-pdf-meta-keywords:"alpha, beta, gamma";
}
```

If you need to extract them by other criteria from the merged map, you can use the `oxy_xpath()` function instead of the hard-coded list.

### How to Remove the PDF Publication Title Property

The title defined in the PDF reader is automatically collected from the map's main title.

If you want to display the map name instead of the title, you can use one of the following rules in your customization CSS *(on page 41)*:

```css
/*
* Titles (maps).
*/
*[class =~ "front-page/front-page-title"] *[class =~ "topic/title"]:not([class =~ 'bookmap/booktitle']) {
- oxy-pdf-meta-title: unset;
}
```

```css
/*
* Titles (bookmaps).
*/
*[class =~ "front-page/front-page"] *[class =~ "bookmap/booktitle"] > *[class =~ "bookmap/mainbooktitle"] {
- oxy-pdf-meta-title: unset;
}
```

### How to Change the PDF Publication Title Property

The `<title>` element of a bookmap is quite complex and contains elements for the book library and an alternate title:

```xml
<booktitle>
  <booklibrary>Retro Tools</booklibrary>
  <mainbooktitle>Main Book Title</mainbooktitle>
  <booktitlealt>Book Title Alternative</booktitlealt>
</booktitle>
```

For the publication title, the built-in CSS uses only the content of the `<mainbooktitle>`. If you want to collect all of the text from the `<booktitle>`, you can add the following rule to your customization CSS *(on page 41)*:
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.

```xml
<bookmeta>
  ....
  <data keyref="my_key"/>
  ....
</bookmeta>
```

This is expanded in the merged HTML file to:

```xml
<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:

```xml
* [class =~ "topic/data"] [keyref="my_key"] {
  string-set: key-string content (text);
}
```

```xml
@page {
  @bottom-left {
    content: "My key is: " string(key-string) !important;
  }
}
```
Or you can use the value from a :before pseudo-element, like the one for the title:

```css
*[class =~ "topic/title"]:before {
  content: oxy_xpath("//*[contains(@class, 'topic/data')][@keyref = 'my_key']//text()");
}
```

Another use-case is to use the key as a source for a custom PDF document property:

```css
*[class =~ "topic/data"][@keyref="my_key"] {
  -oxy-pdf-meta-custom: attr(keyref) content(text);
}
```

## How to Control the PDF Viewer

The PDF document may contain settings for the PDF Viewer. This helps to make the viewing experience common for all of the readers. For example, you can specify the zoom level that the document is presented, or whether the outline view should be displayed.

There are several CSS properties you can use. These properties should be set on the root element. If they are set on multiple elements, the first one will be taken into account.

### Examples

- To hide the PDF Viewer toolbar and menu bar:

  ```css
  :root { 
    -oxy-pdf-viewer-hide-menubar: true;
    -oxy-pdf-viewer-hide-toolbar: true;
  }
  ```

- To make the document be displayed with a different zoom level:

  ```css
  :root { 
    -oxy-pdf-viewer-zoom: 50%;
  }
  ```

- To make the PDF Viewer just as large as the displayed document (e.g. if there is a zoom level that makes the document smaller, then the window of the viewer will be just as big as the page):

  ```css
  :root { 
    -oxy-pdf-viewer-fit-window: true;
  }
  ```

- If you need the pages to be displayed as a single continuous column (to be able to scroll in a single view port), use:

  ```css
  :root { 
    -oxy-pdf-viewer-page-layout: one-column;
  }
  ```
The supported includes: `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 42)*, 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">
        ...
      </topicref>
      ...
    </frontmatter>
  </opentopic:map>
  ...
</bookmap>
```

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.
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 54)

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 41):

```html
*\[class \= \"map/map\"] > *\[class \= \"topic/topic\"]\[is-frontmatter\]{
  page-break-before: auto;
}
```

How to Style the Front Matter and Back Matter Topics

Style all the Topics with the Same Aspect

All the topics referenced from the <frontmatter> and <backmatter> bookmark elements are formatted using the matter-page as defined in Default Page Definitions (on page 48). In the merged file, the <backmatter> and
<frontmatter> elements are omitted, and their child topic content is matched using a CSS rule like the one below:

```css
*[class ~="map/map"] > *[class ~="topic/topic"] [is-backmatter],
*[class ~="map/map"] > *[class ~="topic/topic"] [is-frontmatter] {
    page: matter-page;
    ...
}
```

### Style the Topics Depending on Their Role

There might be cases when you need to distinguish between certain types of topics that have different roles in your publication:

- Preface
- Notice
- Abstract
- Copyright

These are referenced from the DITA map by specialized `<topicref>` elements, with different class attribute values.

The class attribute values are then passed by the transformation process onto the corresponding topic elements from the merged map content. For example, a topic that was referenced by a `<preface>` map element now has a "bookmap/preface" value in its `@topicrefclass` attribute:

```xml
<topic
    class="- topic/topic"
    id="unique_1"
    topicrefclass="- map/topicref bookmap/preface ".>
    ...
</topic>
```

This can be used to match and apply various styling choices, or even a particular page layout:

```css
@page preface-page {
    background-color:silver;
    @top-center{
        content: "Custom Preface Header";
    }
}
*
*{class ~="topic/topic"}[@topicrefclass ~="bookmap/preface"] {
    page: preface-page;
}
```
Numbering

The topics in this section contain some technical details in case you need to fine-tune the way the numbering works.

Numbering - Built-in CSS

The built-in CSS rules are in:

- [PLUGIN_DIR]/css/print/p-numbering-shallow.css
- [PLUGIN_DIR]/css/print/p-numbering-deep.css
- [PLUGIN_DIR]/css/print/p-numbering-deep-chapter-scope.css
- [PLUGIN_DIR]/css/print/p-numbering-deep-chapter-scope-no-page-reset.css

The first CSS (shallow) contains rules that add a "Chapter NN" before the first-level topics from the publication, the second one (deep) contains rules that add a deep structure of counters on all topics referenced from the map (at any level), the third one (chapter-scope) creates a chapter scope-oriented numbering (meaning that the numbering for pages, tables, figures, and links to them are reset for each chapter), and the last one is similar to the third except that page numbers do not reset. For more details, see Numbering Types (on page 111).

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.

```xml
<map xmlns:ditaarch="http://dita.oasis-open.org/architecture/2005/" ...>
                  class=" front-page/front-page">
    ...
  </oxy:front-page>
  <opentopic:map xmlns:opentopic="http://www.idiominc.com/opentopic" class=" toc/toc">
    <title class="- topic/title">Publication Title</title>
    ...
    <topicref is-chapter="true" class="- map/topicref">
      <topicmeta class="- map/topicmeta">
        <navtitle href="#unique_1" class="- topic/navtitle">Overview</navtitle>
        ...
      </topicmeta>
    </topicref>
    <topicref class="- map/topicref">
      <topicmeta class="- map/topicmeta" data-topic-id="dcpp_resources">
        <navtitle href="#unique_2" class="- topic/navtitle">Resources</navtitle>
```
Note:
The `<opentopic:map>` element contains the effective table of contents structure.

Note:
The TOC items are the elements with the class: `- map/topicref`.

Note:
The ones identified as chapters have the `@is-chapter` attribute set.

For the DITA Map PDF - based on HTML5 & CSS transformation type, the merged map is further processed resulting in a collection of HTML5 `<div>` elements. These elements preserve the original DITA `@class` attribute values and add a new value derived from the DITA element name.

```html
<div class="- map/map" ...>
  <div class=" front-page/front-page front-page">
    ...
  </div>

  <div class=" toc/toc toc">
    <div class="- topic/title title">
      Publication Title</div>

    <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-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.

For the **DITA Map PDF - based on HTML5 & CSS** transformations:

```xml
  <oxy:front-page-title>
    <title class="- topic/title ">Publication Title</title>
  </oxy:front-page-title>
</oxy:front-page>
```

The main content is organized as follows:

```xml
<map xmlns:ditaarch="http://dita.oasis-open.org/architecture/2005/" ...
```

```xml
<opentopic:map xmlns:opentopic="http://www.idiominc.com/opentopic">
```

```xml
<topic is-chapter="true" oid="dcpp_overview">
  <title class="- topic/title ">Overview</title>
  <body class="- topic/body ">
    ...
  </body>
</topic>
```

```xml
<topic class="- topic/topic " id="unique_2" oid="dcpp_resources">
  <title class="- topic/title ">Resources</title>
  ...
</topic>
```

```xml
<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:

```html
<div class="map/map" ...
...
</div>

```html
<div is-chapter="true" oid="dcpp_overview" class="- topic/topic">
  <div class="- topic/title">Overview</div>
  ...
</div>

```html
</div>

```html
<div class="- topic/topic" id="unique_2" oid="dcpp_resources">
  <div class="- topic/title">Resources</div>
  ...
</div>

```html
</div>

```html
</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:

```html
<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:

```html
<div class="- topic/topic topic" id="unique_2" oid="dcpp_parameters">
  <div class="- topic/title title ">
    Parameters
  </div>
  ...
</div>
```

**Note:**

The title elements have the class: `topic/title`. The actual element name can be different.

**Numbering Types**

The type of numbering that appears in your publication is controlled by the `args.css.param.numbering` parameter.

This parameter activates various sets of CSS rules from the built-in CSS. By default, only the first-level topics (the chapters) are numbered (shallow numbering). The following values are accepted:

<table>
<thead>
<tr>
<th>Table 1. Types of Numbering</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value</td>
</tr>
<tr>
<td>------</td>
</tr>
<tr>
<td>shallow</td>
</tr>
<tr>
<td>deep</td>
</tr>
<tr>
<td>deep-chapter-scope</td>
</tr>
<tr>
<td>deep-chapter-scope-no-page-reset</td>
</tr>
</tbody>
</table>
Note:
When using any of the deep numbering types, no distinction is made between sections and nested topics. For example, if a topic contains two sections, followed by another nested topic, the sections will be numbered with 1 and 2, and the nested topic with 3.

Notice:
The `deep-chapter-scope` and `deep-chapter-scope-no-page-reset` values are only available for the DITA Map PDF - based on HTML5 & CSS transformation scenario.

Examples

Shallow

Each chapter (or first-level topic) is numbered, but sections/nested topics are not numbered. Figures, tables, and pages are numbered sequentially from the start of the publication and they do not reset.

<table>
<thead>
<tr>
<th>Chapter 1. First Chapter</th>
<th>Page 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Topic</td>
<td></td>
</tr>
<tr>
<td>Section</td>
<td></td>
</tr>
<tr>
<td>Table 1</td>
<td></td>
</tr>
<tr>
<td>Table 2</td>
<td></td>
</tr>
<tr>
<td>Topic</td>
<td></td>
</tr>
<tr>
<td>Section</td>
<td></td>
</tr>
<tr>
<td>Page 2</td>
<td></td>
</tr>
<tr>
<td>Table 3</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Chapter 2. Second Chapter</th>
<th>Page 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Topic</td>
<td></td>
</tr>
<tr>
<td>Table 4</td>
<td></td>
</tr>
<tr>
<td>Table 5</td>
<td></td>
</tr>
<tr>
<td>Topic</td>
<td></td>
</tr>
</tbody>
</table>

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 |
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
...
Deep Chapter Scope

Each chapter (or first-level topic) is independent (so it can be read separately, as a separate part of your publication). The sections/nested topics, pages, figures, and table counters (and links to them) restart at each chapter. The general cross reference links also display the chapter number before the page number to clearly specify the target.

1. First Chapter
   Page 1.1
   Topic 1.1
   Table 1-1
   Link to page 2.2
   Topic 1.2
   Page 1.2
   Table 1-2
2. Second Chapter
   Page 2.1
   Topic 2.1
   Table 2-1
   Table 2-2
   Table 2-3
   Topic 2.2
   Table 2-4
   Page 2.2
   Link to page 1.1

It will result in the following content inside the PDF:

1. Introduction......................................................................1
2. Care and Preparation..............................................................1
   2.1. Pruning.......................................................................1
   2.2. Garden Preparation............................................................2
3. Flowers by Season..................................................................1
   3.1. Spring Flowers.................................................................1
       3.1.1. Iris......................................................................1
       3.1.2. Snowdrop.................................................................3
   ...

List of Figures
   Figure 3-1: Iris
Deep Chapter Scope No Page Reset

Each chapter (or first-level topic) is independent (so it can be read separately, as a separate part of your publication). The sections/nested topics, figures, and table counters (and links to them) restart at each chapter, but the page numbers do not reset. The generic cross reference links contain only the page number.

1. First Chapter
   Page 1
   Topic 1.1
   Table 1-1
   Link to page 4
   Topic 1.2
   Page 2
   Table 1-2

2. Second Chapter
   Page 3
   Topic 2.1
   Table 2-1
   Table 2-2
   Table 2-3
   Topic 2.2
   Table 2-4
   Page 4
   Link to page 1

It will result in the following content inside the PDF:

1. Introduction......................................................................1
2. Care and Preparation..............................................................2
   2.1. Pruning.......................................................................2
   2.2. Garden Preparation............................................................3
3. Flowers by Season.................................................................4
   3.1. Spring Flowers.................................................................4
      3.1.1. Iris......................................................................4
      3.1.2. Snowdrop.................................................................6
   ...

List of Figures
   Figure 3-1: Iris
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 117).

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 41):

```css
*[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 41):

```css
*[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 41):

```css
*[^class ~="map/map"][^numbering ~= 'deep'] > *[^class ~= "topic/topic"][^is-chapter] {
  counter-reset: page section1 tablecount figcount !important;
}
*[^class ~="map/map"][^numbering ~= 'deep'] > *[^class ~= "topic/topic"][^is-part] {
  counter-reset: page 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 111). Suppose you have a link in the third chapter that points to a paragraph in the second subtopic of the first chapter and you need this structural information (1.2) presented to the user, just after the link text. To do this, you can use the `target-counters` CSS function to extract the entire context of the counters from the target. The `chapter-and-sections` built-in counter is already updated with both the chapter number and the nested topics:

```css
[^class ~= "topic/xref"]::after {
  content: target-counters(attr(href), chapter-and-sections, ".") !important;
}
```

This counter does not include the part number, so be careful when linking between parts (consider adding the target part number explicitly):

```css
[^class ~= "topic/xref"]::after {
  content: 
  "[" target-counter(attr(href), part, upper-roman) "/" target-counters(attr(href), chapter-and-sections, ".") "]" !important;
  color:blue;
}
```

Related Information:
Numbering Types (on page 111)

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 15) to yes. In this case, they are numbered according the numbering scheme set by the `args.css.param.numbering` parameter (on page 111).

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 125)*
- List of Tables/Figures *(on page 130)*
- Index *(on page 139)*

Table of Contents - XML Fragment

In the merged map file *(on page 42)*, 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>

  ...
```
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.
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 54)

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 120)) that hides topics on higher levels is:

```css
/* 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:

```css
*[class =~ "map/topicref"]{is-chapter} >
*[class =~ "map/topicref"]:not([is-chapter]) >
*[class =~ "map/topicref"] >
```
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:

```css
* [class ~= "map/topicref"] {
    counter-increment: toc-chapter-and-sections;
}
```

In your customization CSS (on page 41), add the following two selectors:

```css
* [class ~= "map/topicmeta"] + [class ~= "map/topicref"] {
    counter-reset: toc-chapter-and-sections;
}
```

How to Style the Table of Contents Entries

**Note:**

Each of the items from the table of contents is an element that has the `map/topicref` class.

The following example uses the italic font for the label and changes the color and style of the connecting line between the title and the page number.

In your customization CSS (on page 41), add the following two selectors:
And if you need to alter the indent of the nested table of content items, use the following selector:

```css
*[class ~="map/topicref"] *[class ~="map/topicref"] {  
  margin-left: 1em;
}
```

The numbers can be styled like this:

```css
*[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.

```css
*[class ~="map/topicref"]@[is-chapter ="true"] > *[class ~="map/topicmeta"] > *[class  
  ~="topic/navtitle"][
    font-size:2em;
}
```

**How to Change the Header of the Table of Contents**

In the built-in CSS, there is a page named `table-of-contents`. The default is to have the word 'Contents' in its header (this is localized, using the `toc-header` string defined in the `p-18n.css`) alternating in the left or right side of the header:

```css
@page table-of-contents:left {  
  @top-left {  
    content: string(toc-header) " | " counter(page, lower-roman);
    font-size: 8pt;
  }
}
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 41):

```css
@page table-of-contents: right {
  @top-right {
    content: string(toc-header) " | " counter(page, lower-roman);
    font-size: 8pt;
  }
}
```

Important:

The first page from the table of contents does not have any content displayed in the header. The default CSS contains rules that disable the content. If you need to also display the numerals on the first page, use the following:

```css
@page table-of-contents:first: left {
  @top-left {
    content: string(toc-header) " | " counter(page, lower-roman);
  }
}
```

```css
@page table-of-contents:first: right {
  @top-right {
    content: string(toc-header) " | " counter(page, lower-roman);
  }
}
```

Related information

Localization (on page 254)
How to Make the Table of Contents Start on an Odd Page

In your customization CSS (on page 41), add the following snippet for the table-of-contents page:

```css
@page table-of-contents {
    -oxy-initial-page-number: auto-odd;
}
```

Related Information:
Double Side Pagination (on page 132)

How to Display a Topic Before the Table of Contents

To display a topic before the table-of-contents page, follow these steps:

1. Make sure the topic is referenced on the first level in the DITA map.
2. Set the @outputclass to before-toc on the <topicref>.

```xml
<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 218)

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 41), add the following CSS selector:

```css
* [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.
How to Remove Entries from the TOC

To remove entries from the table of contents, set the `@toc="no"` attribute on the `topicrefs` from the map that need to be removed. This is sometimes desirable for the topics listed in the frontmatter or backmatter when using a bookmap.

How to Hide the TOC

To hide the TOC, you have multiple options:

- [Recommended] Use a DITA `<bookmap>` instead of a `<map>`, and omit the `<toc>` element from the `<booklists>`. An example bookmap can be found in the DITA 1.3 Spec.
- Use the transformation parameter: `hide.frontpage.toc.index.glossary (on page 20)`. 
- Use a `display:none` property to hide the element that contains the TOC structure, and also remove it from the PDF bookmarks tree:

```css
*[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 15)

Table of Contents on a Page (Mini TOC)

To add a mini table of contents for each chapter, you need to:

- Use DITA bookmaps instead of regular maps.
- Set the `args.chapter.layout` transformation parameter to either of the following values: **MINITOC** or **MINITOC-BOTTOM-LINKS**.

**Note:**
If the chapter does not have child topics, it will not have a mini TOC in the PDF output.

**Layout for MINITOC**

This table of contents is positioned between the chapter title and the chapter child topics. It consists of a list of links pointing to the child topics, positioned in the left side of the page, and a description in the right side. This content is collected from the topic file referenced by the chapter `topicref` in the map.
Chapter 1. Introduction

Topics:

About this framework.

Description

DITA Open Toolkit, or DITA-OT for short, is a set of Java-based, open-source tools that provide processing for content authored in the Darwin Information Typing Architecture.

The DITA Open Toolkit documentation provides information about installing, running, configuring and extending the toolkit.

About this framework.

The framework is DITA.

Description

The framework is composed by a large set of modules.

Layout for MINITOC-BOTTOM-LINKS

This table of contents is positioned between the chapter title and the chapter child topics. It consists of a chapter description and list of links pointing to the child topics, under the description. This description is collected from the topic file referenced by the chapter <topicref> in the map.
Chapter 1. Introduction

DITA Open Toolkit, or DITA-OT for short, is a set of Java-based, open-source tools that provide processing for content authored in the Darwin Information Typing Architecture.

The DITA Open Toolkit documentation provides information about installing, running, configuring and extending the toolkit.

Topics:

About this framework.
Description

About this framework.
The framework is DITA.

Description
The framework is composed by a large set of modules.

The above chapter example has the following DITA map fragment:

```
<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.

```xml
<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>
  <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>
```

When using the `pdf-css-html5` transformation, this structure is converted to a set of HTML elements, preserving the class values:

```html
<div class="--topic/div chapter/minitoc div minitoc">
  <div class="--topic/div chapter/minitoc-links div minitoc-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">Topics: </span>
        </div>
      </div>
    </nav>
  </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 41), 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 45)

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):
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:

```css
*[^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:

```css
@page figurelist { 
    @top-left { content: none; } 
    @top-center { content: "Figure List"; } 
    @top-right { content: none; } 
}

@page figurelist:first {
    @top-left { content: none; } 
    @top-center { content: none; } 
    @top-right { content: none; } 
}

@page tablelist { 
```
How to Remove the Numbers Before a List of Tables or Figures

Suppose you need to remove the "Figure NN" prefix before each entry of a list of figures.

An entry in the generated list of figures from the merged map looks like this:

```
<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.
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 41).

**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 41), add the following:

```css
@page chapter {
  -oxy-initial-page-number: auto-odd;
}
@page table-of-contents {
  -oxy-initial-page-number: auto-odd;
}
```

Supported values for `-oxy-initial-page-number` include: `auto`, `auto-even`, `auto-odd`, or a number.

**How to Style the Empty (Blank) Pages**

By making the chapters start on an odd page, the CSS processor might add blank pages to the previous page sequence as padding.

To style those blank pages add the following code in your customization CSS (on page 41):

```css
@page chapter:blank, table-of-contents:blank {
  @top-left      { content: none; }
  @top-center    { content: none; }
  @top-right     { content: none; }
  @bottom-left   { content: none; }
  @bottom-center { content: none; }
  @bottom-right  { content: none; }
}
```

**Note:**

This just removes the headers and footers, but you can use a background image or a header with "Intentionally left blank" text.
How to Force an Odd or Even Number of Pages in a Chapter

Another use case is to specify a number of pages for a section. Suppose that you have a table of contents that follows the cover page and you need to have an even number of pages. Hence, the next chapter would start on an even page.

In your customization CSS (on page 41), use the -oxy-force-page-count property with an even value:

```css
@page table-of-contents {
  -oxy-force-page-count: even;
}
```

Supported values for -oxy-force-page-count include: even, odd, end-on-even, end-on-odd, auto, no-force.

How to Style the First page of a Chapter

You can use the :first page rule selector to control how the first page of a chapter looks. Suppose that you have defined the following layout for your default page and you want to put the publication title (the `maptitle` 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 41), add the following:

```css
@page chapter:first {
  @top-right-corner { content: string(maptitle); }
  @top-left { content: none; }
}
```

Multiple Column Pages

This section contains information about how to handle pages that have multiple columns.

How to Use a Two Column Layout

Change Layout for Predefined Pages

First, you need to identify which of the pages need to be changed. Pages are already defined for the cover page, table of contents, chapter content, and others. The complete list is here: Default Page Definitions (on page 48).

Next, add the column-count and column-gap properties to that page. For example:

```css
@page chapter{
  column-count:2;
}
```
If you need some of the elements to expand on all the columns, use the `column-span:all` CSS property. The next snippet makes the chapter titles span both columns:

```css
*[class ~="topic/topic"] &[is-chapter] > *[class ~="topic/title"] {
  column-span:all;
}
```

**Limitation:**

You cannot use multiple column configurations on the same page. Oxygen DITA-OT CSS-based PDF Publishing plugin 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.

   ```html
   <topic outputclass="two_columns" ...
   
2. Define a CSS rule that changes the `page` property for the matching element.

   ```css
   *[class ~="two_columns"],
   *[outputclass ~="two_columns"] {
     page: two_column_page !important;
   }
   ```

   **Tip:**
   
   In the selector, use the `class` attribute for the HTML transformation, or `outputclass` for the direct transformation, or leave them both if you are not sure.

   **Note:**
   
   The topics from the first level use the `chapter` page. You must use `!important` because the built-in rules are more specific and you need to override the `page` property.

3. Define a page layout.

   ```css
   @page two_column_page {
     column-count: 2;
   }
   ```

   Note that the topic will be separated from other sibling topics with different page layouts by page breaks.
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:

```xml
<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:

```css
@page {
  column-count: 2;
}
[class ~~ "topic/topic"] *[class ~~ "topic/title"] > *[class ~~ "topic/topic"] {
  -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

PDF Bookmarks

The PDF Bookmarks are used to generate a hierarchical structure similar to a table of contents in a specialized view of your PDF Reader.

By default, the titles defined in the topics are used as bookmark labels.

PDF Bookmarks - Built-in CSS

The PDF bookmarks are generated by matching the titles from the topics in the content. The built-in CSS rules are in: [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 41):

```
*{[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/title"],
*[class="topic/topic"] *[class="topic/topic"] *[class="topic/topic"] *[class="topic/topic"]
 *[class="topic/topic"] *[class="topic/topic"] > *[class="topic/title"]
```
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 41):

```css
*[,class~="topic/topic"] > *,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 41).

For example, to specify that all bookmarks for the first three levels are opened (expanded) in the initial state, use:

```css
*[,class~="topic/topic"] > *,class~="topic/title",
*[,class~="topic/topic"] *[class~="topic/topic"] > *,class~="topic/title",
*[,class~="topic/topic"] *[class~="topic/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 41)*, add the following CSS rules:

```css
*[class~="topic/topic"] > *[class~="topic/title"] {  
  bookmark-label: content(text);  
  -ah-bookmark-label: content();  
}
```

⚠️ **Important:**
This is a simple example that does not use the possible navigation titles, just the content of the `<title>` element. Copy and modify the built-in CSS for the full CSS rule that matches the `<title>` and `<titlealts>` elements:

```css
*[class~="topic/topic"]:has(*[class~="topic/titlealts"])) > *[class~="topic/title"] {...}
```

Related Information:
Numbering *(on page 107)*

## Index

The content of an `<indexterm>` element is used to produce an index entry in the generated index. You can nest `<indexterm>` elements to create multi-level indexes. The content is not output as part of the topic content, only as part of the index tree.

To add an index to your publication, you just need to add `<indexterm>` elements inside the `<prolog>` section (inside a `<metadata>` element):

```xml
<title>The topic title.</title>

<prolog>
  <metadata>
    <keywords>
      <indexterm>Installing <indexterm>Water Pump</indexterm></indexterm>
    </keywords>
  </metadata>
</prolog>

<body>
  ....
</body>
```

or in the content itself:

```xml
...  
<p>Open the lid then turn the body pump to the right.</p>  
<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>`):

```xml
<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 42), the structure that holds the index tree is the `<opentopic-index:index.groups>` element.

```xml
<map class="- map/map ">
  <oxy:front-page>
    ...
  </oxy:front-page>
  <opentopic:map xmlns:opentopic="http://www.idiominc.com/opentopic">
    ...
  </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.
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.

```xml
<div class="- map/map map" >
    <div class="front-page/front-page">
    ... 
</div>
```
The index group content becomes:

```html
<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>

  <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>
```

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 41), add the following CSS rules:

```css
*[^class~="index/groups"]*:before {
    content: "- Index - ";
    color: navy;
    font-size: 4em;
}
```

The result is:

- Index -

  F
  footer 37
  H
  header 37
  T
  table of contents 32
  change header 35
  style 34

How to Style the Index Terms Labels

In your customization CSS (on page 41), add the following CSS rule:

```css
^[^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 41), add the following CSS rule:

```css
/* Add filling dots between index labels and page numbers. */
*[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;
}

/* Hide the sequences of links that actually do not contain links. */
*[class="index/group"] *[class="index/entry"] > *[class="index/refid"]:has(*[class="index/link"]){
    display:inline;
}

/* Justify the index entries. */
*[class="index/group"] *[class="index/entry"] {
    text-align:justify;
}

/* Add dots before the index entry. */
*[class="index/group"] *[class="index/entry"] > *[class="index/refid"]:before{
    content:leader('.',");
}
```

The output now contains the dots:
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 41):

```
@page index {
    @bottom-center  {  content: counter(page, decimal)  }
}
```

How to Impose a Table-like Index Layout

In case you need to place the index labels and links on the same line but with some extra alignment constraints, you can use inline blocks to give the index a table-like appearance:
You need to place the elements that have the following class on the same line:

**index/formatted-value**

This is the text of the index term.

**index/refid**

This element contains a list of links.

A fixed width is used for the formatted value and the links container (almost half of the available width). To achieve the index hierarchical layout, set progressive padding to the formatted value text.

In your customization CSS *(on page 41)*, add the following CSS rule:

```css
*\[class~="index/formatted-value"\],
*\[class~="index/refid"\]{
    display:inline-block;
}

*\[class~="index/formatted-value"\] {
    width:45%;
}

*\[class~="index/refid"\] {
    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 indexterm 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;
}

To avoid bleeding of the index term label, you may need to mark it as being hyphenated:
To activate hyphenation, see: How to Enable Hyphenation for Entire Map (on page 157).

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 bookmark 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/>
  <appendix/>
</appendices>
```

**Warning:**
If the `<appendices>` element is not defined and the `<appendix>` is used directly instead, then it will behave like a `Part` or `Chapter` using the same pattern as for `<appendices>`.
How To Control Page Break Within Appendices

If you define a bookmap with `<appendices>` and some `<appendix>` elements, you may want the parent `<appendices>` to be on a separate page than its children. This is done automatically if the bookmap contains `<part>` elements. Otherwise, you may need to use the following in your CSS:

```
*{topicrefclass =~ "bookmap/appendix"}:first-of-type {
  page-break-before: always;
}
```

Footnotes

Footnotes are pieces of information 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 41)*:

```
*{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 41)*:

```
*{class =~ "topic/entry"} > *{class =~ "topic/fn"} {
  padding-left: 1in;
}
```
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.

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 41):

```
@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:

```
*[class = "topic/entry"] > *[class = "topic/fn"]:footnote-marker {
  margin-left: 1in;
}
```
**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.
2. Link the folder associated with the publishing template to your current project in the Project view.
3. Using the Project view, create an xslt folder inside the project root folder.
4. In the newly created folder, create an XSL file (for example, named `merged2mergedExtension.xsl`) with the following content:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    xmlns:xs="http://www.w3.org/2001/XMLSchema"
    xmlns:opentopic-func="http://www.idiominc.com/opentopic/exsl/function"
    exclude-result-prefixes="xs opentopic-func"
    version="2.0">

  <!-- 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>
  </xsl:template>

  <!-- 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. -->
    </xsl:for-each>
  </ol>
</xsl:stylesheet>
```
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]="/>
  <!-- There is a reference in the table, recalculate index. -->
  <xsl:variable name="fn-number" select="index-of($top-level-table//*[contains(@class, 'topic/fn')], $fn)"/>
  <xsl:copy>
    <xsl:apply-templates select="@*"/>
    <xsl:apply-templates select="node() except *[contains(@class, 'hi-d/sup')]"/>
    <sup><xsl:apply-templates select="child::*[contains(@class, 'hi-d/sup')]/@*"/></sup>
    <xsl:value-of select="$fn-number"/>
  </xsl:copy>
</xsl:when>
```

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')], $fn)"/>
      <xsl:copy>
        <xsl:apply-templates select="@*"/>
        <xsl:apply-templates select="node() except *[contains(@class, 'hi-d/sup')]"/>
        <sup><xsl:apply-templates select="child::*[contains(@class, 'hi-d/sup')]/@*"/></sup>
        <xsl:value-of select="$fn-number"/>
      </xsl:copy>
    </xsl:when>
  </xsl:choose>
</xsl:template>
```
5. Open the template descriptor file (on page 29) 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:

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;
}

/* 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) ");
}

/* Customize xrefs pointing to footnotes with IDs. */
*[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 29) associated with your publishing template (the .opt file) and reference your custom CSS file in the resources element:

<publishing-template>
  ...
</publishing-template>
8. Edit the DITA Map PDF - based on HTML5 & CSS transformation scenario.
9. In the Templates tab, click the Choose Custom Publishing Template link and select your template.
10. Click OK to save the changes and run the transformation scenario.

Hyphenation

Hyphenation specifies how words should be hyphenated when text wraps across multiple lines.

The transformation plugin uses the capabilities of the PDF Chemistry processor to perform hyphenation.

Hyphenation Dictionaries

The Oxygen DITA-OT CSS-based PDF Publishing plugin provides built-in hyphenation patterns for the following languages:

<table>
<thead>
<tr>
<th>Code</th>
<th>Language</th>
</tr>
</thead>
<tbody>
<tr>
<td>da</td>
<td>Danish</td>
</tr>
<tr>
<td>de</td>
<td>German</td>
</tr>
<tr>
<td>de_CH</td>
<td>German (Switzerland)</td>
</tr>
<tr>
<td>en</td>
<td>English</td>
</tr>
<tr>
<td>en-GB</td>
<td>English (Great Britain)</td>
</tr>
<tr>
<td>es</td>
<td>Spanish</td>
</tr>
<tr>
<td>fr</td>
<td>French</td>
</tr>
<tr>
<td>it</td>
<td>Italian</td>
</tr>
<tr>
<td>nb</td>
<td>Norwegian Bokmål</td>
</tr>
<tr>
<td>nl</td>
<td>Dutch</td>
</tr>
<tr>
<td>ro</td>
<td>Romanian</td>
</tr>
<tr>
<td>ru</td>
<td>Russian</td>
</tr>
<tr>
<td>sv</td>
<td>Swedish</td>
</tr>
<tr>
<td>th</td>
<td>Thai</td>
</tr>
<tr>
<td>pt</td>
<td>Portuguese</td>
</tr>
<tr>
<td>Code</td>
<td>Language</td>
</tr>
<tr>
<td>------</td>
<td>----------</td>
</tr>
<tr>
<td>da</td>
<td>Danish</td>
</tr>
</tbody>
</table>

The built-in hyphenation pattern license terms are listed in the XML files in the `CHEMISTRY_INSTALL_DIR/config/hyph` folder. Most of them comply with the \LaTeX\ distribution policy.

### Installing New Hyphenation Dictionaries

Oxygen DITA-OT CSS-based PDF Publishing plugin uses the \TeX\ hyphenation dictionaries converted to XML by the OFFO project: [https://sourceforge.net/projects/offo/](https://sourceforge.net/projects/offo/).

The `.xml` files allow you to access the licensing terms and you can use them as a starting point to create customized dictionaries (see How to Alter a Hyphenation Dictionary (on page 156)).

The `.hyp` files are the compiled dictionaries that the Oxygen DITA-OT CSS-based PDF Publishing plugin actually uses.


One simple way to add more dictionaries:

1. Download and extract the `offo-hyphenation-compiled.zip` file. This file is a bundle of many dictionary files.
3. If you just need a single dictionary, place the `.hyp` or `.xml` file in the `DITA_OT_DIR/plugins/com.oxygenxml.pdf.css/lib/oxygen-pdf-chemistry/config/hyph` directory.

### How to Alter a Hyphenation Dictionary

The hyphenation dictionaries are stored as XML files in the `OPE_INSTALL_DIR/plugins/com.oxygenxml.pdf.css/lib/oxygen-pdf-chemistry/config/hyph` directory.

You can copy the dictionaries you need to change in another directory, then use the `-hyph-dir` parameter to refer them inside your transformation.

Each file is named with the language code and has the following structure:

```xml
<hyphenation-info>
  <hyphen-min before="2" after="3"/>
  <exceptions>
    o-mni-bus
  </exceptions>
</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 41), 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:
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:

   ```xml
   <codeblock outputclass="hyphens">
       ...
   </codeblock>
   ```

   If you want to disable hyphenation inside tables:

   ```xml
   <table outputclass="no-hyphens">
       ...
   </table>
   ```

   **Note:**

   The default built-in CSS enables hyphenation for tables:

   ```
   *[class =~ "topic/table"] {
       hyphens: auto;
   }
   ```
How to Define Hyphenation for a Specific Word

1. Create a new hyphenation dictionary (on page 156).
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.

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:

```html
<table>
  <row>
    <cell>Product-1233-55-88</cell>
  </row>
  <row>
    <cell>120</cell>
  </row>
</table>
```
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 41). For instance, you can use:

```css
body { font-family: Arial }

@page {
    @top-left {font-family: Arial }
    @top-right {font-family: Arial }
    @top-center {font-family: Arial }
    @top-left-corner {font-family: Arial }
    @top-right-corner {font-family: Arial }
    @bottom-left {font-family: Arial }
    @bottom-right {font-family: Arial }
}
```
2. Create a new transformation scenario, based on the DITA Map PDF - based on HTML5 & CSS built-in scenario.
3. In the Parameters tab, change the value of the `pdf.accessibility` parameter to `yes`.
4. Run the transformation.

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 41). For instance, you can use:

```css
@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.

## Fonts

Fonts are an important part of the publication. Your font selection should take into consideration both design and the targeted ranges of characters.

**Note:**

Before using a font, make sure you have the rights to use it, and you comply with all license terms.

To use them in the customization CSS *(on page 41)*:

- You can place the font files in the same folder as your CSS and use a `@font-face` definition to reference them.
- You can use web fonts (for example, Google Fonts), and import the CSS snippet into your CSS.
- You can use system fonts.

All these techniques are explained in: *Oxygen PDF Chemistry User Manual: Fonts*.

### How to Avoid Characters Being Rendered as #

When the processor renders text with a font that does not include certain characters, those characters are replaced with the `#` symbol. This can easily be seen from *Oxygen's Results* view. For example:

<table>
<thead>
<tr>
<th>Glyph</th>
<th>Codepoint</th>
<th>Available in Font</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>?”</code></td>
<td>0x7ae0</td>
<td>Roboto-Regular</td>
</tr>
</tbody>
</table>

To prevent this, make sure you use the proper font.

As an example, suppose the right arrow character is used in a definition list like this:

```xml
<dlentry>
  <dt>&#8594;</dt>
  <dd><ph>This is the right arrow.</ph></dd>
</dlentry>
```

If the font does not include this character, the output will look something like this:

```
# This is the right arrow.
```

To fix this you can either:

1. Install Arial Unicode MS font on your system. This is one of the PDF processor fallback fonts. Starting with version 24 there are additional fonts used as fallback as *SimSun*, *Malgun Gothic* and more, so if one of these are found on the system they will be used directly.
2. Specify the fallback font in your customization.
For the second case, if you use *Times New Roman* for the entire publication, you could add *Symbol* as the fallback font. In your customization CSS *(on page 41)*, add:

```css
*[class ~="topic/dlentry"] {  
  font-family: "Times New Roman", Symbol;
}
```

To change the font for the entire publication, not just an element, you can use:

```css
:root {font-family: "Times New Roman", Symbol !important; }

@page {  
  @top-left {font-family: "Times New Roman", Symbol !important; }
  @top-right {font-family: "Times New Roman", Symbol !important; }
  @top-center {font-family: "Times New Roman", Symbol !important; }
  @top-left-corner {font-family: "Times New Roman", Symbol !important; }
  @top-right-corner {font-family: "Times New Roman", Symbol !important; }
}
```

**Tip:**

On Windows, one simple way to determine the font needed to display the text is to copy the text fragment that has rendering problems from the DITA source document and paste it into Microsoft WordPad or Word. It will automatically select a font capable of rendering the text. Simply click on the text to see the name of the font from the "Font" ribbon toolbar. Then you can use it as a fallback font in the CSS. Make sure there are no licensing restrictions on that particular font.

**Note:**

It is also possible to use a generic family name as fallback, like *serif*, *sans-serif* or *monospace*, like this you will call upon the processor default fallback fonts system.

**Warning:**

Even if the message is a warning, sometimes it can lead to a failed transformation. This usually occurs for really special characters (different from letters or common characters).
How to Set Fonts in Titles and Content

Suppose that in your customization CSS (on page 41), 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 41), add an !important rule that associates a font to all the elements from the document:

```css
* {
  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 41) should contain the following rule:

```css
*[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 42).

For example, if you want the titles or the pre-formatted text to have a different font from the rest, matched by the above * selector, you need to use more specific CSS selectors:

```css
*[class="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 58).

Related information
How to Change the Font of the Headers and Footers (on page 58)

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:

```css
font-family: Calibri, SimSun, "Malgun Gothic", "Microsoft JhengHei";
```

To apply this font sequence, see: How to Set Fonts in Titles and Content (on page 164).

Some of the Asian fonts do not have italic, bold, or bold-italic variants. In this case, you may use the regular font file with multiple font face definitions to simulate (synthesize) the missing variants. You need to use the -oxy-simulate-style:yes CSS property in the font face definition as explained in: Using Simulated/Synthetic Styles in Oxygen Chemistry.

How to Use Asian Fonts in Linux

For Asian languages on Linux distributions, PDF Chemistry automatically uses DejaVu and Noto CJK as fallback fonts for Serif, Sans-Serif, and Monospace content.

Warning:
On some distributions, the Noto CJK fonts are not available. In this case, you need to install them using the system package manager:

- fonts-noto-cjk on Debian family distributions (e.g. Ubuntu).
- google-noto-cjk-fonts on Red Hat family distributions (e.g. CentOS).

How to Add a New Asian Font

If you want to add a specific font for Asian languages, you need to declare it inside your customization CSS (on page 41). The following example uses the Noto Sans Tamil font-family:
How to Set Fonts for Displaying Music

Music is rendered as normal text in most fonts, but some of them will render them as musical glyphs. For example, the MusGlyphs font converts the text to music and adjusts it to the surrounding text.

This font is divided in two sub-fonts that act for each of the following categories:

- **MusGlyphs** - Converts all characters that match a musical pattern into music glyphs. It should be used inside the elements that contain only music.
- **MusGlyphs-Text** - Converts only the text prefixed with the @ symbol into music glyphs. The remaining text is displayed normally.

To use this font, you simply need to declare each sub-font then use them in appropriate elements:
Comments, Highlights, and Tracked Changes

The comments and tracked changes can be made visible in the PDF output by setting the `show.changes.and.comments` transformation parameter to `yes`.

Figure 5. Chemistry Annotations in Acrobat Reader

By default, they are shown as PDF text annotations (sticky notes). These are graphical markers in the document content and are also listed in the Comments section when opening the output file in Acrobat Reader.
Note:
Comments with the Mark as Done flag selected appear with a check mark in the Comments section and with a Completed label (✓ John Doe Completed).

To avoid rendering the elements as PDF annotations and show them as footnotes instead, you can use the show.changes.and.comments.as.pdf.sticky.notes transformation parameter set to no.

The comments and changes are included in the merged map file (on page 42) 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 22).

Related Information:
Transformation Parameters (on page 15)
Debugging the CSS (on page 42)

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 42).

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.

```html
<span class="oxy-range-start" id="sc_1" hr_id="1"/>

<span class="oxy-insert" href="#sc_1" 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.

Note:

Comments that are marked as done have a flag="done" attribute:

Attribute changes

The attribute changes are more complex. The range is empty, and is directly above the affected element (the one that has modified attributes). The element with the class oxy-attributes contains details about multiple attribute changes, each stored in an element with the class oxy-attribute-change.
Deletions

For a deletion, there are some elements that define the start and end of the deletion, and the highlighted text is wrapped in an element with the class oxy-delete-hl.

There is a structure that offers details about the deletion change, using the element with the class oxy-delete. This is linked to the above deletion range by the same ID value:

Colored Highlights

To show some text as highlighted with a background color:
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 41):

```css
.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 41) (the following example results in the deletions and insertions being hidden, but the comments remain visible):

```css
.oxy-attributes,
.oxy-comment,
.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 41):

```css
.oxy-attributes,
.oxy-comment,
.oxy-delete,
.oxy-insert {
    float: initial;
    display: none;
}
```

3. Remove the change range markers (the { and } symbols):
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 41)*:

```css
@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 41).

**Related Information:**
Images and Figures (on page 187)

### 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:

```css
@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).

```xml
<bookmeta>
  <author>John</author>
  <critdates>
    <created date="1/1/2015"/>
    <revised modified="3/4/2016"/>
    <revised modified="3/5/2016"/>
  </critdates>
  <bookchangehistory>
    <approved/>
  </bookchangehistory>
...
Use `oxy_xpath` every time you need to probe the value from an element other than the one matched by the CSS selector, and test the expression on the merged HTML file using the Oxygen XPath Builder view.

You can either use a page selector that imposes the draft watermark on the entire page surface (recommended):

```css
@page {
    background-image: url(oxy_xpath("if(/*[contains(@class, 'bookmap/approved')]) then '' else 'draft-watermark.png'"));
    background-position: center;
    background-repeat: no-repeat;
}
```

or use an element selector that restricts the watermark image only to the page area covered by that element:

```css
:root, body{
    /* same as properties above */
}
```

**Note:**
You can use another element selector to target a specific part of your publication (for example, marking only the tables as drafts).

**Related Information:**
- Metadata *(on page 91)*
- How to Debug XPath Expressions *(on page 46)*

**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:

```xml
<val>
    ...
    <prop action="flag" att="product" val="YourProd" bgcolor="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:

```xml
<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.

```xml
<val>
  <prop action="flag">
    att="product" val="MyProd"
    bgcolor="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>
  
</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 41](#)) and then add CSS rules. To create the CSS rules, you can use the development tools described in Debugging the CSS ([on page 42](#)).
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:

```html
<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>
</div>
```

And the same example from the PDF transformation (note the additional emphasized class values):

```html
<h1 class="- topic/title title topictitle1" id="ariaid-title2">Care and Preparation</h1>
<div class="- topic/body body">
  <p class="- topic/shortdesc shortdesc">When caring ...</p>
  <p class="- topic/p p">When caring for your flower garden you want ...</p>
</div>
```

It makes sense to reuse the same CSS rules you developed for one transformation type to the other. The main rule is to use the short class names instead of the long ones. For example, to style the short descriptions with italic font, use:

```css
.shortdesc {
  font-style: italic;
}
```

The rule of thumb is that if you have a CSS rule that successfully styles an element in WebHelp, it should apply without any modification in the PDF output.

**Titles**

Titles 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 41):

```css
*[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 41):

```css
*[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/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 41):

```css
*[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; }

*\[class =~ "map/topicref"\] [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; }

*\[class =~ "map/topicref"\] [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 41):

*\[class =~ "map/map"\] [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 41):

@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:
Then, you need to set it back on the content following the title:

```
*[^class ~="topic/topic"][^is-chapter] > *[^class ~="topic/title"] ~ *[^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 49)

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.

Lists

This is the default layout for lists (values are in px):

Markers are displayed in the padding area, so they are not included in the principal block box.

The lists are treated differently than ordinary block elements in the sense that their margins are not collapsed with the margins of the neighboring blocks or lists. This is also visible for nested lists. To summarize:

- Setting the padding-left or margin-left properties on lists will move the whole list.
- Setting the margin-left property on list items will move the whole list.
- Setting the padding-left property on list items will only move the list item content (not the marker).

Note:
If the padding-left property is set on lists and the margin-left property is set on list items, the result will move the whole list with a combination of both padding and margin values.

How to Align Lists with Page Margins

It is possible to reposition the lists to align them with the rest of the text from the body.

The default CSS rules for the lists are as follows:

```
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;
```
To align the lists, the following rules are sufficient in the customization CSS *(on page 41)*:

```css
*[^class~="topic/ol"],
*[^class~="topic/ul"] {
  padding-left: 0;
  list-style-position: inside;
}
```

**Note:**
By default, the `list-style-position` property is set to `outside`.

### How to Style the List Markers

For the media print, the default CSS rules for the list items use the `:marker` pseudo-class.

```css
@media print {
  *[^class~="topic/li"]::marker {
    text-align: center;
    font-weight: bold;
  }
  *[^class~="topic/li"] {
    margin-left: 0.25in;
  }
}
```

To eliminate the bold font weight for the items form ordered lists then add the following rules to your customization CSS *(on page 41)*:

```css
*[^class~="topic/ol"] > *[^class~="topic/li"]::marker {
  font-weight: normal;
}
```

For the unordered lists:

```css
*[^class~="topic/ul"] > *[^class~="topic/li"]::marker {
  font-weight: normal;
}
```

### How to Continue List Numbering

It is possible to continue the numbering of an ordered list even when the content is split on multiple `<ol>` elements.
You just need to define an `<outputclass>` attribute on the lists where numbering should continue:

```
<ol outputclass="continue">
  <li>Third Item</li>
</ol>
```

Then set the following content inside your CSS customization:

```
/*[class ~= "topic/ol"] {
  counter-reset: list-counter;
}

/*[class ~= "topic/ol"] [outputclass ~= "continue"] {
  counter-reset: none;
}

/*[class ~= "topic/ol"] > *[class ~= "topic/li"] {
  counter-increment: list-counter;
}

/*[class ~= "topic/ol"] > *[class ~= "topic/li"]::marker {
  content: counter(list-counter) ".00a0";
}
```

### How to Change the Numbering System of Ordered Lists

It is possible to change all lists to have a different numbering system and there are several methods that can be used to achieve this.

**Use the `list-style-type` CSS Property.**

The **Chemistry** engine supports the following types: `decimal`, `decimal-leading-zero`, `lower-roman`, `upper-roman`, `lower-latin`, `upper-latin`, `lower-alpha`, `upper-alpha`.

```
/*[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!

```css
*{class =~ "topic/ol"}[outputclass =~ "cyrillic"] > *{class =~ "topic/li"}:marker {
  width: 3em;
}
*{class =~ "topic/ol"}[outputclass =~ "cyrillic"] > *{class =~ "topic/li"}:nth-of-type(1):marker{ content:"а" }
*{class =~ "topic/ol"}[outputclass =~ "cyrillic"] > *{class =~ "topic/li"}:nth-of-type(2):marker{ content:"б" }
*{class =~ "topic/ol"}[outputclass =~ "cyrillic"] > *{class =~ "topic/li"}:nth-of-type(3):marker{ content:"в" }
*{class =~ "topic/ol"}[outputclass =~ "cyrillic"] > *{class =~ "topic/li"}:nth-of-type(4):marker{ content:"г" }
*{class =~ "topic/ol"}[outputclass =~ "cyrillic"] > *{class =~ "topic/li"}:nth-of-type(5):marker{ content:"д" }
*{class =~ "topic/ol"}[outputclass =~ "cyrillic"] > *{class =~ "topic/li"}:nth-of-type(6):marker{ content:"е" }
*{class =~ "topic/ol"}[outputclass =~ "cyrillic"] > *{class =~ "topic/li"}:nth-of-type(7):marker{ content:"ж" }
*{class =~ "topic/ol"}[outputclass =~ "cyrillic"] > *{class =~ "topic/li"}:nth-of-type(8):marker{ content:"з" }
*{class =~ "topic/ol"}[outputclass =~ "cyrillic"] > *{class =~ "topic/li"}:nth-of-type(9):marker{ content:"и" }
*{class =~ "topic/ol"}[outputclass =~ "cyrillic"] > *{class =~ "topic/li"}:nth-of-type(10):marker{ content:"к" }
*{class =~ "topic/ol"}[outputclass =~ "cyrillic"] > *{class =~ "topic/li"}:nth-of-type(11):marker{ content:"л" }
*{class =~ "topic/ol"}[outputclass =~ "cyrillic"] > *{class =~ "topic/li"}:nth-of-type(12):marker{ content:"м" }
*{class =~ "topic/ol"}[outputclass =~ "cyrillic"] > *{class =~ "topic/li"}:nth-of-type(13):marker{ content:"н" }
*{class =~ "topic/ol"}[outputclass =~ "cyrillic"] > *{class =~ "topic/li"}:nth-of-type(14):marker{ content:"о" }
*{class =~ "topic/ol"}[outputclass =~ "cyrillic"] > *{class =~ "topic/li"}:nth-of-type(15):marker{ content:"п" }
*{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" }
```
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, add the following in your customization CSS (on page 41):

```css
*[class =~ "topic/xref"] [href]:after, 
 *[class =~ "topic/link"] [href]:after {
   content: none !important;
}
```
A variant is to remove the "on page" label only and keep the page number:

```css
*[^= "topic/xref"][href]:after,
*[^= "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:

```css
*[^= "topic/entry"] *[^= "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 41), add this snippet:

```css
*[^= "topic/xref"][href]:after,
*[^= "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 42) like this:

```xml
<related-links class="- topic/related-links ">
  <linkpool class="- topic/linkpool ">
    <link class="- topic/link ">
      ...
      role="friend" scope="local" type="topic">
        <linktext class="- topic/linktext ">Salvia</linktext>
        <desc class="- topic/desc ">The salvia plant</desc>
      </link>
    </linkpool>
    ...
  </related-links>
```

If you need to hide these descriptions, add the following code in your customization CSS (on page 41):
How to Group Related Links by Type

By default, all links from DITA relationship tables or related link elements within topics are grouped under one "Related information" heading:

<table>
<thead>
<tr>
<th>Related information</th>
</tr>
</thead>
<tbody>
<tr>
<td>Target Topic</td>
</tr>
<tr>
<td>Target Concept</td>
</tr>
<tr>
<td>Target Task</td>
</tr>
</tbody>
</table>

It is possible to group the links by target type (topic type) by setting the `args.rellinks.group.mode=group-by-type` parameter. The output will look like this:

<table>
<thead>
<tr>
<th>Related concepts</th>
</tr>
</thead>
<tbody>
<tr>
<td>Target Concept</td>
</tr>
<tr>
<td>Related tasks</td>
</tr>
<tr>
<td>Target Task</td>
</tr>
<tr>
<td>Related information</td>
</tr>
<tr>
<td>Target Topic</td>
</tr>
</tbody>
</table>

Images and Figures

Images are an important part of a publication.

Note:

You can use raster image formats (such as PNG or JPEG), but it is best to use vector images (such as SVG or PDF). They scale very well and produce better results when printed. In addition, the text from these images is searchable and can be selected (if the glyphs have not been converted to shapes) in the PDF viewer.

Images - Built-in CSS

Image properties are defined in `PLUGIN_DIR/css/print/p-figures-images.css`.

```css
*[^topic/link] > *[^topic/desc] { display: none; }
```

```
*[^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 41), add the following snippet:

```
/* 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 193) 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 41):

```
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 193).
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 15) 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 41), 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 41), add:

```css
*[outputclass = "side-by-side"] > *[class = "topic/p"] {
    display:inline-block;
    width: 45%;
}

*[outputclass = "side-by-side"] > *[class = "topic/fig"] {
    display:inline-block;
    width: 45%;
}
```

The image should fill the entire width of the parent `<fig>` element:

```css
*[outputclass = "side-by-side"] > *[class = "topic/fig"] > *[class = "topic/image"] {
    width:100%;
}
```

By default, the bottom of the image is on the same line as the text baseline. If you want the text and the image to be aligned at the top, add these lines:

```css
*[outputclass = "side-by-side"] > *[class = "topic/p"] {
    vertical-align:top;
}

*[outputclass = "side-by-side"] > *[class = "topic/fig"] {
    vertical-align:top;
}
```

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.

```css
*[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:
If you want to change the size of the image, you have two options:

- Use the `image-resolution` CSS property:

```css
*[class =~ "topic/note"] {  
  image-resolution: 300dpi;
}
```

- Separate the image from the text and apply the width and height CSS properties only on the image, using the width and height properties. You could use multiple :before pseudo-elements for that, considering that the farthest content presented before the actual content of an element is matched by the :before with the highest number in the brackets:

```css
*[class =~ "topic/note"]:before(2) {  
  content: url('note.png');  
  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 41)*, add:

```css
*[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.

```css
/*
 *[class ~= "topic/image"] [placement='break'] [width] {
   margin-left: auto;
   margin-right: auto;
   border: 2pt solid red;
}
```

### How to Change/Reset the Figure Numbering

**Note:**
This topic is applicable for the DITA Map PDF - based on HTML5 & CSS DITA PDF - based on HTML5 & CSS transformation types.

There are cases when you need to change the aspect of the figure counter that is shown before the figure titles. By default, the figure titles are formatted like this:

```
Figure NN. Lore Ipsum Title
```

*NN* is the number of the figure that starts being counted from the beginning of the publication.

One use-case is to have the *NN* counter be incremented only within one chapter (for example, the first chapter contains "Figure 1" and "Figure 2", and the second chapter starts over with "Figure 1" instead of incrementing to "Figure 3").

You should reset the figure counter on each topic marked as chapter, then hide the label from the figure `<figcaption>` (this is an HTML element generated by the XSL transformation), and create another label using a `:before` selector on the `<figcaption>`.

```css
*[class ~= "topic/topic"] [is-chapter] {
  counter-reset: figcount;
}

.fig--title-label{
  display: none;
}

*[class ~= "topic/fig"] > .figcap:before{
  /* Add more styling or change the content if needed */
  content: "Figure " counter(figcount) ". ";
  counter-increment: figcount;
}
```
How to Fix Missing Images

If your images are not accessible, you may receive an error message in the transformation console like this:

```
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 188), 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.

   ```xml
   <imagemap>
     <image href="../images/Gear_pump_exploded.png" id="gear_pump_exploded" width="640" height="480">
       <alt>Gear Pump</alt>
     </image>
   </imagemap>
   ```

2. In the map element, add areas (each with a shape and a set of coordinates):
<imagemap>
  <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 one of the following methods:
Using the `show.image.map.area.numbers` and `show.image.map.area.shapes` transformation parameters.

- Adding a CSS snippet to your customization. The shapes have the `image-map-shape` class, the bullet around the image map number (`image-map-number`), and the text inside the bullet (`image-map-number-text`). To make them translucent yellow:

```css
.image-map-shape{
  fill: yellow;
  fill-opacity: 0.5;
  stroke-opacity: 0.5;
}
.image-map-number-text {
  visibility: visible;
}
.image-map-number {
  fill: yellow;
  fill-opacity: 0.4;
  stroke-opacity: 0.7;
}
```

**Tip:**

An SVG with links can be used as an alternative to the DITA `<imagemap>` element. Make sure that each link is a relative URI to an ID inside the publication content.

---

**How to Hide the Image Map Links List**

Below every image map, a list of links that point to the image map targets is displayed. This list can be hidden from the final output by using the following CSS selector:

```css
.imagemap--areas {
  display: none;
}
```

**How to Use SVG Syntax Diagrams**

The DITA `<syntaxdiagram>` element is supported by the PDF transformation. To use SVG syntax diagrams, follow this procedure:

1. Download the latest version of the `svg-syntaxdiagrams` plugin, unzip it, and copy all the folders into your `DITA-OT-DIR\plugins` folder (they all start with "com.").
2. Open a command prompt inside `DITA-OT-DIR\bin` and run the `dita install` command.
3. You can now add your custom `<syntaxdiagram>` element in your topic, as in the following example:

```xml
<syntaxdiagram id="syntaxdiagram_ok4_clk_xnb">
  <title>CopyFile</title>
</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:

![Syntax Diagram]

**Videos**

Videos can be referenced in a DITA topic by using the `<object>` element:

```html
<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:

```xml
<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:

```html
<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 41)*:

```css
.video {
  width: 480px;
}
```
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:

```css
.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:

```css
.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.

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%;
    page-break-before: avoid;
}
```

Note:
The landscape-page page layout is defined in the `PLUGIN_DIR/css/print/p-pages-and-headers.css` file.

If you want to rotate the entire topic that contains the big table, use:

```
*[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;
}
```
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:
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 ... >
...
<caption class="- topic/title title tablecap">
  <span class="table--title-label">Table
  <span class="table--title-label-number">1. </span></span>
  <span class="table--title">The title of the table</span>
</caption>
...

To hide it, set its display to none:

.table--title-label {
  display:none;
}

For the direct transformation, use:

*{class ~= "topic/table"} > *{class ~= "topic/title"}:before {
  content: none;
}

How to 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):
Applying Properties to Specific Elements

If you need to apply some properties to specific elements, you can use the DITA `@outputclass` attribute:

```xml
<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 outputclass="colored">
        <entry>Chrysanthemum</entry>
        <entry>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`:

```css
[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:

```css
/* Header background and foreground */
*[class = "topic/table"][outputclass =~ = "stripes"] > *[class =~ = "topic/thead"] > *[class ~ = "topic/row"] {
    background-color: blue;
    color: white;
}

/* A default background for the entire table body */
*[class =~ = "topic/table"][outputclass =~ = "stripes"] > *[class =~ = "topic/tbody"] {
    background-color: #eeeeee;
}

/* Color for the stripes */
*[class =~ = "topic/table"][outputclass =~ = "stripes"] > *[class =~ = "topic/tbody"] > *[class =~ = "topic/row"]:nth-child(odd) {
    background-color: cyan;
}

/* Border for the cells */
*[class =~ = "topic/table"][outputclass =~ = "stripes"] *[class =~ = "topic/entry"] {
    border: blue;
}
```

The above rules assume that tables that are to be painted with stripes are marked with an `@outputclass` attribute:

```xml
<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.
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.

```xml
<table frame="all" rowsep="1" colsep="1" id="table_d1p_flb_crb">
  <title>Car Features</title>
  <tgroup cols="4">
    <colspec colspan="1" colnum="1" colwidth="14*"/>
    <colspec colspan="2" colnum="2" colwidth="1*" align="center"/>
    <colspec colspan="2" colnum="3" colwidth="1*" align="center"/>
    <colspec colspan="2" colnum="4" colwidth="1*" align="center"/>
    <thead>
      <row>
        <entry morerows="1">Car Name</entry>
        <entry namestart="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>
    </tbody>
  </tgroup>
</table>
```
Table 2. Car Features

<table>
<thead>
<tr>
<th>Car Name</th>
<th>Features</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Sport</td>
</tr>
<tr>
<td>Tesla Model S</td>
<td>X</td>
</tr>
<tr>
<td>Nissan Leaf</td>
<td></td>
</tr>
<tr>
<td>Dacia Duster</td>
<td>X</td>
</tr>
</tbody>
</table>

The resulting output will be:

```
<table>
<thead>
<tr>
<th>Car Name</th>
<th>Features</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Sport</td>
</tr>
<tr>
<td>Tesla Model S</td>
<td>X</td>
</tr>
<tr>
<td>Nissan Leaf</td>
<td></td>
</tr>
<tr>
<td>Dacia Duster</td>
<td>X</td>
</tr>
</tbody>
</table>
```

The built-in CSS matches the cells with this attribute and applies the following properties:

```css
* [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:

```css
* [class~="topic/entry"] [rotate='1'] {
  vertical-align: middle;
}
```

The resulting output will be:
To make the text wrap (for instance, the "All terrain" could be split on two lines), you need to inhibit the whitespace preservation from the built-in CSS. In this case, all spaces will create a line break in the rotated layout. Thus, you can add this in your customization:

```
*{class="topic/entry"}[rotate='1'] {
  vertical-align:middle;
  white-space:normal;
}
```

The resulting output will be:

<table>
<thead>
<tr>
<th>Car Name</th>
<th>Features</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Sport</td>
</tr>
<tr>
<td>Tesla Model S</td>
<td>X</td>
</tr>
<tr>
<td>Nissan Leaf</td>
<td></td>
</tr>
<tr>
<td>Dacia Duster</td>
<td>X</td>
</tr>
</tbody>
</table>

**Note:**
The padding and borders set on the table cells are not rotated (only the content of the cell is rotated). You can use `padding-left` (for instance) to move the labels to the horizontal axis.

```
*{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.

\[
\begin{align*}
*\{\text{class} = \text{”task/choptionhd”}\}, \\
*\{\text{class} = \text{”task/choptionhd”}\}, \\
*\{\text{class} = \text{”task/chdeschd”}\}, \\
*\{\text{class} = \text{”task/choption”}\} \\
\quad \text{background-color: } \#\text{EEEEEE;} \\
\quad \text{text-align: } \text{left;} \\
\} \\
\*\{\text{class} = \text{”task/choicetable”}\} \\
\quad \text{border: } 2\text{pt solid } \#\text{EEEEEE;} \\
\} \\
*\{\text{class} = \text{”task/choicetable”}\} *\{\text{class} = \text{”task/chrow”}\}, \\
*\{\text{class} = \text{”task/choicetable”}\} *\{\text{class} = \text{”task/chhead”}\} \\
\quad \text{border-bottom: } 2\text{pt solid } \#\text{EEEEEE;} \\
\} \\
*\{\text{class} = \text{”task/choicetable”}\} *\{\text{class} = \text{”topic/stentry”}\} \\
\quad \text{border-bottom: none;} \\
\quad \text{border-right: none;} \\
\}
\]

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 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 \texttt{@outputclass} attribute on the \texttt{<codeblock>} elements to one of these values:

- \texttt{language-json}
- \texttt{language-yaml}
- \texttt{language-xml}
- \texttt{language-bourne}
- \texttt{language-c}
- \texttt{language-cmd}
- \texttt{language-csharp}
- \texttt{language-cpp}
- \texttt{language-csharp}
- \texttt{language-css}
- \texttt{language-dtd}
- \texttt{language-ini}
- \texttt{language-java}
- \texttt{language-javascript}
- \texttt{language-lua}
- \texttt{language-perl}
- \texttt{language-powershell}
- \texttt{language-php}
- \texttt{language-python}
- \texttt{language-ruby}
- \texttt{language-sql}
- \texttt{language-xquery}

For example, for a java snippet:

\begin{verbatim}
<codeblock outputclass="language-java">
   for (int i=0; i <100; i++) {
      // do something
   }
</codeblock>
\end{verbatim}

The resulting HTML fragment in the merged HTML5 document is:

\begin{verbatim}
<pre class="" topic/pre-d/codeblock pre codeblock language-java"
xml:space="preserve">
   <strong class="hl-keyword" style="color:#7f0055">for</strong>(int \texttt{i=0}; \texttt{i <100}; \texttt{i++}) {
      \texttt{// do something}
   }
</pre>
\end{verbatim}
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 41):

```css
.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:

```java
<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:

```html
<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">public void convert(String systemId, InputStream is) {</span>
</pre>
</pre>

And in the output, it is rendered as:

```
public void convert(String systemId, InputStream is) {
    return new FileInputStream();
}
```

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:

```java
<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:

```html
<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>
```
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 41):

```
*{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 221).

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 41):

```
*: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 to manually format the command line, add a line continuation character like this:

```
$gist = ls -l * | count -n \\
| some more
```

**Note:**

For Linux/macOSX, the continuation character is the backslash `\`. For Windows, this is the shift character `^`.

The command-line processor will now recognize that the first line is continuing on to the next one.

**Notes**

Notes contain an additional piece of information that calls attention to particular content. They may have various types (tip, caution, danger, restriction, important, warning).

For information on how to add and manage mixed content before the note icons and labels, see How to Control the Image Size in Complex Static Content *(on page 190)*.

**How to Change Note Icons**

The recommended icon format is SVG. The default size of the note icons is 24x24 px.

To change the default icons for notes that do not have a `@type` attribute, add the following rule to your customization CSS *(on page 41)*:

```
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 {
```
For a note with @type attribute set to *other* and @othertype attribute set to *Safety*, add the following CSS rule:

```css
div.note[type="other"][othertype=Safety] {
  background-image: url("../img/life-preserver.svg");
}
```

### How to Change Note Colors

To change the color of the default icons for notes that do not have a @type attribute, add the following rule to your customization CSS *(on page 41)*:

```css
everything-[class~="topic/note"]:not([class="hazard-hazardstatement"]) {
  background-color: #50bbff;
}
```

For a note with a @type attribute set to *restriction*, add the following CSS rule:

```css
everything-[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:

```css
everything-[class~="topic/note"][type = "other"][othertype = Safety] {
  background-color: #ffaa00;
}
```

### 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 41)*:

```css
*[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,
```
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 41):
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:

```xml
<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:

```html
<span class="- topic/tm tm" tmtype="tm">My Product Name</span><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:

```css
* [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:

```css
* [class ~= "topic/tmmark"] {
    vertical-align: super;
    font-size: smaller;
}
```

**Styling Through Custom Parameters**

You can activate parts of your CSS by using custom transformation parameters that start with the `args.css.param` prefix.
These parameters are recognized by the publishing pipeline and are forwarded as synthetic attributes on the root element of the merged map. The last part of the parameter name will become the attribute name, while the value of the parameter will become the attribute value. The namespace of these synthetic attributes is:

```
http://www.oxygenxml.com/extensions/publishing/dita/css/params
```

When using the DITA Map PDF - based on HTML5 & CSS or the DITA PDF - based on HTML5 & CSS transformations, the generated attribute will be in no namespace.

Notes:

- Make sure the name of your custom parameter does not conflict with an attribute name that may already exist on the root element.
- Use only Latin alphanumeric characters for parameter names.
- You can set multiple styling parameters at the same time.

How to Limit the Depth of the TOC Using a Parameter

In the following example, a custom parameter is used to switch from a full depth table of contents to a flat one that shows only the titles of the first-level topics (such as chapters, notices, or the preface).

The custom parameter is:

```
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:

```xml
<args css param="cover-page"="file:/path/to/cover-page.svg"/>
```

Then in the CSS, the attribute value is extracted and used as follows:

```css
@page front-page {
    background-image: url(oxy_xpath('/*/@*[local-name()="cover-page"]')[1]);
}
```
Controlling the Publication Content

Using a plain DITA map, the transformation will produce a publication with a front page, a table of contents, chapters with content, and an index at the end. This is appropriate for most cases, but there are use cases where some adjustments are necessary. For example, if you want to do one of the following:

- Remove the TOC or index.
- Add a glossary.
- Change the position of the TOC or the index relative to the sibling topics.
- Add a preface, frontmatter, or backmatter with copyright notices, abstracts, list of tables, list of figures, etc.

All of these can be achieved using a DITA `<bookmap>` element.

A bookmap has a more elaborate structure than a regular map. You should start by defining the title structure, with a main title and alternative title:

```xml
<!DOCTYPE bookmap PUBLIC "-//OASIS//DTD DITA BookMap//EN" "bookmap.dtd">
<bookmap id="taskbook">
  <booktitle>
    <mainbooktitle>Publication Title</mainbooktitle>
    <booktitlealt>A very short description of the publication</booktitlealt>
  </booktitle>
  <frontmatter>
    <topicref href="topics/abstract.dita"/>
    <booklists><toc/></booklists>
    <topicref href="topics/foreword.dita"/>
  </frontmatter>
</bookmap>
```

Then you may define a frontmatter. For this, you can link the topics that need to appear before the main content. You can also define the location where the table of contents will be placed. In the example below, it appears between the `abstract.dita` and `foreword.dita` topics:

```xml
<frontmatter>
  <topicref href="topics/abstract.dita"/>
  <booklists>
    <toc/>
  </booklists>
  <topicref href="topics/foreword.dita"/>
</frontmatter>
```

Note:
To remove the TOC from the publication, just omit the `<toc>` element from the `<booklists>` element.

Next, the topics are grouped into chapters:
At the end, you could define the structure of the backmatter. Just like for the frontmatter, you can include some topics and some generated content (such as the index). In the example below, the glossary is defined to come after the index, followed by a list of figures and list of tables. At the very end, there is a topic with some thank you notes.

```xml
<backmatter>
  <topicref href="topics/conclusion.dita"/>
  <booklists>
    <indexlist/>
    <glossarylist>
      <topicref href="topics/xp.dita" keys="xp" print="yes"/>
      <topicref href="topics/anti_lock_braking_system.dita" keys="abs" print="yes"/>
    </glossarylist>
    <figurelist/>
    <tablelist/>
  </booklists>
  <topicref href="topics/thanks.dita"/>
</backmatter>
```

As you can see, the bookmap offers much better control over the final content of the publication. It also offers more options in controlling the metadata that will go into the PDF (see the Metadata (on page 91) 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 15).

Related Information:
- How to Remove Entries from the TOC (on page 125)
- How to Hide the TOC (on page 125)
How to Make Chapters Look Like Individual Publications

Note: This topic is only applicable for the DITA Map PDF - based on HTML5 & CSS transformation scenario.

Sometimes you want to make each chapter independent (i.e. it can be read separately, as a separate part of your publication). For this, you need the page counter, figure, and table counters to restart at each chapter. You can control this by using the `args.css.param.numbering (on page 107)` command-line parameter.

In addition to numbering, you can force the creation of a chapter TOC (on page 125).
6. 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 42) 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 42) 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 12) and merged2html5 (on page 13).
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.
2. Link the folder associated with the publishing template to your current project in the Project view.
3. Using the Project view, create an `xslt` folder inside the project root folder.
4. In this folder, create an XSL file (for example, named `merged2html5Extension.xsl`) with the following content:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    xmlns:xs="http://www.w3.org/2001/XMLSchema"
    exclude-result-prefixes="xs"
    version="2.0">

    <xsl:template match="*[contains(@class, ' 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:element>
    </xsl:template>
</xsl:stylesheet>
```
5. Open the template descriptor file (on page 29) associated with your publishing template (the .opt file) and set the XSLT stylesheet created in the previous step with the com.oxygenxml.pdf.css.xsl.merged2html5 XSLT extension point:

```xml
<publishing-template>
  ...
  <pdf>
    ...
    <xslt>
      <extension
        id="com.oxygenxml.pdf.css.xsl.merged2html5"
        file="xslt/merged2html5Extension.xsl"/>
    </xslt>
  </pdf>
</publishing-template>
```

6. Create a css folder in the publishing template directory. In this directory, save a custom CSS file with rules that style the `<codeblock>` structure. For example:

```css
.zebra {
  padding: 0;
}

.zebra > *:nth-of-type(odd) {
  background-color: lightgray;
}
```

7. Open the template descriptor file (on page 29) associated with your publishing template (the .opt file) and reference your custom CSS file in the resources element:

```xml
<publishing-template>
  ...
  <pdf>
    ...
    <resources>
      ...
    </resources>
</publishing-template>
```
8. Edit the DITA Map PDF - based on HTML5 & CSS transformation scenario.
9. In the Templates tab, click the Choose Custom Publishing Template link and select your template.
10. Click OK to save the changes and run the transformation scenario.

How to Remove the Related Links Section

Suppose that you want the related links sections to be removed from the PDF output.

To add this functionality using an Oxygen Publishing Template, follow these steps:

1. If you have not already created a Publishing Template, you need to create one. For details, see How to Create a Publishing Template.
2. Link the folder associated with the publishing template to your current project in the Project view.
3. Using the Project view, create an xslt folder inside the project root folder.
4. In this folder, create an XSL file (for example, named merged2mergedExtension.xsl) with the following content:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    xmlns:xs="http://www.w3.org/2001/XMLSchema"
    exclude-result-prefixes="xs"
    version="2.0">
    <xsl:template match="*[contains(@class, ' topic/related-links ')]">
        <!-- Remove. -->
    </xsl:template>
</xsl:stylesheet>
```

5. Open the template descriptor file (on page 29) associated with your publishing template (the .opt file) and set the XSLT stylesheet created in the previous step with the com.oxygenxml.pdf.css.xsl.merged2merged XSLT extension point:

```xml
<publishing-template>
    ...
    <pdf>
        ...
        <xslt>
            <extension
                id="com.oxygenxml.pdf.css.xsl.merged2merged"
                file="xslt/merged2mergedExtension.xsl"/>
        </xslt>
    </pdf>
    ...
</publishing-template>
```

6. Edit the DITA Map PDF - based on HTML5 & CSS transformation scenario.
7. In the **Templates** tab, click the **Choose Custom Publishing Template** link and select your template.
8. Click **OK** to save the changes and run the transformation scenario.

### How to Wrap Words in Markup

Suppose you want compound words that contain hyphens (or any other criteria) to be wrapped with inline elements (such as the HTML `<code>` element).

To add this functionality using an *Oxygen Publishing Template*, follow these steps:

1. If you have not already created a Publishing Template, you need to create one. For details, see *How to Create a Publishing Template*.
2. Link the folder associated with the publishing template to your current project in the **Project** view.
3. Using the **Project** view, create an xslt folder inside the project root folder.
4. In this folder, create an XSL file (for example, named `merged2htmlExtension.xsl`) with the following content:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    xmlns:xs="http://www.w3.org/2001/XMLSchema"
    exclude-result-prefixes="xs"
    version="2.0">

    <xsl:template match="text()">

        <xsl:variable name="txt">
            <xsl:next-match/>
        </xsl:variable>

        <xsl:analyze-string regex="\w|-\w\-\s+" select="$txt">
            <!-- 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:analyze-string>
    </xsl:template>
</xsl:stylesheet>
```
5. Open the template descriptor file (on page 29) associated with your publishing template (the .opt file) and set the XSLT stylesheet created in the previous step with the com.oxygenxml.pdf.css.xsl.merged2merged XSLT extension point:

```xml
<publishing-template>
  ...
  <pdf>
    ...
    <xslt>
      <extension
        id="com.oxygenxml.pdf.css.xsl.merged2merged"
        file="xslt/merged2mergedExtension.xsl"/>
    </xslt>
  </pdf>
</publishing-template>
```

6. Edit the DITA Map PDF - based on HTML5 & CSS transformation scenario.

7. In the Templates tab, click the Choose Custom Publishing Template link and select your template.

8. Click OK to save the changes and run the transformation scenario.

How to Convert Definition Lists into Tables

Suppose you want your definitions lists (`<dl>`) to be displayed as tables in your PDF output.

To add this functionality using an Oxygen Publishing Template, follow these steps:

1. If you have not already created a Publishing Template, you need to create one. For details, see How to Create a Publishing Template.

2. Link the folder associated with the publishing template to your current project in the Project view.

3. Using the Project view, create an `xslt` folder inside the project root folder.

4. In this folder, create an XSL file (for example, named `merged2html5Extension.xsl`) with the following content:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
 xmlns:xs="http://www.w3.org/2001/XMLSchema"
 exclude-result-prefixes="xs"
 version="2.0">
```
<xsl:template match="*[contains(@class, ' topic/dl ')]">
  <xsl:apply-templates select="*[contains(@class, ' topic/dlentry ')]">
    <!-- Wrap in a table row -->
    <tr>
      <td>
        <!-- Wrap in a cell -->
        <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 select="../*[contains(@class, ' ditaot-d/ditaval-endprop ')]" mode="out-of-line"/>
      </td>
    </tr>
  </xsl:template>
  <xsl:apply-templates select="* *[contains(@class, ' topic/dd ')] | *[contains(@class, ' topic/dt ')]">
    <!-- Wrap in a cell -->
    <td>
      <!-- Wrap in a cell -->
      <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 select="../*[contains(@class, ' ditaot-d/ditaval-endprop ')]" mode="out-of-line"/>
    </td>
  </xsl:template>
</xsl:template>
5. Open the template descriptor file (on page 29) 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>
  </pdf>
</publishing-template>
```

6. Edit the DITA Map PDF - based on HTML5 & CSS transformation scenario.

7. In the Templates tab, click the Choose Custom Publishing Template link and select your template.

8. Click OK to save the changes and run the transformation scenario.

### How to 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.

2. Link the folder associated with the publishing template to your current project in the Project view.

3. Using the Project view, create an xslt folder inside the project root folder.

4. In the newly created folder, create an XSL file (for example, named merged2mergedExtension.xsl) with the following content:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    xmlns:xs="http://www.w3.org/2001/XMLSchema"
    xmlns:opentopic-func="http://www.idiominc.com/opentopic/exsl/function"
    exclude-result-prefixes="xs opentopic-func"
    version="2.0">

    <!--
    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>
</xsl:template>

<!--[if !print]><!-->

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:apply-templates select="node()"/>
</li>
</xsl:for-each>
</ol>
</xsl:template>

<!--[if !print]><!-->

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]"/>

<!--[if !print]><!-->

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>
Oxygen DITA-OT CSS-based PDF Publishing plugin 25.1 | 6 - XSLT Extensions for PDF Transformations | 230

<xsl:stylesheet>
<xsl:when test="$fn"/>
<what>There is a reference in the table, recalculate index. --></what>
<xsl:variable name="fn-number" select="
index-of($top-level-table//*[contains(@class, 'topic/fn')], $fn)"/>
<xsl:copy>
<xsl:apply-templates select="@*"/>
<xsl:apply-templates select="node() except *[contains(@class, 'hi-d/sup')]"/>
<sup>
<xsl:apply-templates select="child::*[contains(@class, 'hi-d/sup')]/@*"/>
<xsl:value-of select="$fn-number"/>
</sup>
</xsl:copy>
</xsl:when>
<xsl:otherwise/>
<what>There is no reference in the table, keep original index. --></what>
<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)]">
<what>Determine the footnote index in the document order. --></what>
<xsl:param name="top-level-table" tunnel="yes"/>
<xsl:variable name="fn-number" select="
index-of($top-level-table/*[contains(@class, 'topic/fn')], .)"/>
<xref type="fn" class="- topic/xref " href="#{generate-id(.)}" outputclass="table-fn-call">
<what>Generate an extra <sup>, identical to what DITA-OT generates for other xrefs. --></what>
<sup class="+ topic/ph hi-d/sup ">
<xsl:value-of select="$fn-number"/>
</sup>
</xref>
</xsl:template>
</xsl:stylesheet>
5. Open the template descriptor file (on page 29) associated with your publishing template (the .opt file) and set the XSLT stylesheet created in the previous step with the com.oxygenxml.pdf.css.xsl.merged2merged XSLT extension point:

```xml
<publishing-template>
...
<pdf>
...
<xslt>
  <extension id="com.oxygenxml.pdf.css.xsl.merged2merged" file="xslt/merged2mergedExtension.xsl"/>
</xslt>
</publishing-template>
```

6. Create a css folder in the publishing template directory. In this directory, save a custom CSS file with rules that style the glossary structure. For example:

```css
/* Customize footnote calls, inside the table. */
*[outputclass ~='table-fn-call'] {  
  line-height: none;
}

/* 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) ");";
}

/* Customize xrefs pointing to footnotes with IDs. */
*[class ~='topic/table'] *[class ~='topic/xref'][type = 'fn'] {  
  color: unset;
  text-decoration: none;
}
*[class ~='topic/table'] *[class ~='topic/xref'][type = 'fn']::after {  
  content: none;
}
Oxygen DITA-OT CSS-based PDF Publishing plugin 25.1 | 6 - XSLT Extensions for PDF Transformations | 232

7. Open the **template descriptor file (on page 29)** associated with your **publishing template** (the .opt file) and reference your custom CSS file in the **resources** element:

```xml
<publishing-template>
  ...
  <pdf>
    ...
    <resources>
      <css file="css/custom.css"/>
    </resources>
  </pdf>
</publishing-template>
```

8. Edit the **DITA Map PDF - based on HTML5 & CSS** transformation scenario.

9. In the **Templates** tab, click the **Choose Custom Publishing Template** link and select your template.

10. Click **OK** to save the changes and run the transformation scenario.

---

### How to Wrap Scientific Numbers in Tables Cells

In your PDF output, you may need to wrap scientific numbers on two lines when they are included in table cells.

To add this functionality, use an **Oxygen Publishing Template** and follow these steps:

1. If you have not already created a Publishing Template, you need to create one. For details, see **How to Create a Publishing Template**.

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**)

   ```xml
   <?xml version="1.0" encoding="UTF-8"?>
   <xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
                   xmlns:xs="http://www.w3.org/2001/XMLSchema"
                   exclude-result-prefixes="xs"
                   version="2.0">
   <!-- Matches text from table cells. -->
   ```
<xsl:template match="*[contains(@class, ' topic/entry ')]/text()">
  <xsl:analyze-string select="." regex="[0-9]\.[0-9]{2}e-\-[0-9]{2}"/>
  <!-- The cell contains a scientific number like 1.23e-08. -->
  <xsl:variable name="text" select="concat(substring-before(., 'e'), 'e', substring-after(., 'e'))"/>
  <xsl:value-of select="$text"/>
</xsl:analyze-string>
</xsl:template>

5. Open the template descriptor file (on page 29) 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>
  </pdf>
</publishing-template>
```

6. Edit the DITA Map PDF - based on HTML5 & CSS transformation scenario.
7. In the Templates tab, click the Choose Custom Publishing Template link and select your template.
8. Click OK to save the changes and run the transformation scenario.

**How to 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.

2. Link the folder associated with the publishing template to your current project in the Project view.

3. Using the Project view, create an xslt folder inside the project root folder.

4. In the newly created folder, create an XSL file (for example, named merged2html5Extension.xsl) with the following content:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    xmlns:xs="http://www.w3.org/2001/XMLSchema"
    exclude-result-prefixes="xs"
    version="2.0">

    <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 29) associated with your publishing template (the .opt file) and set the XSLT stylesheet created in the previous step with the com.oxygenxml.pdf.css.xsl.merged2html5 XSLT extension point:

```xml
<publishing-template>
    ...
    <pdf>
        ...
    <xslt>
```

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:

```xml
<extension
  id="com.oxygenxml.pdf.css.xsl.merged2html5"/>
  file="xslt/merged2html5Extension.xsl"
</xslt>
```

7. Open the template descriptor file (on page 29) associated with your publishing template (the `.opt` file) and reference your custom CSS file in the `resources` element:

```xml
<publishing-template>
  ...
  <pdf>
    ...
    <resources>
      <css file="css/custom.css"/>
      </resources>
  </pdf>
</publishing-template>
```

8. Edit the DITA Map PDF - based on HTML5 & CSS transformation scenario.

9. In the Templates tab, click the Choose Custom Publishing Template link and select your template.

10. Click OK to save the changes and run the transformation scenario.

## How to Change the Critical Dates Format

By default, the dates are entered in a `YYYY-MM-DD` format (where `YYYY` is the year, `MM` is the number of the month, and `DD` is the number of the day. You can change the format (for example, to something like `January 1, 2020`) using an XSLT extension.

To add this functionality, use an Oxygen Publishing Template and follow these steps:

1. If you have not already created a Publishing Template, you need to create one. For details, see How to Create a Publishing Template.

2. Link the folder associated with the publishing template to your current project in the Project view.

3. Using the Project view, create an `xslt` folder inside the project root folder.

4. In the newly created folder, create an XSL file (for example, named `merged2mergedExtension.xsl`) with the following content:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  exclude-result-prefixes="xs"/>
```
5. Open the template descriptor file (on page 29) 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>
  </pdf>
</publishing-template>
```

6. Edit the DITA Map PDF - based on HTML5 & CSS transformation scenario.
7. In the Templates tab, click the Choose Custom Publishing Template link and select your template.
8. Click OK to save the changes and run the transformation scenario.

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.
2. Link the folder associated with the publishing template to your current project in the Project view.
3. Using the Project view, create an `xslt` folder inside the project root folder.
4. In the newly created folder, create an XSL file (for example, named `merged2html5Extension.xsl`) with the following content:

```xml
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
 xmlns:xs="http://www.w3.org/2001/XMLSchema"
 exclude-result-prefixes="xs"
 version="2.0">

  <xsl:template match="*[contains(@class, ' topic/term ')]" name="topic.term">
    <!-- Save the current @href value -->
    <xsl:variable name="current-href" select="@href"/>
    <!-- Get the closest parent topic -->
    <xsl:variable name="closest-parent" select="ancestor::*[contains(@class, ' topic/topic ')][1]"/>
    <!-- Get the first <term> having the same href -->
    <xsl:variable name="first-term-with-same-href" select="($closest-parent//*[@href=$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="$result" mode="remove-extra-links">
      <xsl:with-param name="is-first-term-with-same-href" tunnel="yes">
        <xsl:choose>
          <!-- Process the first term as a link -->
          <xsl:when test="$is-first-term-with-same-href">
            <xsl:next-match/>
          </xsl:when>
        </xsl:choose>
      </xsl:with-param>
    </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:choose>
  </xsl:template>

</xsl:stylesheet>
```
<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 29) associated with your publishing template (the .opt file) and set the XSLT stylesheets 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>
   </pdf>
   ...
</publishing-template>

6. Edit the DITA Map PDF - based on HTML5 & CSS transformation scenario.
7. In the Templates tab, click the Choose Custom Publishing Template link and select your template.
8. Click OK to save the changes and run the transformation scenario.

**How to 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:](image.png)

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.
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="2.0">

<!-- 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>
    <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', $nestlevel)"/>
      </xsl:call-template>
    </xsl:otherwise>
  </xsl:choose>
  <xsl:call-template name="gen-toc-id"/>
  <xsl:call-template name="setidaname"/>
  <xsl:choose>
    <xsl:when test="contains(@class, 'glossgroup/glossgroup')">
      <!-- Custom processing for glossgroup. -->
    </xsl:when>
    <xsl:otherwise>
      <xsl:call-template name="gen-toc-id"/>
      <xsl:call-template name="setidaname"/>
      <xsl:choose>
        <xsl:when test="contains(@class, 'glossgroup/glossgroup')">
          <!-- Custom processing for glossgroup. -->
        </xsl:when>
      </xsl:choose>
    </xsl:otherwise>
  </xsl:choose>
</xsl:template>
</xsl:stylesheet>
<xsl:template match="* [contains(@class, 'glossgroup/glossgroup')]">
  <xsl:variable name="glossentry" as="node()">
    <xsl:next-match/>
  </xsl:variable>
  <tr>
    <xsl:copy-of select="$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')]"/>
  </tr>
</xsl:template>

<!-- Create a row for each glossentry. -->
<xsl:template match="* [contains(@class, 'glossentry/glossentry')]">
  <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')]"/>
  </tr>
</xsl:template>

<!-- Default processing. -->
<xsl:apply-templates/>
</xsl:otherwise>
</xsl:choose>
</xsl:template>

<!-- Create a row for each glossentry. -->
<xsl:template match="* [contains(@class, 'glossentry/glossentry')]">
  <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')]"/>
  </tr>
</xsl:template>

<!-- Default processing. -->
<xsl:apply-templates/>
</xsl:otherwise>
</xsl:choose>
</xsl:template>

<!-- Create a row for each glossentry. -->
<xsl:template match="* [contains(@class, 'glossentry/glossentry')]">
  <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')]"/>
  </tr>
</xsl:template>

<!-- Default processing. -->
<xsl:apply-templates/>
</xsl:otherwise>
</xsl:choose>
</xsl:template>

<!-- Create a row for each glossentry. -->
<xsl:template match="* [contains(@class, 'glossentry/glossentry')]">
  <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')]"/>
  </tr>
</xsl:template>

<!-- Default processing. -->
<xsl:apply-templates/>
</xsl:otherwise>
</xsl:choose>
</xsl:template>

<!-- Create a row for each glossentry. -->
<xsl:template match="* [contains(@class, 'glossentry/glossentry')]">
  <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')]"/>
  </tr>
</xsl:template>

<!-- Default processing. -->
<xsl:apply-templates/>
</xsl:otherwise>
</xsl:choose>
</xsl:template>

<!-- Create a row for each glossentry. -->
<xsl:template match="* [contains(@class, 'glossentry/glossentry')]">
  <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')]"/>
  </tr>
</xsl:template>

<!-- Default processing. -->
<xsl:apply-templates/>
</xsl:otherwise>
</xsl:choose>
</xsl:template>

<!-- Create a row for each glossentry. -->
<xsl:template match="* [contains(@class, 'glossentry/glossentry')]">
  <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')]"/>
  </tr>
</xsl:template>

<!-- Default processing. -->
<xsl:apply-templates/>
</xsl:otherwise>
</xsl:choose>
</xsl:template>

<!-- Create a row for each glossentry. -->
<xsl:template match="* [contains(@class, 'glossentry/glossentry')]">
  <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')]"/>
  </tr>
</xsl:template>

<!-- Default processing. -->
<xsl:apply-templates/>
</xsl:otherwise>
</xsl:choose>
</xsl:template>

<!-- Create a row for each glossentry. -->
<xsl:template match="* [contains(@class, 'glossentry/glossentry')]">
  <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')]"/>
  </tr>
</xsl:template>

<!-- Default processing. -->
<xsl:apply-templates/>
</xsl:otherwise>
</xsl:choose>
</xsl:template>

<!-- Create a row for each glossentry. -->
<xsl:template match="* [contains(@class, 'glossentry/glossentry')]">
  <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')]"/>
  </tr>
</xsl:template>

<!-- Default processing. -->
<xsl:apply-templates/>
</xsl:otherwise>
</xsl:choose>
</xsl:template>

<!-- Create a row for each glossentry. -->
<xsl:template match="* [contains(@class, 'glossentry/glossentry')]">
  <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/glossdef')]">
</xsl:copy-of>

<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')]">
</xsl:copy-of>

</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>

<!-- 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>
5. Open the template descriptor file (on page 29) associated with your publishing template (the .opt file) and set the XSLT stylesheet created in the previous step with the com.oxygenxml.pdf.css.xsl.merged2html5 XSLT extension point:

```xml
<publishing-template>
...  
<pdf>
...  
<xslt>
<extension

   id="com.oxygenxml.pdf.css.xsl.merged2html5"

   file="xslt/merged2html5Extension.xsl"/>
</xslt>
</pdf>
</publishing-template>
```

6. Create a css folder in the publishing template directory. In this directory, save a custom CSS file with rules that style the glossary structure. For example:

```css
/*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 */
thin[type-of-type(3)],
/*class ~="glossentry/glossSurfaceForm"*/ {
  display: none;
}
```
7. Open the template descriptor file (on page 29) associated with your publishing template (the .opt file) and reference your custom CSS file in the resources element:

    <publishing-template>
        ...
        <pdf>
            ...
            <resources>
                <css file="css/custom.css"/>
            </resources>
        </pdf>
    </publishing-template>

8. Edit the DITA Map PDF - based on HTML5 & CSS transformation scenario.

9. In the Templates tab, click the Choose Custom Publishing Template link and select your template.

10. Click OK to save the changes and run the transformation scenario.

### 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.

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
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="2.0"
    xmlns:opentopic-index="http://www.idiominc.com/opentopic/index"
    xmlns:opentopic="http://www.idiominc.com/opentopic"
    xmlns:xs="http://www.w3.org/2001/XMLSchema" exclude-result-prefixes="#all">
    <xsl:template match="*[contains(@class, ' topic/topic ')]">
        ...
    </xsl:template>
</xsl:stylesheet>
```
<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
      "}]]>
</xsl:template>
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 ')">  
    <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>
5. Open the template descriptor file (on page 29) associated with your publishing template (the .opt file) and set the XSLT stylesheet created in the previous step with the com.oxygenxml.pdf.css.xsl.merged2merged XSLT extension point:

```xml
<publishing-template>
  ...
  <pdf>
    ...
    <xslt>
      <extension id="com.oxygenxml.pdf.css.xsl.mergedmerged"/>
      file="xslt/merged2mergedExtension.xsl"
    </xslt>
    <parameters>
      <parameter name="args.chapter.layout" value="MINITOC"/>
    </parameters>
  </pdf>
</publishing-template>
```

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.
2. Link the folder associated with the publishing template to your current project in the Project view.
3. Using the Project view, create an xslt folder inside the project root folder.
4. In the newly created folder, create an XSL file (for example, named merged2html5Extension.xsl) with the following content:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  exclude-result-prefixes="xs"
  version="2.0">
```
5. Open the template descriptor file (on page 29) associated with your publishing template (the .opt file) and set the XSLT stylesheet created in the previous step with the com.oxygenxml.pdf.css.xsl.merged2html5 XSLT extension point:

```xml
<publishing-template>
  ...
  <pdf>
    ...
    <xslt>
      <extension
          id="com.oxygenxml.pdf.css.xsl.merged2html5"/>
      file="xslt/merged2html5Extension.xsl"
    </xslt>
  </pdf>
</publishing-template>
```

6. Create a css folder in the publishing template directory. In this directory, save a custom CSS file with rules that style the glossary structure. For example:

```css
@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 29) associated with your publishing template (the .opt file) and reference your custom CSS file in the resources element:
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:

   ```xml
   <plugin id="com.oxygenxml.pdf.custom.codeblocks">
       <feature extension="com.oxygenxml.pdf.css.xsl.merged2html5"
                 file="custom_codeblocks.xsl"/>
   </plugin>
   ```

3. Create your customization stylesheet (for example, `custom_codeblocks.xsl`) with the following content:

   ```xml
   <?xml version="1.0" encoding="UTF-8"?>
   <xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
                   xmlns:xs="http://www.w3.org/2001/XMLSchema"
                   exclude-result-prefixes="xs"
                   version="2.0">
     
       <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 Run DITA-OT Integrator transformation scenario found in the DITA Map section in the Configure Transformation Scenario(s) dialog box.

5. Create a custom CSS file with rules that style the `codeblock` structure. For example:

   ```css
   div.zebra {
     font-family: courier, fixed, monospace;
     white-space: pre-wrap;
   }
   
   div.zebra > *:nth-of-type(odd){
     background-color: silver;
   }
   ```

6. Edit a DITA Map PDF - based on HTML5 & CSS transformation scenario and reference your custom CSS file (using the `args.css` parameter).

7. Run the transformation scenario.
How to Remove the Related Links Section

Suppose you want the related links sections to be removed from the PDF output.

To add this functionality using a DITA-OT plugin, follow these steps:

1. In the DITA-OT-DIR\plugins\ folder, create a folder for this plugin (for example, com.oxygenxml.pdf.custom.codeblocks).

2. Create a plugin.xml file (in the folder you created in step 1) that specifies the extension point and your customization stylesheet. For example:

   ```xml
   <plugin id="com.oxygenxml.pdf.custom.related.links">
     <feature extension="com.oxygenxml.pdf.css.xsl.merged2merged"
              file="custom_related_links.xsl"/>
   </plugin>
   ```

3. Create your customization stylesheet (for example, custom_related_links.xsl) with the following content:

   ```xml
   <?xml version="1.0" encoding="UTF-8"?>
   <xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
                   xmlns:xs="http://www.w3.org/2001/XMLSchema"
                   exclude-result-prefixes="xs"
                   version="2.0">

   <xsl:template match="*[contains(@class, ' topic/related-links ')]">
     <!-- Remove. -->
   </xsl:template>
   </xsl:stylesheet>
   ```

4. Use the Run DITA-OT Integrator transformation scenario found in the DITA Map section in the Configure Transformation Scenario(s) dialog box.

5. Run the DITA Map PDF - based on HTML5 & CSS transformation scenario.

How to Use Custom Parameters in XSLT Stylesheets

Suppose you want to add an attribute with a custom value inside a <div> element.

To add this functionality using a DITA-OT plugin, follow these steps:

1. In the DITA-OT-DIR\plugins\ folder, create a folder for this plugin (for example, com.oxygenxml.pdf.css.param).

2. Create a plugin.xml file (in the folder you created in step 1) that specifies the extension points, your parameter file, and your customization stylesheet. For example:

   ```xml
   <plugin id="com.oxygenxml.pdf.css.param">
     <feature extension="com.oxygenxml.pdf.css.xsl.merged2html5.parameters" file="params.xml"/>
   </plugin>
   ```
<feature extension="com.oxygenxml.pdf.css.xsl.merged2html5" file="custom.xsl"/>

Note:
The `com.oxygenxml.pdf.css.xsl.merged2html5` extension point can also be called from a Publishing Template.

3. Create a `params.xml` file that specifies the name of the custom attribute with the following content:

```xml
<dummy xmlns:if="ant:if">
  <param name="custom-param" expression="${custom.param}" if:set="custom.param"/>
</dummy>
```

4. Create your customization stylesheet (for example, `custom.xsl`) with the following content:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    xmlns:xs="http://www.w3.org/2001/XMLSchema"
    exclude-result-prefixes="xs"
    version="2.0">
  <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 Run DITA-OT Integrator transformation scenario 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 221)

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 248)
8. Localization

DITA-OT supports more than 40 languages. The full list of supported languages (and their codes) is available here: https://www.dita-ot.org/3.6/topics/globalization-languages.html.

There are two ways to switch the labels to a specific language:

- Set the `@xml:lang` attribute on the DITA maps and/or topics root element with one of the supported values (e.g. `de`, `fr-FR`, `ru`, `zh-CN`).
- Set the `default.language` parameter in the transformation dialog box to the desired language code.

You can create language-dependent CSS rules in your customization CSS (on page 41) by adding rules using the `:lang` pseudo-class (see https://developer.mozilla.org/en-US/docs/Web/CSS/:lang).

**Tip:**

It is recommended that you do this customization on a DITA-OT distribution deployed outside of the Oxygen installation. Otherwise, you will lose the customization when upgrading Oxygen. You can contact the Oxygen support team to ask for the Oxygen Publishing Engine package.

How to Customize CSS Strings

Some of the labels come from CSS files located in the `DITA-OT-DIR/plugins/com.oxygenxml.pdf.css/css/print/i18n` directory. These strings can be overridden directly from a custom CSS stylesheet. Simply identify (by debugging the CSS) and copy the rules that apply on your content and change their values. For example:

```
/* 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 42)

How to Modify Existing Strings

If the label you want to modify is not available from the CSS, you need to modify the XML strings. The default XML strings are available at the following two locations:

- DITA-OT-DIR/plugins/org.dita.base/xsl/common
- DITA-OT-DIR/plugins/com.oxygenxml.pdf.css/resources/localization

To modify the generated text, you need to create a DITA-OT extension plugin that uses the dita.xsl.strings extension point. The following example uses English, but you can adapt it for any language:

1. In the DITA-OT-DIR/plugins/ folder, create a folder for this plugin (for example, com.oxygenxml.pdf.css.localization).
2. Create a plugin.xml file (in the folder you created in step 1) that specifies the extension points, your parameter file, and your customization stylesheet. For example:

```xml
<plugin id="com.oxygenxml.pdf.css.localization">
  <require plugin="com.oxygenxml.pdf.css"/>
  <feature extension="dita.xsl.strings" file="pdf-extension-strings.xml"/>
</plugin>
```

3. Create a pdf-extension-strings.xml file with the following content:

```xml
<langlist>
  <lang xml:lang="en" filename="strings-en-us.xml"/>
  <lang xml:lang="en-us" filename="strings-en-us.xml"/>
</langlist>
```

4. Copy the strings you want to change from the default files to the strings-en-us.xml file, then replace their values:

```xml
<strings xml:lang="en-US">
  <str name="Figure">Fig</str>
</strings>
```
Warning:
Make sure the string @name attribute remains the same, it is used by the process as a key to retrieve the strings text.

5. Use the Run DITA-OT Integrator transformation scenario found in the DITA Map section in the Configure Transformation Scenario(s) dialog box.
6. Run the DITA Map PDF - based on HTML5 & CSS transformation scenario.

How to Add New Strings

Some strings are not translated in all languages. In this case, they will appear in English. To add a new language for a given string, you need to create a DITA-OT extension plugin that uses the dita.xsl.strings extension point. The following example uses Polish, but you can adapt it for any language:

1. In the DITA-OT-DIR\plugins\ folder, create a folder for this plugin (for example, com.oxygenxml.pdf.css.localization).
2. Create a plugin.xml file (in the folder you created in step 1) that specifies the extension points, your parameter file, and your customization stylesheet. For example:

```xml
<plugin id="com.oxygenxml.pdf.css.localization">
  <require plugin="com.oxygenxml.pdf.css"/>
  <feature extension="dita.xsl.strings" file="pdf-extension-strings.xml"/>
</plugin>
```

3. Create a pdf-extension-strings.xml file with the following content:

```xml
<langlist>
  <lang xml:lang="pl" filename="strings-pl-pl.xml"/>
  <lang xml:lang="pl-pl" filename="strings-pl-pl.xml"/>
</langlist>
```

4. Copy the strings you want to change from the default files to the strings-pl-pl.xml file, then replace their values:

```xml
<strings xml:lang="pl-PL">
  <str name="Continued">ciąg dalszy</str>
</strings>
```

Warning:
Make sure the string @name attribute remains the same, it is used by the process as a key to retrieve the strings text.
5. Use the Run DITA-OT Integrator transformation scenario 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.
9. 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 15) section can be used for this purpose.

**How to Protect PDF Files by Setting Security Permissions**

For example, to permit only the users that have a password to access the document and also to restrict their printing and copying capability, you can use the following parameter combination:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>pdf.security.owner.password</code></td>
<td><code>&lt;OWNER PASSWORD&gt;</code></td>
<td>People using this password will be able to open the document, with full permissions.</td>
</tr>
<tr>
<td><code>pdf.security.user.password</code></td>
<td><code>&lt;USER PASSWORD&gt;</code></td>
<td>People using this password will be able to open the document, but they will not be able to print.</td>
</tr>
<tr>
<td><code>pdf.security.restrict.print</code></td>
<td>yes</td>
<td>Restricts users from printing.</td>
</tr>
<tr>
<td><code>pdf.security.restrict.copy</code></td>
<td>yes</td>
<td>Restricts users from copying content.</td>
</tr>
</tbody>
</table>

**Important:**

If you specify just the user password (without an owner password), then the people using it will be considered owners, and no restrictions will apply to them.
10. 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.

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.

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 Edit > Preferences > Page Display > Enhance Thin Lines. Deselecting this option makes thin lines displayed as a row of gray pixels (through antialiasing) and they do not disappear. You can experiment by selecting and deselecting the option.

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-forms> 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:

```xml
<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:

```css
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:
or you can use a more complex XPath expression like this:

```xml
<title:before{
  content: oxy_xpath("let $cm:= format-date(current-date(), '[Mn]') \ return concat( \ if ($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  '' \ , \ ', \ '");
}
```
Make sure the entire expression is rendered blue in the CSS editor. Replace the capitalized month names with the translation in the desired language.

**Missing Title when Publishing Single Topics with PDF/UA-1**

**Problem**

When publishing a single topic using the **DITA PDF - based on HTML5 & CSS** transformation scenario and the `pdf.accessibility` parameter set to **yes**, the following error appears: `PDFConformanceException: PDF/UA-1 requires the title to be set`.

**Cause**

By default, the **DITA PDF - based on HTML5 & CSS** transformation scenario does not set the topic title in the metadata.

**Solution**

The title must be declared by using the following CSS rule:

```css
*[class ~="topic/topic"] > *[class ~="topic/title"] {  
  oxy-pdf-meta-title: content();
}
```

**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:

```xml
<?oxy_custom_start type="oxy_content_highlight" color="140,255,140"?>
<?oxy_custom_end?>
```

For comments:
Unexpected Page Break Before or After an Element

Problem

A page break is present before or after an element that has `page-break-before` or `page-break-after` (break-before or break-after) set to `avoid`.

Cause

An empty element (for example `<p>` or `<shortdesc>`) is present before or after the element with the break set to avoid.

Solution

Either remove the empty element from the DITA source topic or set the display to `none` using the following CSS rule:

```css
*[class ~="topic/shortdesc"]:empty {
  display: none;
}
```
11. 

Glossary

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.

DITA Map

A DITA map is a component of the DITA framework (on page 265) 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 265) 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 for your DITA Open Toolkit distribution.

Framework

A framework refers to a package that contains resources and configuration information to provide ready-to-use support for an XML vocabulary or document type. A framework is associated to an XML
document type according to a set of rules. It also includes a variety of settings that improve editing capabilities for its particular file type.

**Oxygen Publishing Template**

*Oxygen Publishing Template* defines all the aspects related with the **look and feel**(*layout and styles*) for the *WebHelp Responsive* output.

The template is self-contained and packed as a ZIP archive making it easy to share with others. It represents the main method for customizing the *WebHelp Responsive* output.
Index

A
Abbreviated-form element
214
Accessibility
160, 160
Add
  Strings
  256
Appendices
148
  Page breaks
  149
Archiving
161, 161
Avoid line breaks at hyphens
159

B
Back matter
104
  Style topics
  105
Bookmap styling
218
Bookmarks
136
  Change labels
  137
Depth
137
Initial state
138
Remove numbering
138
Sections display
137

C
Changing the cover page
217
Changing the page size
216
Changing the TOC depth
216
Command Line
25
Comments
167
  Styling
  171
Cover page
79
  Add empty pages
  88
  Add second cover
  86
  Add text
  84
  Background image
  81
  Copyright page
  89
  Place cover on left side
  85
  Place cover on right side
  85
  Title styling
  84
Creating publishing templates
35
Custom transformation parameters
215
Customization CSS
41
Customize
  Strings
  254
Debugging
42
  XPath expressions
  46, 46
Deep numbering
111
Double sided pagination
132
 Force even number of pages
134
 Force odd number of pages
134
 Start chapters on odd page
133
 Style blank pages
133
 Style first page
134

Draft watermarks
173
 Conditional draft watermark
174, 174

E
Editing publishing templates
38

F
Figures
 Figure numbering
192
Flagging content
175
Fonts
162
 Asian languages
165
 Content
164
 Music
166
 Titles
164

Footer
54
 Add copyright
62
 Add topics

63
 Chapter Number
75
 Page Number
75

Footnotes
149
 Reset Counter
150
 Style Markers and Calls
149
 Force line breaks at hyphens
159

Front matter
104
 Page breaks
105
 Style topics
105

H
Header
54
 Add background image
64
 Add links
65
 Change header at each chapter
58, 60
 Change separators
68
 Changing the heading
73
 Changing the heading language
75
 Only keep chapter title
68
 Style text
66
 Styling
69
 Underlined header
72, 131, 132
XPath
70
Header and Footer
  Chapter first page
  56
Font
  58
Position
  55
Size
  57
Help resources
  10
Hyphenation
  155, 159
    Define for a word
    159
    Enable or disable for elements
    158
    Entire map
    157
I
  i18n
  254
Image maps
  193
Image not found
  193
Images
  187
    Centering
    191
    Control image size
    190
Resolution
  188
Rotate
  189
Side by side with text
  189
Size
  188
Wide
  189
Increase memory
  14
Index
  139
Add a leader
  144
Change number format
  145
Change style and letters
  143
Style labels
  143
Table style layout
  145
Integration Server
  Jenkins
  25
L
  Links
  185
List of Figures
  130
List of Tables
  130
Localization
  254
M
  MathML equation customization
  180
Memory issues
  14
Metadata
  91
    Changing the keywords
    100
    Changing the title property
    101
Cover page
  97
Custom
Removing the title property
101
Title property
101, 101
Use key value in CSS
102
Modify
Strings
255

Notes
212
Numbering
107, 111
Reset page numbering
116

Out of memory
14
OutOfMemory
14
Oxygen Styles Basket
35

Page breaks
76
Add a blank page after a topic
78
Avoid in lists and tables
76
Enforce number of lines
79
Force before or after topic
77
Page size
52
Change setting for an element
54
Changing
53
Orientation
54
PDF Processors
11
Permissions
258
Process overview
11
Programming Elements
207
Protect PDF
258
Publishing Template
Creating
35
Editing
38
Sharing
40

Restrict access to PDF content
258
Security
258
Security permissions
258
Sharing publishing templates
40
Single topic
Cover page
91
Styling
<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Change header</td>
<td>122</td>
</tr>
<tr>
<td>Display short description in TOC</td>
<td>124</td>
</tr>
<tr>
<td>Display topic before TOC</td>
<td>124</td>
</tr>
<tr>
<td>Increase depth</td>
<td>120</td>
</tr>
<tr>
<td>Remove TOC entries</td>
<td>125</td>
</tr>
<tr>
<td>Start on odd page</td>
<td>124</td>
</tr>
<tr>
<td>Style</td>
<td>121</td>
</tr>
<tr>
<td>Centering</td>
<td>200</td>
</tr>
<tr>
<td>Customize</td>
<td></td>
</tr>
<tr>
<td>Cells</td>
<td>201</td>
</tr>
<tr>
<td>Columns</td>
<td>201</td>
</tr>
<tr>
<td>Rows</td>
<td>201</td>
</tr>
<tr>
<td>Layout</td>
<td>197</td>
</tr>
<tr>
<td>Rotate</td>
<td>198</td>
</tr>
<tr>
<td>Small images</td>
<td>200</td>
</tr>
<tr>
<td>Split Cell</td>
<td></td>
</tr>
<tr>
<td>Borders</td>
<td>204</td>
</tr>
<tr>
<td>Stripes</td>
<td></td>
</tr>
<tr>
<td>Text bleeding</td>
<td>199</td>
</tr>
<tr>
<td>Wide</td>
<td>198</td>
</tr>
<tr>
<td>Zebra stripes</td>
<td>203</td>
</tr>
<tr>
<td>Tasks</td>
<td>213</td>
</tr>
<tr>
<td>Change prefix</td>
<td>178</td>
</tr>
<tr>
<td>Layout</td>
<td>177</td>
</tr>
<tr>
<td>Remove prefixes</td>
<td>178</td>
</tr>
<tr>
<td>Separate page</td>
<td>179</td>
</tr>
<tr>
<td>Tracked changes</td>
<td>167</td>
</tr>
<tr>
<td>Style footnotes</td>
<td>171</td>
</tr>
<tr>
<td>Styling</td>
<td>171</td>
</tr>
<tr>
<td>Trademarks</td>
<td>215</td>
</tr>
<tr>
<td>Transformation parameters</td>
<td>15</td>
</tr>
<tr>
<td>Translation</td>
<td>254</td>
</tr>
<tr>
<td>Abbreviated-form</td>
<td>261</td>
</tr>
<tr>
<td>Cell borders missing</td>
<td>260</td>
</tr>
<tr>
<td>Date formatting issues</td>
<td>261</td>
</tr>
<tr>
<td>Disappearing lines</td>
<td>260</td>
</tr>
<tr>
<td>Error parsing</td>
<td>259</td>
</tr>
<tr>
<td>Failed to run pipeline error</td>
<td></td>
</tr>
</tbody>
</table>
Glossentry
260
Glossgroup
261
Highlights span off page
263
I/O exception
259
PDF file is damaged
259
PDF/UA-1
   Requires title
263
Unexpected Page Break
264
Unknown host
259

V
Videos
196
   Key Reference
196