

XMLmind DITA Converter Manual

Hussein Shafie

XMLmind Software
35, rue Louis Leblanc
78120 Rambouillet
France

Phone: +33 (0)9 52 80 80 37
ditac-support@xmlmind.com
www.xmlmind.com/ditac/

May 18, 2018

Table of Contents

List of Figures	ii
List of Tables	iii
Introduction	iv
Part I. Using XMLmind DITA Converter	1
Chapter 1. Installing XMLmind DITA Converter	2
1. Contents of the installation directory	3
Chapter 2. Getting started	6
1. Using the ditac command-line utility	6
Chapter 3. The ditac command-line utility	14
Chapter 4. XSLT stylesheets parameters	22
1. Page headers and footers	49
Chapter 5. Controlling the numbering of ordered lists	54
Chapter 6. Giving a background color to table cells	55
Chapter 7. Syntax highlighting	56
Chapter 8. Rich media content	58
Part II. Customizing the output of XMLmind DITA Converter	65
Chapter 9. Simple customization	66
1. Customize the look of the HTML pages generated by ditac	66
2. Customizing the look of the PDF files generated by ditac	67
Chapter 10. Using ditac to convert documents conforming to a DITA specialization	69
Chapter 11. Extensive customization	72
Part III. Embedding XMLmind DITA Converter in a Java™ application	79
Chapter 12. High-level method: embedding com.xmlmind.ditac.convert.Converter	80
Chapter 13. Low-level method: embedding com.xmlmind.ditac.preprocess.PreProcessor	82
Appendix A. About DITA support in XMLmind DITA Converter	87
Appendix B. Limitations and implementation specificities	89
Appendix C. Translating the messages generated by ditac	96
Index	i

List of Figures

1. XMLmind XSL Utility main window	iv
2-1. XMLmind XSL Utility main window	6
4-1. Page areas	49
4-2. Layout of a header	49
8-1. My name is Hubble. I'm a 7-month old Golden Retriever.	64
11-1. The intermediate files generated by the ditac preprocessor	72

List of Tables

2-1. Supported filename extensions	7
2-2. Supported output formats	9

Introduction

XMLmind DITA Converter (*ditac* for short) allows to convert the most complex DITA 1.0, 1.1, 1.2 and **1.3** documents to production-quality XHTML 1.0, XHTML 1.1, HTML 4.01, Web Help, Java™ Help, HTML Help, Eclipse Help, EPUB, PDF, PostScript®, RTF (can be opened in Word 2000+), WordprocessingML (can be opened in Word 2003+), Office Open XML (.docx, can be opened in Word 2007+), OpenOffice (.odt, can be opened in OpenOffice/LibreOffice 2+).

The first part of this document explains how to install and use *ditac*. The target audience for this part is the DITA author.

The second part of this document explains how to customize the output of *ditac*. The target audience for this part is the DITA consultant.

The third part of this document explains how to embed *ditac* in a Java™ application. The target audience for this part is the Java™ programmer.

You'll find at the end of this document an appendix detailing the limitations and implementation specificities of *ditac*. Please refer to this appendix before posting support requests to the ditac-support+xmlmind.com, public, moderated, mailing list.

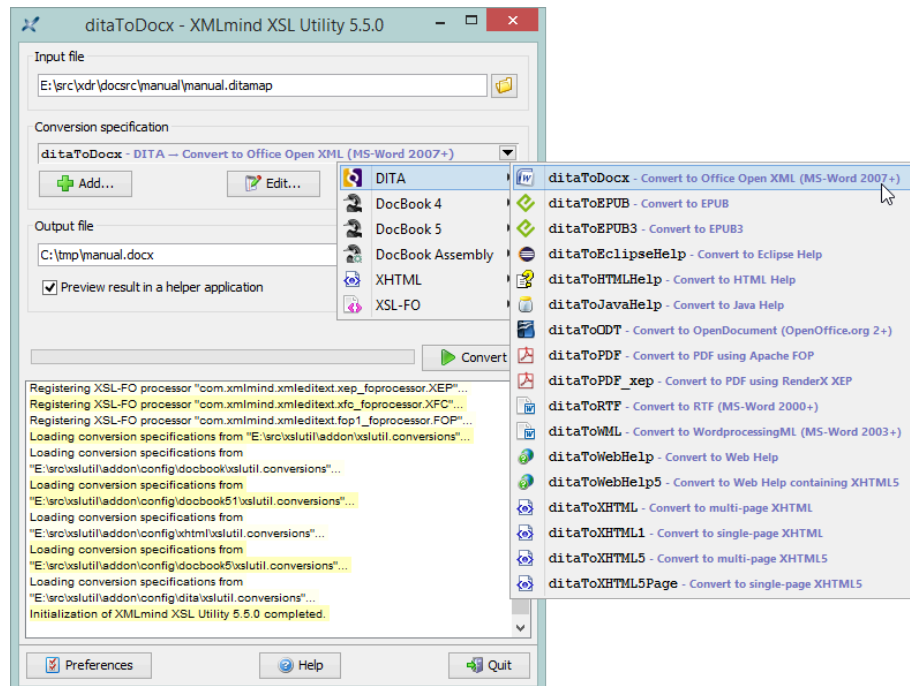


Tip

XMLmind DITA Converter has been integrated to [XMLmind XSL Utility](#), which is part of the [XMLmind XSL-FO Converter](#) commercial product.

Unlike *ditac*, which is a command-line utility, XMLmind XSL Utility is a graphical tool. It makes it easy parameterizing the DITA conversion process and then performing document conversions.

Figure 1. XMLmind XSL Utility main window



Moreover, this graphical tool comes in a Windows, auto-installable, self-contained, `set-up.exe` distribution⁽¹⁾ which includes [Apache FOP](#), [XMLmind XSL-FO Converter](#) and `ditac`.

If you just want to quickly and easily evaluate all the potential of `ditac`, you may want to download XMLmind XSL Utility Evaluation Edition from [XMLmind XSL-FO Converter web site](#). Do not be surprised because XMLmind XSL Utility Evaluation Edition generates output containing *random duplicate letters*. Of course, this does not happen with Professional Edition!

⁽¹⁾Of course, a `.zip` distribution is also available for platforms other than Windows.

Part I. Using XMLmind DITA Converter

Chapter 1. Installing XMLmind DITA Converter

Before you begin

XMLmind DITA Converter (*ditac* for short) requires the Oracle or Apple Java™ runtime 1.6 or above.

On Unix, make sure that the Java `bin/` directory is referenced in the `$PATH` and, at the same time, check that the Java runtime in the `$PATH` has the right version:

```
$ java -version
```

On Windows and on the Mac, this verification is in principle not needed as the `java` executable is automatically found in the `$PATH` when Java has been properly installed.

Procedure

1. Unzip the distribution in any directory you want.

```
C:\> mkdir ditac
C:\> cd ditac
C:\ditac> unzip ditac-3_2_5.zip
C:\ditac> dir ditac-3_2_5
... <DIR> bin
... <DIR> doc
... <DIR> docsrc
...
```

XMLmind DITA Converter is intended to be used directly from the `ditac-3_2_5/` directory. That is, you can run the `ditac` command by simply executing (in a Command Prompt on windows, a terminal on Unix):

```
C:\ditac> ditac-3_2_5\bin\ditac
```

2. Depending the output formats you want to generate, you'll need to download and install third-party external tools.
 - If you want to generate PDF or PostScript®, download and install [Apache FOP](#).

Alternatively, you may prefer to purchase [RenderX XEP](#) or [Antenna House Formatter](#). Note that [RenderX XEP Personal Edition](#) is free to use.



WARNING

Please install and use either Apache FOP 1.1 or [FOP 2.1+](#). Please do *not* install and use Apache [FOP 2.0](#) as we have found this version to have a severe bug ([FOP-2461](#)).



Note

If you have installed Apache FOP and your DITA document contain [MathML](#), you'll want to also install the [JEUclid FOP plug-in](#). This plug-in is needed to add MathML support to Apache FOP.

- If you want to generate RTF (can be opened in Word 2000+), WordprocessingML (can be opened in Word 2003+), Office Open XML (.docx, can be opened in Word 2007+) or OpenOffice (.odt, can be opened in OpenOffice/LibreOffice 2+), then you need to purchase [XMLmind XSL-FO Converter Professional Edition](#).

You can give XMLmind XSL-FO Converter a try by downloading Evaluation Edition from [XMLmind XSL-FO Converter web site](#). Do not be surprised because XMLmind XSL-FO Converter Evaluation Edition generates output containing *random duplicate letters*. Of course, this does not happen with Professional Edition!

- If you want to generate HTML Help, download and install the [HTML Help Workshop](#) (contains `hhc.exe`).
 - If you want to generate Java Help, download and install [Java Help](#) (contains `jhindexer` and `jhindexer.bat`).
3. If you have installed any of the above external tools, you need now to instruct ditac where to find them. This can be done using the following command line options: `-fop`, `-xep`, `-ahf`, `-xfc`, `-jhindexer`, `-hhc`. However, it is much more convenient to specify these command-line options once for all in a `ditac.options` file.

- a. Create `ditac.options`, a plain text file encoded using the native encoding of the platform (e.g. windows-1252 on a Western Windows PC), in the ditac user preferences directory.

The ditac user preferences directory is:

- `$HOME/.ditac/` on Linux.
- `$HOME/Library/Application Support/XMLmind/ditac/` on the Mac.
- `%APPDATA%\XMLmind\ditac\` on XP, Vista, 7, 8, 10.

Example: `C:\Documents and Settings\john\Application Data/XMLmind\ditac\` on Windows XP. `C:\Users\john\AppData\Roaming/XMLmind\ditac\` on Windows Vista, 7, 8, 10.

- b. Add the equivalent of a command-line option for each external tool installed in the preceding step. Use one or more newline characters to separate the options. More information in [The ditac.options file](#).

```
-fop E:\opt\fop-2.2\fop\fop.bat

-xfc E:\opt\xfc_eval_java-5_4_6\bin\fo2rtf.bat

-jhindexer E:\opt\javahelp\javahelp\bin\jhindexer.bat

-hhc "C:\Program Files\HTML Help Workshop\hhc.exe"
```

1. Contents of the installation directory

bin/ditac, ditac.bat

Scripts used to run XMLmind DITA Converter (*ditac* for short). Use `ditac` on any Unix system. Use `ditac.bat` on Windows.

doc/index.html

Contains the documentation of ditac. *XMLmind DITA Converter Manual* is available in all the output formats supported by ditac. You'll also find there the reference manual of the API of ditac (generated by `javadoc`).

docsrc/manual/

Contains the DITA source of *XMLmind DITA Converter Manual*.

LEGAL/, LEGAL.txt

Contains legal information about ditac and about third-party components used in ditac.

lib/

All the (non-system) Java™ class libraries needed to run ditac:

ditac.jar

contains the code of XMLmind DITA Converter.

resolver.jar

is [Apache XML Commons Resolver](#) which implements catalog-based entity and URI resolution.

relaxng.jar

is [Jing](#) version 20030619, James Clark's RELAX NG validator, slightly modified for use in [XMLmind XML Editor](#) and XMLmind DITA Converter. The details of the modifications are found in `LEGAL/relaxng.README`.

saxon9.jar

is Michael Kay's XSLT 2.0 engine. See <http://www.saxonica.com/>.

whcmin.jar**snowball.jar**

contains the code needed to run [XMLmind Web Help Compiler](#).

xslthl.jar

contains the code of the [XSLT syntax highlighting](#) open source software component.

plus/

This directory is present only in the case of the `ditac-N_N_N-plus-fop.zip` distribution. It contains most recent [Apache FOP](#) (including hyphenation and MathML support). This XSL-FO processor is automatically declared and thus, ready to be used to generate PDF or PostScript.

schema/

Contains the DTD, RELAX NG and W3C XML schemas of DITA 1.3 1.2, 1.1, 1.0.1. File `schema/catalog.xml` contains an XML catalog which points to these local copies.

src/

Contains the Java source code of ditac. `src/build.xml` is an [ant](#) build file which allows to rebuild `lib/ditac.jar`.

whc_template/

Contains the template directory of [XMLmind Web Help Compiler](#).

xsl/

Contains the [XSLT 2.0](#) stylesheets used to convert DITA documents to a variety of formats.

fo/fo.xsl

Used to generate an intermediate XSL-FO file. After that, the XSL-FO file is converted to PDF , PostScript , RTF , WordprocessingML , Office Open XML (.docx) or OpenOffice/LibreOffice (.odt) by the means of an XSL-FO processor.

xhtml/xhtml.xsl

Used to generate XHTML 1.0 pages.

xhtml/xhtml1_1.xsl

Used to generate XHTML 1.1 pages.

xhtml/html.xsl

Used to generate HTML 4.01 pages.

xhtml/xhtml5.xsl

Used to generate XHTML 5 pages.

webhelp/webhelp.xsl

Used to generate Web Help containing XHTML 1 pages, which are then compiled using [XMLmind Web Help Compiler](#).

webhelp/webhelp5.xsl

Used to generate Web Help containing XHTML 5 pages, which are then compiled using [XMLmind Web Help Compiler](#).

htmlhelp/htmlhelp.xsl

Used to generate HTML Help files, which are then compiled using `hhc.exe`.

eclipsehelp/eclipsehelp.xsl

Used to generate Eclipse Help files.

javahelp/javahelp.xsl

Used to generate Java™ Help files, which are then archived in a `.jar` file.

epub/epub.xsl

Used to generate EPUB 2 files, which are then archived in a `.epub` file (Zip archive having a `.epub` extension).

epub/epub3.xsl

Used to generate EPUB 3 files, which are then archived in a `.epub` file (Zip archive having a `.epub` extension).

Chapter 2. Getting started

1. Using the `ditac` command-line utility

In this chapter, we'll explain how to run the `ditac` command-line utility by using examples. You'll find all the DITA input files used to run the following examples in the `ditac_install_dir/docsrc/manual/` directory.

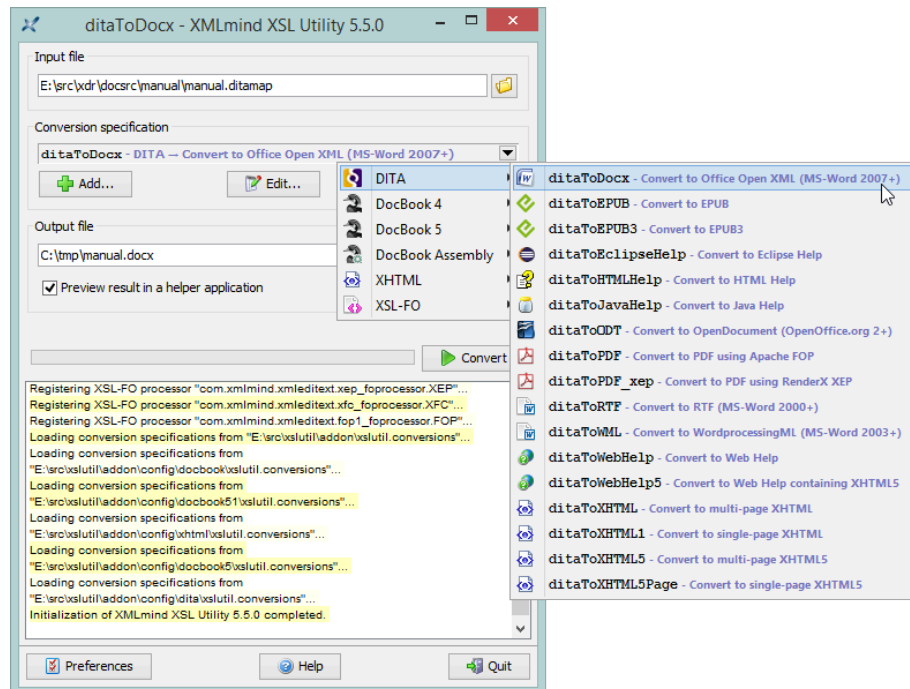


Tip

XMLmind DITA Converter has been integrated to XMLmind XSL Utility, which is part of the XMLmind XSL-FO Converter commercial product.

Unlike `ditac`, which is a command-line utility, XMLmind XSL Utility is a graphical tool. It makes it easy parameterizing the DITA conversion process and then performing document conversions.

Figure 2-1. XMLmind XSL Utility main window



Moreover, this graphical tool comes in a Windows, auto-installable, self-contained, `set-up.exe` distribution⁽²⁾ which includes Apache FOP, XMLmind XSL-FO Converter and `ditac`.

If you just want to quickly and easily evaluate all the potential of `ditac`, you may want to download XMLmind XSL Utility Evaluation Edition from XMLmind XSL-FO Converter web site. Do not be surprised because XMLmind XSL Utility Evaluation Edition generates output containing *random duplicate letters*. Of course, this does not happen with Professional Edition!

⁽²⁾Of course, a `.zip` distribution is also available for platforms other than Windows.

Converting a document to PDF

Converting a document to PDF is done by executing the following command:

```
$ ditac out/manual.pdf manual.ditamap
```

The output directory `out/` is automatically created if it does not already exist.

Unless you have specified in the `ditac.options` file which XSL-FO processor to use, you'll have to execute:

```
$ ditac -fop /opt/fop/fop out/manual.pdf manual.ditamap
```

or:

```
$ ditac -xep /opt/xep/xep out/manual.pdf manual.ditamap
```

or:

```
$ ditac -ahf "C:\AHFv6\AHFCmd.exe" out/manual.pdf manual.ditamap
```



Tip

No need to declare Apache FOP using the `-fop` command-line option if you have installed the distribution called `ditac-N_N_N-plus-fop.zip`. This distribution contains most recent Apache FOP (including hyphenation and MathML support). This XSL-FO processor is automatically declared and thus, ready to be used to generate PDF or PostScript.

The XSL-FO processors allowing to generate PDF also allows to generate PostScript®. Example:

```
$ ditac out/manual.ps manual.ditamap
```

Notice how the output format is determined by examining the filename extension of the output file.

Table 2-1. Supported filename extensions

Format	Extensions
XHTML 1.0	.html, .htm, .xhtml
EPUB 2	.epub
HTML Help	.chm
Java Help	.jar
PDF	.pdf
PostScript®	.ps
RTF (can be opened in Word 2000+)	.rtf, .doc
WordprocessingML(can be opened in Word 2003+)	.wml, .xml
Office Open XML (can be opened in Word 2007+)	.docx
OpenOffice (can be opened in OpenOffice/Libre-Office 2+)	.odt

Note that **ditac** also allows to convert one or more topic files rather than a single map or bookmap file:

```
$ ditac -toc \  
out/draft.pdf embed1.dita embed2.dita
```

Ditac does not generate a table of contents (TOC) by default. Unless the input file is a bookmap containing an empty `toc` element in its `frontmatter/booklists` descendant element, you'll have to explicitly use the `-toc` option. Using the `-toc` option when the input file already specifies a TOC is harmless, so you could as well add a `-toc` line to your `ditac.options` file.

Converting a document to a word processor format

Converting a document to a word processor format just requires the use of an XSL-FO processor different from the one which generates PDF or PostScript. Fortunately all this automatically handled by **ditac**.

Convert a document to RTF (can be opened in Word 2000+):

```
$ ditac out/manual.rtf manual.ditamap
```

Unless you have specified in the `ditac.options` file which XSL-FO processor to use, you'll have to execute:

```
$ ditac -xfc /opt/xfc/fo2rtf out/manual.rtf manual.ditamap
```

Suffice to specify the location of `fo2rtf` (`fo2rtf.bat` on Windows). Using this location, `ditac` infers the locations of `fo2wml`, `fo2docx` and `fo2odt`.

Convert a document to WordprocessingML (can be opened in Word 2003+):

```
$ ditac out/manual.xml manual.ditamap
```

Convert a document to Office Open XML (can be opened in Word 2007+):

```
$ ditac out/manual.docx manual.ditamap
```

Convert a document to OpenOffice (can be opened in OpenOffice.org 2+):

```
$ ditac -v -p number all \  
out/manual.odt manual.ditamap
```

Useful options

- `-v` instructs **ditac** to print progress messages on the console. Recommended when converting large documents.
- `"-p number all"` passes parameter "number" with value "all" to the XSLT stylesheets which generate the XSL-FO. The XSL-FO are then converted to OpenOffice format by the means of XMLmind XSL-FO Converter. The `number='all'` parameter instructs the XSLT stylesheets to number topics, tables and figures.

Converting a document to XHTML

Converting a document to multi-page XHTML 1.0 is done by executing the following command:

```
$ ditac -images img -p xsl-resources-directory res \  
out/manual.xhtml manual.ditamap
```

```
out/manual/_.html manual.ditamap
```

- All the files generated by **ditac** are created in the `out/manual/` directory.
- `-images img` instructs **ditac** to copy all the image files referenced by the input DITA document to `out/manual/img/`. Specifying the `-images` option when generating an output format based on XHTML/HTML is needed in almost all the use cases.
- `-p xsl-resources-directory res` instructs **ditac** to copy all the resources needed by the XSLT stylesheets (CSS stylesheet, navigation icons, etc) to `out/manual/res/`. Specifying a value for the `xsl-resources-directory` parameter when generating an output format based on XHTML/HTML is needed in almost all the use cases.
- Notice the strange name of the output file: `out/manual/_.html`. In fact, this name is just used to specify the filename extension of the output files. The actual basenames of the output files are determined by examining the `chunk` and `copy-to` attributes possibly specified in the DITA map.

Note that a command-line like:

```
$ ditac -images img -p xsl-resources-directory res \
  out/manual/foo.html manual.ditamap
```

works fine too. The only difference is that in such case, when a basename is needed and cannot be determined by examining the `chunk` and `copy-to` attributes specified in the DITA map, **ditac** will use "foo" as a basename and you *may* end up having some output files called `foo.html`, `foo-2.html`, `foo-3.html`, etc. When the basename is specified as "_", it is the basename of the DITA map which is used. That is, you may have some output files called `manual.html`, `manual-2.html`, `manual-3.html`, etc.

What if you want to convert a document to HTML 4.01 or XHTML 1.1 or XHTML 5 rather than to XHTML 1.0? We have learned that there is no way to specify this using a [filename extension](#). The answer is: use the `-format` option (or `-f` in its short form). Example:

```
$ ditac -format html \
  -images img -p xsl-resources-directory res \
  out/manual/_.html manual.ditamap
```

Table 2-2. Supported output formats

Format	Name
XHTML 1.0	xhtml
XHTML 1.1	xhtml1.1
HTML 4.01	html
XHTML 5	xhtml5. html5 is an alias for xhtml5.
Web Help containing XHTML 1 pages	webhelp
Web Help containing XHTML 5 pages	webhelp5
HTML Help	htmlhelp
Eclipse Help	eclipsehelp
EPUB 2	epub
EPUB 3	epub3

Format	Name
Java Help	javahelp
PDF	pdf
PostScript®	ps
RTF (can be opened in Word 2000+)	rtf
WordprocessingML(can be opened in Word 2003+)	wml
Office Open XML (can be opened in Word 2007+)	docx
OpenOffice (can be opened in OpenOffice.org 2+)	odt
XSL-FO	fo

Useful options

- "-p `chain-pages both`". This XSLT stylesheet parameter specifies that a header and a footer containing navigation icons should be generated in order to link together all the HTML pages.
- "-p `chain-topics yes`". This XSLT stylesheet parameter specifies that navigation icons should be generated in order to link together all the topics.
- "-p `default-table-width 100%`". Unless this XSLT stylesheet parameter is specified (or the `expand="page"` attribute is specified for all tables), web browsers tend to layout the generated HTML tables in order to make them as narrow as possible.

A full-fledged command-line is thus:

```
$ ditac -images img -p xsl-resources-directory res \
  -p number all \
  -p chain-pages both \
  -p chain-topics yes \
  -p default-table-width 100% \
  out/manual/_.html manual.ditamap
```

What if you want to generate a single XHTML page rather than multiple XHTML page? No need to create a new DITA map for that. Simply specify option "`-chunk single`" (or `-c` in its short form).

```
$ ditac -chunk single \
  -images img -p xsl-resources-directory res \
  out/manual.html manual.ditamap
```

Converting a document to Web Help

Converting a document to [Web Help](#) is similar to converting a document to [multi-page XHTML 1](#). The main difference is that you need to explicitly specify `-format webhelp`:

```
$ ditac -format webhelp \
  -images img -p xsl-resources-directory res \
  webhelp/_.html manual.ditamap
```


If you prefer to generate Web Help containing XHTML 5 pages rather than XHTML 1 pages, then specify `-format webhelp5`.

**Remember**

Do not specify any of the following command-line options when generating Web Help: `-toc`, `-index`.

Converting a document to HTML Help

Converting a document to HTML Help is done by executing the following command:

```
C:\> ditac -images img -p xsl-resources-directory res \
out\manual.chm manual.ditamap
```

Unless you have specified in the `ditac.options` file the location of `hhc.exe`, you'll have to execute:

```
C:\> ditac -hhc "C:\Program Files\HTML Help Workshop\hhc.exe" \
-images img -p xsl-resources-directory res \
out\manual.chm manual.ditamap
```

**Remember**

Do not specify any of the following command-line options when generating HTML Help: `-toc`, `-index`.

Converting a document to Java™ Help

Converting a document to Java™ Help is done by executing the following command:

```
$ ditac -images img -p xsl-resources-directory res \
out/manual.jar manual.ditamap
```

Unless you have specified in the `ditac.options` file the location of `jhindexer` (`jhindexer.bat` on Windows), you'll have to execute:

```
$ ditac -jhindexer /opt/jh2.0/javahelp/bin/jhindexer \
-images img -p xsl-resources-directory res \
out/manual.jar manual.ditamap
```

**Remember**

Do not specify any of the following command-line options when generating Java™ Help: `-toc`, `-index`.

Converting a document to Eclipse Help

Converting a document to [Eclipse Help](#) is similar to converting a document to multi-page XHTML. The main difference is that you need to explicitly specify `-format eclipsehelp`:

```
$ ditac -format eclipsehelp \  
  -images img -p xsl-resources-directory res \  
  out/com.acme.widget.userguide/_ .html manual.ditamap
```

In order to deploy the generated Eclipse Help, you need to copy the output directory as a whole (`com.acme.widget.userguide/` in the case of the above example) to the `plugins/` directory of Eclipse and then use a text or XML editor to modify the generated `output_directory/plugin.xml`:

```
<plugin name="EDIT HERE: title of this help"  
  id="EDIT HERE: unique.id.of.this.plugin"  
  provider-name="EDIT HERE: author, company or organization"  
  version="1.0.0">  
  <extension point="org.eclipse.help.toc">  
    <toc file="toc.xml" primary="true"/>  
  </extension>  
  <extension point="org.eclipse.help.index">  
    <index file="index.xml"/>  
  </extension>  
</plugin>
```

If you do not want to hand edit `plugin.xml`, suffice to pass extra XSLT stylesheet parameters to `ditac`:

```
$ ditac -format eclipsehelp \  
  -p plugin-name "ACME Widget User's Guide" \  
  -p plugin-id com.acme.widget.userguide \  
  -p plugin-provider "ACME Corp." \  
  -images img -p xsl-resources-directory res \  
  out/com.acme.widget.documentation/_ .html manual.ditamap
```



Remember

If you want to see your document by selecting **Help** → **Help Contents** in Eclipse:

1. Do not specify any of the following command-line options when generating Eclipse Help: `-toc`, `-index`.
2. Parameter `plugin-id` is required to have the same value as the basename of the the output directory (`com.acme.widget.userguide/` in the case of the above example).
3. Copy this output directory to `eclipse_install_dir/dropins/` and not `eclipse_install_dir/plugins/`.

Converting a document to EPUB

Converting a document to [EPUB Help](#) is done by executing the following command:

```
$ ditac -images img -p xsl-resources-directory res \  
  out/com.acme.widget.documentation/_ .html manual.ditamap
```

```
out/manual.epub manual.ditamap
```

If you prefer to generate EPUB 3 rather than EPUB 2, then specify `-format epub3`.

**Remember**

Do not specify any of the following command-line options when generating EPUB: `-toc`.
Note that you may specify option `-index`.

Related information

- Chapter 3. The `ditac` command-line utility

Chapter 3. The `ditac` command-line utility

```
ditac [option]* output_file [in_dita_file]+
```

Command-line usage

Converts specified DITA input files to specified output file.

The input files must comprise a single map or bookmap file or possibly several, possibly multi-topic, topic files.

Example: convert the `userguide.ditamap` map to multi-page XHTML:

```
C:\docsrc> ditac -p center "fig table" ..\doc\userguide.htm userguide.ditamap
```

Example: convert the `introduction.dita` and `quickstart.dita` topics to PDF:

```
C:\docsrc> ditac draft1.pdf introduction.dita quickstart.dita
```

An input file may be specified using its URL or its filename.

The output directory is created if it does not already exist.

In some case, there is no need to specify a real output filename: the output directory and the extension of the output files suffice. In such case, specify "_" as the basename of the output file.

Example: convert `foo.ditamap` to multi-page XHTML. The XHTML pages must be generated in the `bar/` subdirectory.

```
C:\docsrc> ditac bar\_\.html foo.ditamap
```

In the above case, the basenames of the generated XHTML pages will be taken from the `@chunk` and `@copy-to` attributes specified in `foo.ditamap` if any, and from the basename of the map ("`foo`" in the case of our example) otherwise.

Commonly used command-line options

Some options have both a short name and a long name. Example: `-p` is equivalent to `-param`.

`-p param_name param_value`

`-param param_name param_value`

Specifies a XSLT stylesheet parameter. See [Chapter 4](#).

`-t XSLT_stylesheet_URL_or_file`

`-xslt XSLT_stylesheet_URL_or_file`

Use the specified custom XSLT stylesheet rather than the stock one.

`-c none|single|auto`

`-chunk none|single|auto`

The "none" and "single" values may be used to force the generation of a single output file.

For example, "`-chunk single`" allows the reuse of a map designed to output multiple HTML pages in order to generate a PDF file.

For example, "`-chunk none`" allows the reuse of a map designed to output a PDF file in order to generate a single HTML page.

By default, the chunk mode is `auto` which means: generate a single output file (implicit "`-chunk none`") for formats such as `pdf`, `ps`, `rtf`, etc, and generate multiple output files for formats such as `html`, `xhtml`, `javahelp`, etc.

-f `xhtml` | `xhtml1.1` | `html` | `xhtml5` | `html5` | `webhelp` | `webhelp5` | `epub` | `epub3` | `javahelp` | `htmlhelp` | `ps` | `pdf` | `rtf` | `odt` | `wml` | `docx` | `fo`

-format `xhtml` | `xhtml1.1` | `html` | `xhtml5` | `html5` | `webhelp` | `webhelp5` | `epub` | `epub3` | `javahelp` | `htmlhelp` | `ps` | `pdf` | `rtf` | `odt` | `wml` | `docx` | `fo`

Explicitly specifies the output format. By default, the output format is determined using the extension of `output_file`.

Notes:

- A "htm" or "html" filename extension implicitly specifies an XHTML 1.0 output format, and not an HTML 4.01 output format. In order to generate HTML 4.01, explicitly specify "`-f html`". The same remark applies to `xhtml1.1`, `xhtml5`, `webhelp`, `webhelp5`.
- Option `html5` is simply an alias for `xhtml5`.
- Option `webhelp5` means Web Help containing XHTML 5 pages rather than XHTML 1 pages.
- Option `epub` specifies the EPUB 2 format.

-r *resource_path*

-resources *resource_path*

-i *resource_path*

-images *resource_path*

Copy the resource files, typically image files, referenced in the source topics to specified directory. If specified path is relative, it is relative to the output directory.

-resourcehandler *class_name parameters*

Pass the resource files, typically image files, referenced in the source topics to *class_name*, a Java™ class implementing interface `com.xmlmind.ditac.preprocess.ResourceHandler`. String *parameters* is used to configure the newly created `ResourceHandler`.

For example, "`-r res`" is equivalent to "`-resourcehandler com.xmlmind.ditac.convert.ResourceCopier res`".

-filter *ditaval_URL_or_file*

Apply specified conditional processing profile (`.ditaval` file) to the topics.

-toc

Equivalent to "`-frontmatter toc`".

Note that this option will *not* cause a **Table of Contents** to be generated when the map contains a single `<topicref>`⁽³⁾ having no `<topicref>` descendants.

-index

Equivalent to "`-backmatter indexlist`".

-frontmatter *spec*

Automatically generate specified sections: **Table of Contents**, **List of Tables**, etc, before the other pages.

When used on a `<bookmap>`, this option adds elements *after* any existing `<booklists>` elements.

The syntax of *spec* is:

```
spec -> same_page [ ',' same_page ]*
```

⁽³⁾Not counting `<topicref>`s contained in `<frontmatter>` and `<backmatter>`.

```
same_page -> section [ '+' section ]*

section -> 'toc'|'figurelist'|'tablelist'|'examplelist'|
          'equationlist'|'indexlist'
```

Example: generate the **Table Of Contents** in its own page, followed by another page containing both the **List of Figures** and the **List of Tables**.

```
-frontmatter toc,figurelist+tablelist
```

-backmatter *spec*

Automatically generate specified sections: **Table of Contents**, **List of Tables**, etc, after the other pages. See **-frontmatter** for more information.

When used on a <bookmap>, this option adds elements *before* any existing <booklists> elements.

-addindex

When an output file contains the **Table of Contents** (let's call this file `main.html`) and when no file called `index.html` has been generated, this option copies `main.html` to `index.html`. Applies to formats: `xhtml`, `xhtml1.1`, `html`, `webhelp`.

-lang *language_code*

Specifies the main language of the document

Shorthand for:

```
-foconverter pdf "executable_file"
-foconverter ps "executable_file"
```

. Examples: `"fr"`, `"fr-CA"`. Needed to sort the index entries.

By default, this information is taken from the `@xml:lang` attribute of the root element of the topic map (if any, `"en"` otherwise).

-v

-vv

-vvv

Turn verbosity on. More Vs means more verbose.

-o *options_URL_or_file*

-option *options_URL_or_file*

This option lets the user specify a text file containing command-line arguments. This text file has the same format as the `ditac.options` file.

Example:

```
$ ditac -v -o html.options foo.htm foo.ditamap
```

If `html.options` contains:

```
-format html
-p css http://www.acme.com/css/acme.css
```

then this is equivalent to running:

```
$ ditac -v -format html -p css http://www.acme.com/css/acme.css \
  foo.htm foo.ditamap
```

Command-line options used to configure ditac

-fop executable_file

Specifies the location of the fop shell script (fop.bat on Windows).

Shorthand for:

```
-foconverter FOP pdf "executable_file" -q -r -fo "%I" -pdf "%O"
-foconverter FOP ps "executable_file" -q -r -fo "%I" -ps "%O"
```

-xep executable_file

Specifies the location of the xep shell script (xep.bat on Windows).

Shorthand for:

```
-foconverter XEP pdf "executable_file" -quiet -valid -fo "%I" -pdf "%O"
-foconverter XEP ps "executable_file" -quiet -valid -fo "%I" -ps "%O"
```

-ahf executable_file

Specifies the location of AHFCmd.exe (run.sh on platforms other than Windows).

Shorthand for:

```
-foconverter AHF pdf "executable_file" -x 3 -p @PDF -d "%I\" -o "%O"
-foconverter AHF ps "executable_file" -x 3 -p @PS -d "%I" -o "%O"
```

-xfc executable_file

Specifies the location of the fo2rtf shell script (fo2rtf.bat on Windows).

Suffice to specify the location of fo2rtf. Using this location, ditac infers the locations of fo2wml, fo2docx and fo2odt.

Shorthand for:

```
-foconverter XFC rtf "fo2rtf_executable_file" "%I" "%O"
-foconverter XFC wml "fo2wml_executable_file" "%I" "%O"
-foconverter XFC docx "fo2docx_executable_file" "%I" "%O"
-foconverter XFC odf "fo2odt_executable_file" "%I" "%O"
```

-foconverter processor_name target_format command

Register specified XSL-FO converter with ditac, a lower-level alternative to using **-xep**, **-fop**, **-ahf** or **-xfc**. Example:

```
-foconverter XFC rtf '/opt/xfc/bin/fo2rtf "%I" "%O"'
```

Note that this option can be specified several times with different values in the same command-line.

This low-level option may be used for example to specify a [configuration file for Apache FOP](#):

```
-foconverter FOP pdf \
'/opt/fop/fop -c /home/john/docs/fop.conf -q -r -fo "%I" -pdf "%O"'
```

-jhindexer executable_file

Specifies the location of the jhindexer shell script (jhindexer.bat on Windows), the Java™ Help indexer.

-hhc exe_file

Specifies the location of hhc.exe, the HTML Help compiler.

-plugin *plugin_name*

Use the DTDs/schemas and the XSLT stylesheets found in the plug-in subdirectory having specified name preferably to those found in *ditac_install_dir/schema/* and in *ditac_install_dir/xsl/*. See [What is a plug-in?](#).

Command-line options used to debug ditac

-preprocess

Stop after preprocessing input files.

-automap *save_file*

Save the automatically generated topic map (if any) to specified file.

-keepfo

When generating PDF, RTF, etc, do not delete the temporary XSL-FO file.

-errout

Output all messages, including errors and warnings, to stdout.

-ignoreoptionsfile

Do not load the `ditac.options` options file. See below [The ditac.options file](#).

-validate

Validate *all* the XML files loaded by **ditac**. Any validation error will cause **ditac** to immediately stop running. Therefore the combination of the **-validate** and **-dryrun** options gives you a simple way to thoroughly check your DITA document.

Note that for the **-validate** option to work, *all* the XML files (maps, topics, even `.ditaval` filter files) loaded by **ditac** must start with the proper `<!DOCTYPE>` declaration.

-dryrun

Use **ditac** as a validator, and most notably check cross-references. That is, do not generate any file; just report errors if any .

-version

Print version number and exit.

The ditac.options file

It is also possible to specify command-line options in the `ditac.options` options file. The content of this plain text file, encoded in the native encoding of the platform (e.g. `Windows-1252` on a Western Windows PC), is automatically loaded by **ditac** each time this command is executed. The content of this file, command-line options separated by whitespace, is *prepended* to the options specified in the command-line.

Example: If `ditac.options` contains:

```
-v -p number all
```

Running:

```
~/docsrc$ ditac -p center "fig table" ../doc/userguide.htm userguide.ditamap
```

is equivalent to running:

```
~/docsrc$ ditac -v -p number all -p center "fig table" \
  ../doc/userguide.htm userguide.ditamap
```

The `ditac.options` options file is found in the `ditac` user preferences directory. This directory is:

- `$HOME/.ditac/` on Linux.
- `$HOME/Library/Application Support/XMLmind/ditac/` on the Mac.
- `%APPDATA%\XMLmind\ditac\` on Windows XP, Vista, 7, 8, 10.

Example: `C:\Documents and Settings\john\Application Data\xmlmind\ditac\` on Windows XP. `C:\Users\john\AppData\Roaming\xmlmind\ditac\` on Windows Vista, 7, 8, 10.

The `ditac.options` options file is mainly useful to configure ditac once for all by specifying values for the `-fop`, `-xep`, `-xfc`, `-jhindexer`, `-hhc`, `-plugin` options.

Example:

```
-v
-xep E:\opt\xep\xep.bat
-fop E:\opt\fop-2.2\fop\fop.bat
-xfc E:\opt\xfc_eval_java-5_5_0\bin\fo2rtf.bat
-jhindexer E:\opt\javahelp\javahelp\bin\jhindexer.bat
-hhc "C:\Program Files\HTML Help Workshop\hhc.exe"
```



Remember

- Relative filenames found in this file are relative to the current working directory, and not to the `ditac.options` options file. Therefore it is recommended to always specify absolute filenames.
- No comments (e.g. lines starting with '#') are allowed in `ditac.options`. Options must be separated by whitespace.
- In the above example, FOP is declared *after* XEP. This implies that it is FOP and not XEP, which will be used by ditac to generate PDF and PostScript®.
- An XSL-FO processor tend to consume a lot of memory. If the DITA conversion fails with an out-of-memory error, you need to edit the `xep` (`xep.bat`), `fop` (`fop.bat`), `fo2xxx` (`fo2xxx.bat`) scripts in order to increase the maximum amount of memory that the Java™ runtime may allocate. This is done by using the `-xmx` option of the Java™ command-line. Example: `"java ... -xmx512m ..."`.
- Starting from Java™ 1.6.0_23, converting XML documents to PDF using RenderX XEP randomly fails with false XSL-FO errors (e.g. attribute "space-before" may not be empty). This problem seems specific to the 64-bit runtime.

The workarounds for the above bug ("renderx #22766") are:

- Use a 32-bit Java™ runtime.
- OR Use a 64-bit Java™ runtime older than 1.6.0_23.
- OR Specify option `-valid` in the `xep` command-line. Note that this workaround is automatically used when you specify which RenderX XEP executable to use by the means of the `-xep` command-line option.

What is a plug-in?

A plug-in is simply a subdirectory of `ditac_install_dir/plugin/`. For example, `ditac_install_dir/plugin/MyPlugin/`.

This subdirectory may contain an [XML catalog file](#). This XML catalog file must be named `catalog.xml`. In the case of a DITA specialization, `catalog.xml` points to local copies of customized DTDs. Example: `ditac_install_dir/plugin/MyPlugin/catalog.xml`:

```
<catalog xmlns="urn:oasis:names:tc:entity:xmlns:xml:catalog"
  prefer="public">
  <public publicId="-//OASIS//DTD DITA Concept//EN"
    uri="dtd/concept.dtd"/>
  ...
</catalog>
```

This subdirectory may contain an `xsl/` subdirectory organized *exactly* like `ditac_install_dir/xsl/`. That is, this `xsl/` subdirectory may contain one or more of the following XSLT stylesheets:

XSLT stylesheet	Description
<code>xsl/fo/fo.xml</code>	Used to generate an intermediate XSL-FO file. After that, the XSL-FO file is converted to PDF , PostScript , RTF , WordprocessingML , Office Open XML (.docx) or OpenOffice/LibreOffice (.odt) by the means of an XSL-FO processor.
<code>xsl/xhtml/xhtml.xml</code>	Used to generate XHTML 1.0 pages.
<code>xsl/xhtml/xhtml1_1.xml</code>	Used to generate XHTML 1.1 pages.
<code>xsl/xhtml/html.xml</code>	Used to generate HTML 4.01 pages.
<code>xsl/xhtml/xhtml5.xml</code>	Used to generate XHTML 5 pages.
<code>xsl/webhelp/webhelp.xml</code>	Used to generate Web Help containing XHTML 1 pages, which are then compiled using XMLmind Web Help Compiler .
<code>xsl/webhelp/webhelp5.xml</code>	Used to generate Web Help containing XHTML 5 pages, which are then compiled using XMLmind Web Help Compiler .
<code>xsl/htmlhelp/htmlhelp.xml</code>	Used to generate HTML Help files, which are then compiled using <code>hhc.exe</code> .
<code>xsl/eclipsehelp/eclipsehelp.xml</code>	Used to generate Eclipse Help files.
<code>xsl/javahelp/javahelp.xml</code>	Used to generate Java™ Help files, which are then archived in a <code>.jar</code> file.
<code>xsl/epub/epub.xml</code>	Used to generate EPUB 2 files, which are then archived in a <code>.epub</code> file (Zip archive having a <code>.epub</code> extension).
<code>xsl/epub/epub3.xml</code>	Used to generate EPUB 3 files, which are then archived in a <code>.epub</code> file (Zip archive having a <code>.epub</code> extension).

When `ditac` is passed command-line option `-plugin plugin_name`, it will use the DTDs/schemas and the XSLT stylesheets found in the plug-in subdirectory having specified name preferably to those found in `ditac_install_dir/schema/` and in `ditac_install_dir/xsl/`.

**Tip**

If you don't want your plug-ins to reside inside *ditac_install_dir/plugin/*, you may specify an alternate parent directory by the means of the `DITAC_PLUGIN_DIR` environment variable. Example:

- On Windows:

```
C:\>set DITAC_PLUGIN_DIR=C:\Users\john\ditac_plugins
```

- On Unix:

```
$ export DITAC_PLUGIN_DIR=/home/john/ditac_plugins
```

Related information



- [Chapter 4. XSLT stylesheets parameters](#)


Chapter 4. XSLT stylesheets parameters

Parameters common to all stylesheets




Note

- Parameters marked using this icon  are *system parameters*. They are automatically specified by the application executing the XSLT stylesheets. Such system parameters must not be specified by the end-user. Such system parameters are documented here only because the end-user may see them referenced in some configuration files.
- Parameters marked using this icon  are *pseudo-parameters*. They may or may not be passed to the XSLT stylesheets, but the important thing to remember is that they are also interpreted by ditac itself. By consequence, you cannot specify them in an XSLT stylesheet which customizes the stock ones (as explained in [Part II, Chapter 9, Section 2](#)).

Parameter	Value	Description
appendix-number-format	Allowed values are: 'I', 'i', 'A', 'a', '1'. Default value: 'A'.	The number format of topics referenced in a bookmap as <code>appendix</code> . By default, such topics are numbered as follows: Appendix A . <i>Title of first appendix</i> , Appendix B . <i>Title of second appendix</i> , etc.
cause-number-format	Allowed values are: 'I', 'i', 'A', 'a', '1'. Default value: 'A'.	In a <code><troubleshooting></code> topic, multiple <code><remedy></code> elements having no title are given numbers formatted using this format.
center	List of element names separated by whitespace. Example: 'fig equation-figure simpletable table'. Default value: ''.	Specifies which elements are to be centered horizontally on the page.
 ditacListsURI	URL ⁽⁴⁾ . Default value: <code>output_dir/ditac_lists.ditac_lists</code> .	The URL of file <code>ditac_lists.ditac_lists</code> .
equation-number-after	String. Default value: ') '.	Text added after the contents of a <code><equation-number></code> element.

⁽⁴⁾Unlike a filename, an URL must contain properly quoted characters. For example, do not specify 'Hello world.htm', instead specify 'Hello%20world.htm'.

Parameter	Value	Description
equation-number-before	String. Default value: ' ('.	Text added before the contents of a <equation-number> element.
extended-toc	Allowed values are: 'frontmatter', 'backmatter', 'both', 'none'. Default value: 'none'.	Allows to add <frontmatter> and <backmatter> <topicref>s to the Table of Contents (TOC) of a document. Note that the @toc, @navtitle, @locktitle, etc, attributes are applied normally to <frontmatter> and <backmatter> <topicref>s when an extended TOC is generated.
external-resource-base	Allowed values are: '', an URL ending with "/" or '#REMOVE'. Default value: '#REMOVE' for EPUB 2 and EPUB 3, '' for all the other output formats.	Specifies how to resolve <xref> or <link> elements having an <i>external</i> @scope attribute and a <i>relative</i> @href attribute. Example of such <xref> elements: <xref scope="external" format="java" href="src/Test.java">Test.java</xref>. '' Do not resolve the @href attribute. In this case, the external resource files are expected to be copied "by hand" to the output directory. An URL ending with "/" This URL is prepended to the value of the @href attribute. '#REMOVE' The <xref> or <link> element is processed as if it did not have an @href attribute.
highlight-source	Allowed values are: 'yes' and 'no'. Default value: 'yes'.	Allows to turn off syntax highlighting in elements specializing <pre>. By default, syntax highlighting is turned on for all elements specializing <pre> and having an @outputclass attribute equals to language-c, language-cpp, language-csharp, language-delphi, language-ini, language-java, language-javascript, language-m2, language-perl, language-php, language-python, language-ruby, language-tcl.
index-range-separator	String. Default value: '–' (EN DASH).	The string used to separate the first page number from the last page number in a page range of an indexed term. Example: index-range-separator='<-->': C Cat 54, 87<-->90


Parameter	Value	Description
link-auto-text	List of values separated by whitespace. Allowed values are: 'number' and 'text'. Default value: 'number text'.	This parameter specifies which text to generate for a <link> element, when this <link> element has no <linktext> child element or when this <linktext> child element is empty. Similar to above parameter xref-auto-text but for <link> elements.
note-icon-list	List of type attribute values separated by whitespace. Default value: 'attention caution danger fastpath important note notes remember restriction tip'.	Specifies the type (attribute @type) of the <note> elements for which icons should be used rather than text in order to represent note labels. Ignored unless use-note-icon='yes'.
number	List of values separated by whitespace. Allowed values are: 'topic', 'table', 'fig', 'equation-figure', 'all'. Default value: '' (number nothing).	Specifies which elements are to be numbered. 'all' is a short form for 'topic table fig equation-figure'. 'chapter-only' means: number topics, but only those referenced in a bookmark as <part>, <chapter> and <appendix>. <hr/>  Note Please note that 'all' does not include 'example'. If you want to number all formal elements including examples, then you must specify 'all example'. <hr/>
number-separator1	String. Default value: '.'.	The string used to separate the hierarchical number of topics acting as sections.
number-separator2	String. Default value: '-'.	The string used to separate the hierarchical number of figures, tables, examples and equations. When possible, the number of figure, table, example or equation is made relative to the number of the ancestor chapter or appendix. This gives for example (for descendants of chapter 5): Figure 5-1. Title of first figure of chapter 5, Figure 5-2. Title of second figure of chapter 5 , etc.
mark-important-steps	Allowed values are: 'yes' and 'no'. Default value: 'no'.	Generates a "Required" (respectively "Optional") label for <step> and <substep> elements having

Parameter	Value	Description
		an @importance attribute set to "required" (resp. "optional").
part-number-format	Allowed values are: 'I', 'i', 'A', 'a', '1'. Default value: 'I'.	The number format of topics referenced in a bookmap as part. By default, such topics are numbered as follows: Part I. Title of first part , Part II. Title of second part , etc.
prepend-chapter-to-section-number	Allowed values are: 'yes' and 'no'. Default value: 'no'.	Normally topics which are descendants of chapters (that is, topics referenced in a bookmap as <chapter>) are numbered as follows: 1. Title of first section , 1.1. Title of first subsection , etc. Specifying prepend-chapter-to-section-number='yes' prepends the number of the chapter ancestor to the section number. This gives for example (for descendants of chapter 5): 5.1. Title of first section , 5.1.1. Title of first subsection , etc.
remedy-number-format	Allowed values are: 'I', 'i', 'A', 'a', '1'. Default value: 'A'.	In a <troubleshooting> topic, multiple <remedy> elements having no title are given numbers formatted using this format.
show-draft-comments	Allowed values are: 'yes' and 'no'. Default value: 'no'.	Specifies whether <draft-comments> elements should be rendered.
troubleSolution-number-format	Allowed values are: 'I', 'i', 'A', 'a', '1'. Default value: '1'.	In a <troubleshooting> topic, multiple <troubleSolution> elements having no title are given numbers formatted using this format.
text-file-encoding	An encoding name such as UTF-8 or ISO-8859-1. Default value: ''.	Encoding of the text file referenced by coderef/@href. An empty string means: to be determined automatically.
title-after	List of element names separated by whitespace. Example: 'fig equation-figure table'. Default value: ''.	Specifies which elements should have their titles displayed after their bodies.
title-page	Allowed values are: 'auto', 'none' or the URI of a custom title page. Default value: 'auto'.	Specifies the kind of "title page" (contains the title of the document, its author, etc) to be generated before the actual contents of the document. 'auto' Automatically generate a title page based on the title and metadata of the map.

Parameter	Value	Description
		<p>'none'</p> <p>Do not generate a title page.</p> <p>URI of a custom title page</p> <p>Specifies the URI of a custom title page. If the URI is relative, it is relative to the current working directory of the user.</p> <p>This custom title page is an XHTML file for XHTML-based formats (XHTML, HTML Help, etc). This custom title page is an XSL-FO file for FO-based formats (PDF, RTF, etc). Such custom title pages are generally hand-written.</p> <ul style="list-style-type: none"> The child nodes of the <code>body</code> element of the custom XHTML title page are wrapped in a <code>div</code> contained in the XHTML/HTML file being generated by the XSLT stylesheet. <p>Do not add a <code><!DOCTYPE></code> to such custom XHTML title page because otherwise, the XSLT stylesheet may fail loading it.</p> <p>See sample custom XHTML title page.</p> <ul style="list-style-type: none"> The child nodes of the first <code>fo:flow[@flow-name='xsl-region-body']</code> element of the custom XSL-FO title page are wrapped in a <code>fo:block</code> contained the XSL-FO file being generated by the XSLT stylesheet. <p>See sample custom XSL-FO title page.</p>
<code>title-prefix-separator1</code>	String. Default value: ' . '.	The string used to separate the number of an formal object from its title.
<code>use-note-icon</code>	Allowed values are: 'yes' and 'no'. Default value: 'no'.	Specifies whether icons should be used rather than text in order to represent the label of a <code><note></code> element.
<code>watermark-image</code>	URI. If the URI is relative, it is relative to the current working directory of the user. No default value.	Specifies an image file which is to be used as a watermark in all the pages comprising the output document. See also parameter watermark . If you need this feature when generating RTF, WordprocessingML, Office Open XML (.docx),


Parameter	Value	Description
		OpenDocument (.odt), please make sure to use XMLmind XSL-FO Converter v5.3+ .
xref-auto-text	<p>List of values separated by whitespace. Allowed values are: 'number' and 'text'.</p> <p>Default value: 'number'.</p>	<p>This parameter specifies which text to generate for an <xref> element, when this <xref> element contains no text at all⁽⁵⁾.</p> <p>Let's suppose that an <xref> element containing no text at all points to a topic titled "Installation".</p> <p>Because the <xref> element points to an element having a <title> child element, ditac may use this title as a starting point for the generated text.</p> <p>Now let's suppose that topics are numbered and that the number of the "Installation" topic is "Chapter 5".</p> <p>The text generated for this <xref> element is thus:</p> <p>If xref-auto-text='number' Chapter 5</p> <p>If xref-auto-text='text' Installation</p> <p>If xref-auto-text='number text' Chapter 5. Installation</p> <p>Note that this specification is just a hint because ditac needs anyway to generate some text. For example, if topics are not numbered and xref-auto-text='number', the generated text will be "Installation".</p>
★xsl-re-sources-directory	<p>URL. A relative URL is relative to the output directory.</p> <p>Default value: 'resources/' resolved against the directory which contains the XSLT stylesheets.</p>	<p>Most XSLT stylesheets generate files which reference resources such as icons or CSS stylesheets. This parameter specifies the target directory which is to contain such resources.</p> <p>If this directory does not exist, it is automatically created.</p> <p>If this directory does not already contain the resources needed by the XSLT stylesheets, such resources are automatically copied to this directory.</p> <p>The default value of this parameter is something like <code>file:/opt/ditac/xsl/xhtml/resources/</code> for the stylesheets generating XHTML. URL <code>file:/opt/ditac/xsl/xhtml/resources/</code> specifies an existing directory containing <code>basic.css</code>, <code>note.png</code>, <code>important.png</code>, etc.</p>



⁽⁵⁾This implies that the xref-auto-text parameter is ignored when an <xref> element contains some text.


Parameter	Value	Description
		<p>This means that by default, no directory is created and no resource is copied.</p> <p>If the value of this parameter is an absolute URI, then ditac assumes that no resource directory is to be created and no resource is to be copied because this has already been done by the user.</p> <hr/> <p> Important</p> <ul style="list-style-type: none"> • Explicitly specifying something like <code>xsl-resources-directory='res'</code> is almost <i>always required</i> when generating files having an XHTML/HTML based format (XHTML, HTML Help, etc). • Explicitly specifying something like <code>xsl-resources-directory='res'</code> is almost <i>never required</i> when generating files converted from XSL-FO (PDF, RTF, etc).

Parameters common to the stylesheets that basically generate XHTML or HTML

This applies to the stylesheets that generate XHTML, HTML, Web Help, Java™ Help, HTML Help, Eclipse Help, EPUB.


Parameter	Value	Description
<code>add-index-toc</code>	<p>Allowed values are: 'yes' and 'no'.</p> <p>Default value: 'yes'.</p>	Specifies whether an A-Z list should be added at the beginning of the back-of-the-book index.
<code>chain-pages</code>	<p>Allowed values are: 'none', 'top', 'bottom' or 'both'.</p> <p>Default value: 'none'.</p>	<p>Specifies whether a header and/or a footer containing navigation icons should be generated in order to link together all the HTML pages.</p> <hr/> <p> Note</p> <p>There is no need to specify a value other than 'none' when generating Web Help,</p>

Parameter	Value	Description
		HTML Help, Eclipse Help, EPUB and Java™ Help.
chain-topics	<p>Allowed values are: 'yes' and 'no'.</p> <p>Default value: 'no'.</p>	<p>Specifies whether navigation icons should be generated in order to link together all the topics.</p> <p>See also related parameter: ignore-navigation-links.</p> <hr/> <p> Note</p> <p>There is no need to specify a value other than 'no' when generating Web Help, HTML Help, Eclipse Help, EPUB and Java™ Help.</p> <hr/>
css	<p>URL.</p> <p>Default value: ''.</p>	<p>Specifies which CSS stylesheet to use. Has priority over the CSS stylesheet specified by the <code>css-name</code> parameter.</p> <p>An end-user wishing to use a custom CSS must typically:</p> <ol style="list-style-type: none"> 1. Import the <code>basic.css</code> stock stylesheet in its own <code>custom.css</code> as follows: <pre>@import url(basic.css);</pre> 2. Invoke <code>ditac</code> with the following parameters: <code>xsl-resources-directory='res'</code>, <code>css='res/custom.css'</code>. 3. Copy by hand <code>custom.css</code> to <code>output_dir/res/</code> after <code>ditac</code> has finished its job. <hr/> <p> Restriction</p> <p>Not supported by the stylesheets that generate EPUB.</p> <hr/>
css-name	<p>URL basename relative to the directory specified by parameter <code>xsl-resources-directory</code>.</p> <p>Default value: 'basic.css'; 'java-help.css' when the output format is Java™ Help.</p>	<p>Specifies which CSS stylesheet to use. This CSS stylesheet is expected to be found in the resources directory.</p>


Parameter	Value	Description
		 Restriction Not supported by the stylesheets that generate EPUB.
default-table-width	A percentage, typically something like '100%' or '90%'. Default value: '' (as narrow as possible).	The default width of <table> and <simplatable> elements.
external-link-icon-height	Length. A length may have a unit. Default is px. Default value: '10'.	The height of the “opens in new window” icon.
external-link-icon-name	Basename. Default value: 'new_window.png'.	The basename of the “opens in new window” icon. This icon is found in the resources directory.
external-link-icon-width	Length. A length may have a unit. Default is px. Default value: '10'.	The width of the “opens in new window” icon.
format-to-type	Zero or more DITA format/MIME type pairs. Example: "txt text/plain xml application/xml html-text/html". Default value: '', which means that DITA xref/@format is <i>not</i> converted to XHTML a/@type.	Allows to map DITA xref/@format to XHTML a/@type. Using default empty value, <xref scope="external" format="txt" href="http://acme.com/info.xyz"> is converted to . The fact that file extension ".xyz" is unknown may cause problems when attempting to navigate or download file "info.xyz" using a Web browser. If -p format-to-type "txt text/plain" is passed to ditac then <xref scope="external" format="txt" href="http://acme.com/info.xyz"> is converted to , which is better.
generator-info	String Default value: 'XMLmind DITA Converter VERSION'.	The name of the software which has been used to create the HTML pages.

Parameter	Value	Description
		Specify an empty string if you don't want to have a <code><meta name="generator" content="XXX"/></code> element added to your HTML pages.
<code>ignore-navigation-links</code>	<p>Allowed values are: 'yes', 'no' and 'auto'.</p> <p>Default value: 'auto' for XHTML and its variants; 'yes' for Web Help, Java Help, HTML Help, Eclipse Help and EPUB</p>	<p>If 'yes', do not generate the navigation links corresponding to <code>topicref</code> attribute <code>@collection-type</code>.</p> <p>If 'no', generate the navigation links corresponding to <code>topicref</code> attribute <code>@collection-type</code>.</p> <p>If 'auto', generate the navigation links corresponding to <code>topicref</code> attribute <code>@collection-type</code>, unless <code>chain-topics=yes</code>.</p>
<code>javascripts</code>	<p>String. List of URLs separated by whitespace.</p> <p>Default value: ''.</p>	<p>The URLs specified in this parameter must point to JavaScript files. These URLs are converted to <code><script></code> XHTML elements added to the <code><html>/<head></code> elements of the XHTML files generated by ditac.</p> <p>Note that an URL may end with <code> ;async</code>, <code> ;defer</code> or a combination of both flags. These flags are translated to the corresponding attributes of the <code><script></code> element. Example:</p> <pre>https://cdn.mathjax.org/mathjax/latest/MathJax.js?config=MML_CHTML;async</pre> <p>is translated to:</p> <pre><script type="text/javascript" async="async" src="https://cdn.mathjax.org/mathjax/latest/MathJax.js?config=MML_CHTML"> </script></pre>
<code>mathjax</code>	<p>Allowed values are: 'yes', 'no' and 'auto'.</p> <p>Default value: 'no'.</p>	<p>Very few web browsers (Firefox) can natively render MathML. Fortunately, there is MathJax. MathJax is a JavaScript display engine for mathematics that works in all browsers.</p> <p>'yes'</p> <p>Add a <code><script></code> XHTML element loading MathJax to the <code><html>/<head></code> elements of all XHTML files generated by ditac.</p> <p>'auto'</p> <p>Same as 'yes', but add <code><script></code> only to generated XHTML files containing MathML.</p> <p>Ignored by all XHTML-based formats but XHTML and Web Help.</p>

Parameter	Value	Description
<code>mathjax-url</code>	String. Default value: the URL pointing to the MathJax CDN, as recommended in the MathJax documentation.	The URL allowing to load the MathJax engine configured for rendering MathML. Ignored unless parameter <code>mathjax</code> is set to 'yes' or 'auto'.
<code>mark-external-links</code>	Allowed values are: 'yes' and 'no'. Default value: 'no'.	Specifies whether an external link should be marked using a “opens in new window” icon.
<code>navigation-icon-height</code>	Length. A length may have a unit. Default is px. Default value: '16'.	The height of a navigation icon.
<code>navigation-icon-suffix</code>	String. Default value: '.png'.	The suffix of a navigation icon. The root names of navigation icons are fixed: <pre>first, first_disabled, last, last_disabled, next, next_disabled, previous, previous_disabled, parent, parent_disabled, child, child_disabled.</pre> For example, if <code>note-icon-suffix='.svg'</code> , the default resources directory is expected to contain <code>first.svg</code> , <code>first_disabled.svg</code> , <code>last.svg</code> , etc. In principle, there is no need for an end-user to specify any of the <code>navigation-icon-suffix</code> , <code>navigation-icon-width</code> or <code>navigation-icon-height</code> parameters.
<code>navigation-icon-width</code>	Length. A length may have a unit. Default is px. Default value: '16'.	The width of a navigation icon.
<code>screen-resolution</code>	Positive integer. Default value: '96'.	The resolution of the screen in dot per inch (DPI). This resolution is used to convert image dimensions such as 3cm to pixels.
<code>xhtml-mime-type</code>	A MIME type without a parameter such as 'text/html', 'application/xhtml+xml', 'application/xml' or the empty string ('').	Applies to all the XHTML-based formats (XHTML, EPUB), not to the HTML-based formats (Web Help, Java™ Help, HTML Help, Eclipse Help). By default ('text/html'), serve XHTML as HTML.

Parameter	Value	Description
	Default value: 'text/html'.	<p>Specify 'application/xhtml+xml' if you prefer to serve XHTML as XML.</p> <p>Specify an empty string if you prefer not to generate <code><meta http-equiv="Content-Type" ...></code>.</p> <hr/> <p> Tip</p> <p>Web browsers such as Firefox or Opera will <i>not</i> render the MathML embedded in XHTML, if this XHTML is served as HTML. Therefore when your DITA document contains MathML equations, you'll have to generate ".xhtml" files (".html" files won't work) and also, preferably, to specify <code>xhtml-mime-type="application/xhtml+xml"</code> or <code>xhtml-mime-type=""</code>.</p> <hr/>

Parameters common to the stylesheets that generate Web Help, Java™ Help, HTML Help, Eclipse Help and EPUB

Parameter	Value	Description
add-toc-root	<p>Allowed values are: 'yes' and 'no'.</p> <p>Default value: 'yes'.</p>	<p>If 'yes', add a pseudo TOC entry, bearing the title of the document, containing all the actual TOC entries.</p> <hr/> <p> Restriction</p> <ul style="list-style-type: none"> Value 'no' is not supported by the stylesheets that generate Eclipse Help. Ignored by the stylesheets that generate Web Help and EPUB. <hr/>
number-toc-entries	<p>Allowed values are: 'yes' and 'no'.</p> <p>Default value: 'yes' for Web Help, 'no' for the other formats.</p>	<p>If 'yes', number the TOC entries. No effect unless the <code>number</code> parameter is used to specify that topics should be numbered.</p>

Parameters specific to the stylesheets that generate Web Help

Parameter	Value	Description
★wh-collapse-toc	Allowed values are: 'yes' and 'no'. Default value: 'no'.	Specifies whether the TOC should be initially collapsed.
★wh-index-numbers	Allowed values are: 'yes' and 'no'. Default value: 'no'.	Specifies whether words looking like numbers are to be indexed. Examples of such number-like words: 3.14, 3, 14, 3times4equals12, +1, -1.0, 3px, 1, 2cm, 100%, 1.0E+6, 1,000.00\$.
★wh-jquery	Relative or absolute URI. A relative URI is relative to the URI of a page of the Web Help. Default value: absolute URI of the corresponding file found on the Google CDN.	Specifies the location of the JavaScript file containing jQuery . Example: <pre>http://ajax.aspnetcdn.com/ajax/\njQuery/jquery-3.2.1.min.js</pre> Specifying an "https:" URL is recommended when the generated Web Help is stored on an HTTPS server.
★wh-jquery-css	Relative or absolute URI. A relative URI is relative to the URI of a page of the Web Help. Default value: absolute URI of the corresponding file found on the Google CDN.	Specifies the location of the CSS stylesheet of jQuery UI . Example: <pre>http://ajax.aspnetcdn.com/ajax/\njquery.ui/1.12.1/themes/redmond/\njquery-ui.css</pre> Specifying an "https:" URL is recommended when the generated Web Help is stored on an HTTPS server.
★wh-jquery-custom-theme	Filename or absolute URI of a .zip file created using jQueryUI ThemeRoller . A relative filename is relative to the current working directory. No default.	Specifies a .zip file created using jQueryUI ThemeRoller containing a custom jQueryUI theme. Example: jquery-ui-1.12.1.custom.zip. The files comprising the custom theme are copied to <code>_wh/jquery/</code> , where <code>_wh/</code> is the directory containing the other Web Help files. Ignored if parameter <code>wh-jquery-css</code> has been used to specify the CSS stylesheet of jQuery UI .
★wh-jquery-theme	The name of a theme. Examples: 'redmond', 'cupertino'. Default value: 'smoothness'.	Specifies the name of the jQuery UI theme used by the compiler. Ignored if parameter <code>wh-jquery-css</code> or <code>jquery-custom-theme</code> have been used to specify the CSS stylesheet of jQuery UI .
★wh-jquery-ui	Relative or absolute URI. A relative URI is relative to the URI of a page of the Web Help.	Specifies the location of the JavaScript file containing jQuery UI . Example: <pre>http://ajax.aspnetcdn.com/ajax/\n</pre>

Parameter	Value	Description
	Default value: absolute URI of the corresponding file found on the Google CDN.	<pre>jquery.ui/1.12.1/jquery-ui.min.js'</pre> <p>Specifying an "https:" URL is recommended when the generated Web Help is stored on an HTTPS server.</p>
★wh-local-jquery	yes or no. Default value: no.	<p>Specifies whether all jQuery files should be copied to <code>_wh/jquery/</code>, where <code>_wh/</code> is the directory containing the other Web Help files.</p> <p>By default, the jQuery files are accessed from the Web (typically from a CDN).</p> <p>Note that this parameter is applied <i>after</i> JQuery and JQueryUI have been possibly customized using parameters <code>jquery</code>, <code>jquery-ui</code>, <code>jquery-css</code>, <code>jquery-theme</code> and <code>jquery-custom-theme</code>. For example, "<code>-p jquery-theme-cupertino -p local-jquery yes</code>" copies the Cupertino CSS files to <code>_wh/jquery/</code>.</p>
★wh-layout	The name of a layout. Default value: classic.	<p>Selects a layout for the generated Web Help.</p> <p>For now, only 2 layouts are supported: <code>classic</code> and <code>simple</code>.</p>
★wh-use-stemming	yes or no. Default value: yes.	<p>Specifies whether stemming should be used to implement the search facility. By default, stemming is used whenever possible, that is,</p> <ul style="list-style-type: none"> • when the main language of the document can be determined; • when this main language is one of: Danish, Dutch, English, Finnish, French, German, Hungarian, Italian, Norwegian, Portuguese, Russian, Spanish, Swedish, Romanian, Turkish. <p>The main language of the document is specified by the <code>@xml:lang</code> attribute found on the root element of DITA map being converted; otherwise using the -lang command-line option; otherwise, it is assumed to be "en".</p>
★wh-user-css	Filename or absolute URI of a CSS file. A relative filename is relative to the current working directory.	<p>Specifies the user's CSS stylesheet which is to be added to each page of the Web Help.</p> <p>This file is copied to <code>output_directory/_wh/user/</code>.</p> <p>Sample user's CSS <code>wh_resources/header_footer.css</code> as used in the following example:</p> <pre>-p wh-user-header - wh_resources/header.html -p wh-user-footer -</pre>

Parameter	Value	Description
		<pre>wh_resources/footer.html -p wh-user-css ↵ wh_resources/header_footer.css -p wh-user-resources ↵ wh_resources/header_footer_files</pre>
<p>★wh-user-footer</p>	<p>Filename or absolute URI of an XHTML file. A relative filename is relative to the current working directory.</p>	<p>Specifies the user's footer which is to be added to each page of the Web Help.</p> <p>The content of the <body> element of wh-user-footer is inserted as is in the <div id="wh-footer"> found in each page of the Web Help.</p> <p>Same remark as for parameter wh-user-header about the resources referenced by a user's footer.</p> <p>Sample user's footer wh_resources/footer.html as used in the following example:</p> <pre>-p wh-user-header ↵ wh_resources/header.html -p wh-user-footer ↵ wh_resources/footer.html -p wh-user-css ↵ wh_resources/header_footer.css -p wh-user-resources ↵ wh_resources/header_footer_files</pre>
<p>★wh-user-header</p>	<p>Filename or absolute URI of an XHTML file. A relative filename is relative to the current working directory.</p>	<p>Specifies the user's header which is to be added to each page of the Web Help.</p> <p>The content of the <body> element of wh-user-header is inserted as is in the <div id="wh-header"> found in each page of the Web Help.</p> <p>If a user's header references resources (e.g. image files), then these resources must either be referenced using absolute URLs or these resources must be found in a user's resource directory and parameter wh-user-resources must be specified.</p> <p>Example:</p> <ul style="list-style-type: none"> • The user's resource directory is called header_footer_files/ and contains header_footer_files/logo100x50.png. • ditac is passed parameters: -p user-resources PATH_TO/header_footer_files and -p user-header-PATH_TO/header.html. • header.html looks like this: <pre><html> ...</pre>

Parameter	Value	Description
		<pre><body> </body> </html></pre> <p>Notice the path used to reference logo100x50.png.</p> <p>Sample user's header wh_resources/header.html as used in the following example:</p> <pre>-p wh-user-header ↵ wh_resources/header.html -p wh-user-footer ↵ wh_resources/footer.html -p wh-user-css ↵ wh_resources/header_footer.css -p wh-user-resources ↵ wh_resources/header_footer_files</pre>
★wh-user-resources	Filename or absolute "file:" URI of a <i>directory</i> . URI schemes other than "file" (e.g. "http") are not supported for this parameter. A relative filename is relative to the current working directory.	<p>Specifies a user's resource directory which is to be recursively copied to <i>output_directory/_wh/user/</i>.</p> <p>This directory typically contains image files referenced by the user's header, footer or CSS stylesheet.</p> <p>Sample user's resource directory wh_resources/header_footer_files/ as used in the following example:</p> <pre>-p wh-user-header ↵ wh_resources/header.html -p wh-user-footer ↵ wh_resources/footer.html -p wh-user-css ↵ wh_resources/header_footer.css -p wh-user-resources ↵ wh_resources/header_footer_files</pre>
whc-index-basename	URL basename. Default value: 'whc_index.xml'.	<p>Basename of the Index XML input file of XMLmind Web Help Compiler.</p> <p>In principle, there is no need for an end-user to specify this parameter.</p>
whc-toc-basename	URL basename. Default value: 'whc_toc.xml'.	<p>Basename of the TOC XML input file of XMLmind Web Help Compiler.</p>

Parameter	Value	Description
		In principle, there is no need for an end-user to specify this parameter.



Parameters specific to the stylesheets that generate Java™ Help

In principle, there is no need for an end-user to specify any of the following parameters.

Parameter	Value	Description
helpset-basename	URL basename. Default value: 'jhelpset.hs'.	Basename of the Java™ Help HelpSet file.
index-basename	URL basename. Default value: 'jhelpidx.xml'.	Basename of the Java™ Help Index file.
indexer-directory-basename	URL basename. Default value: 'Java-HelpSearch'.	Basename of the directory which will contain the data generated by running <code>jhindexer</code> . A properly quoted relative URL, not a filename.
map-basename	URL basename. Default value: 'jhelpmap.jhm'.	Basename of the Java™ Help Map file.
toc-basename	URL basename. Default value: 'jhelp-toc.xml'.	Basename of the Java™ Help Contents file.


Parameters specific to the stylesheets that generate HTML Help

In principle, there is no need for an end-user to specify any of the following parameters.

Parameter	Value	Description
 chmBasename	URL basename. Default value: 'help.chm'.	Basename of the compiled HTML Help file.
hnc-basename	URL basename. Default value: 'toc.hnc'.	Basename of the HTML Help contents file.
hnp-template	URL basename. Default value: 'template.hnp' resolved against the directory which contains the XSLT stylesheets.	URL of the file containing the template of the HTML Help project file. This plain text file encoded in UTF-8 contains variables such as <code>%compiledFile%</code> , <code>%contentsFile%</code> , <code>%default-Topic%</code> , etc, which are substituted with their values.
 hnpBasename	URL basename.	Basename of the HTML Help project file.

Parameter	Value	Description
	Default value: 'project.hhp'.	
hhx-basename	URL basename. Default value: 'index.hhx'.	Basename of the HTML Help index file.

Parameters specific to the stylesheets that generate Eclipse Help

Parameter	Value	Description
plugin-id	String No default value.	An ID uniquely identifying the plug-in, typically a Java-like fully qualified name. Example: 'com.acme.widget.userguide'. <div style="border: 1px solid #ccc; padding: 10px; margin: 10px 0;"> <p> Important</p> <p>The subdirectory of <code>plugins/</code> containing the plug-in must have the same base-name as the value of parameter <code>plugin-id</code>.</p> </div>
plugin-index-basename	URL basename. Default value: 'index.xml'.	Basename of the index file.
plugin-name	String No default value.	The name of the plug-in, typically the title of the document. Example: 'ACME Widget User's Guide'.
plugin-provider	String No default value.	The author, company or organization which has contributed the plug-in. Example: 'ACME Corp.'.
plugin-toc-basename	URL basename. Default value: 'toc.xml'.	Basename of the table of contents file.
plugin-version	String Default value: '1.0.0'.	The version of the plug-in.

Parameters specific to the stylesheets that generate EPUB


Parameter	Value	Description
cover-image	URI. If the URI is relative, it is relative to the	Specifies an image file which is to be used as the cover page of the EPUB file. This image must be

Parameter	Value	Description
	current working directory of the user. No default value.	a PNG or JPEG image. Its size must not exceed 1000x1000 pixels. In theory, EPUB 3 also accepts SVG 1.1 cover images.
epub-identifier	String Default value: dynamically generated UUID URN.	A globally unique identifier for the generated EPUB document (typically the permanent URL of the EPUB document).
epub2-compatible	Allowed values are: 'yes' and 'no'. Default value: 'yes'.	Only applies to EPUB 3. By default, the EPUB 3 files generated by ditac are made compatible with EPUB 2 readers. Specify 'no' if you don't need this compatibility.
generate-epub-trigger	Allowed values are: 'yes' and 'no'. Default value: 'yes'.	Only applies to EPUB 3. Specify 'no' if your EPUB 3 reader does not support <code>epub:trigger</code> yet. When <code>generate-epub-trigger=no</code> , ditac generates an <code>@onclick</code> attribute containing simple JavaScript code and declares the containing XHTML 5 page as being scripted.

Parameters specific to the stylesheets that generate XSL-FO

The XSL-FO file generated by the XSLT stylesheets is converted to PDF, PostScript®, RTF, WordprocessingML, Office Open XML (.docx), OpenOffice/LibreOffice (.odt) by the means of XSL-FO processors such as Apache FOP, RenderX XEP, Antenna House XSL Formatter or XMLmind XSL-FO Converter.

Parameter	Value	Description
base-font-size	Default value: '10pt'.	The size of the "main font" of the document. All the other font sizes are computed relatively to this font size
body-bottom-margin	Length. Default value: '0.5in'.	See Figure 4-1 below.
body-font-family	A string containing one or more font families separated by commas. Default value: 'serif'.	Specifies the family of the font used for the text of all elements except topic titles.
body-start-indent	Length. Default value: '2pc'.	Applies only to <i>alternate XSLT stylesheet</i> <code>ditac_install_dir/xsl/fo/fo_indent.xsl</code> . This stylesheet:

Parameter	Value	Description
		<ul style="list-style-type: none"> Indents all blocks but topic and section titles by the value of XSLT stylesheet parameter <i>body-start-indent</i>. By default <i>body-start-indent</i> is 2pc. Adds more vertical space after topic and section titles. Only part, appendices, chapter and appendix titles are underlined. <p>This stylesheet is invoked by passing option <code>-t ditac-xsl:fo/fo_indent.xsl</code> to ditac. Example of its output: <code>manual-fop.pdf</code>.</p>
<code>body-top-margin</code>	<p>Length.</p> <p>Default value: <code>'0.5in'</code>.</p>	See Figure 4-1 below.
<code>choice-bullets</code>	<p>A string containing one or more single characters separated by whitespace.</p> <p>Default value: <code>'&#x2022;'</code> (BULLET).</p>	<p>Specify which bullet character to use for a <code><choice></code> element. Additional characters are used for nested <code><choice></code> elements.</p> <p>Changing the value of this parameter may imply changing the <code>font-family</code> attribute of the attribute-set <code>choice-label</code>.</p>
<code>equation-block-equation-width</code>	<p>Length.</p> <p>Default value: <code>'90%'</code>.</p>	In a numbered <code><equation-block></code> element, this parameter specifies the width of the column containing the equation.
<code>equation-block-number-width</code>	<p>Length.</p> <p>Default value: <code>'10%'</code>.</p>	In a numbered <code><equation-block></code> element, this parameter specifies the width of the column containing the <code><equation-number></code> element.
<code>external-href-after</code>	<p>String.</p> <p>Default value: <code>''</code>.</p>	Appended after the external URL referenced by an <code><xref></code> or <code><link></code> element. Ignored unless <code>show-external-links='yes'</code> .
<code>external-href-before</code>	<p>String.</p> <p>Default value: <code>' '</code>.</p>	Separates the text of an <code><xref></code> or <code><link></code> element from its referenced external URL. Ignored unless <code>show-external-links='yes'</code> .
 <code>foProcessor</code>	<p>String. Examples: <code>'FOP'</code>, <code>'XEP'</code>, <code>'AHF'</code>, <code>'XFC'</code>.</p> <p>Default value: <code>''</code>.</p>	The name of the XSL-FO processor used to convert the XSL-FO file generated by the XSLT stylesheets to the target output format.
<code>footer-center</code>	<p>String.</p>	<p>Specifies the contents of the central part of a page footer. See Specifying a header or a footer.</p> <p>Supports a conditional specification.</p> <p>Default value:</p> <pre>two-sides even:: {{chapter-title}};; two-sides part chapter appendices appendix odd::-</pre>

Parameter	Value	Description
		<pre> {{section1-title}};; one-side:: {{chapter-title}} </pre>
footer-center-width	String representing an integer larger than or equal to 1. Default value: '6'.	Specifies the proportional width of the central part of a page footer. See Specifying a header or a footer . Supports a conditional specification .
footer-height	Length. Default value: '0.4in'.	See Figure 4-1 below.
footer-left	String. Default value:	Specifies the contents of the left part of a page footer. See Specifying a header or a footer . Supports a conditional specification . Default value: <pre> two-sides even:: {{page-number}} </pre>
footer-left-width	String representing an integer larger than or equal to 1. Default value: '2'.	Specifies the proportional width of the left part of a page footer. See Specifying a header or a footer . Supports a conditional specification .
footer-right	String. Default value:	Specifies the contents of the right part of a page footer. See Specifying a header or a footer . Supports a conditional specification . Default value: <pre> two-sides first odd:: {{page-number}};; one-side:: {{page-number}} </pre>
footer-right-width	String representing an integer larger than or equal to 1. Default value: '2'.	Specifies the proportional width of the right part of a page footer. See Specifying a header or a footer . Supports a conditional specification .
footer-separator	Allowed values are: 'yes' and 'no'. Default value: 'yes'.	Specifies whether an horizontal rule should be drawn above the page footer.
header-center	String. Default value: '{{document-title}}'.	Specifies the contents of the central part of a page header. See Specifying a header or a footer . Supports a conditional specification .

Parameter	Value	Description
header-center-width	String representing an integer larger than or equal to 1. Default value: '6'.	Specifies the proportional width of the central part of a page header. See Specifying a header or a footer . Supports a conditional specification.
header-height	Length. Default value: '0.4in'.	See Figure 4-1 below.
header-left	String. Default value: ''.	Specifies the contents of the left part of a page header. See Specifying a header or a footer . Supports a conditional specification.
header-left-width	String representing an integer larger than or equal to 1. Default value: '2'.	Specifies the proportional width of the left part of a page header. See Specifying a header or a footer . Supports a conditional specification.
header-right	String. Default value: ''.	Specifies the contents of the right part of a page header. See Specifying a header or a footer . Supports a conditional specification.
header-right-width	String representing an integer larger than or equal to 1. Default value: '2'.	Specifies the proportional width of the right part of a page header. See Specifying a header or a footer . Supports a conditional specification.
hyphenate	Allowed values are: 'yes' and 'no'. Default value: 'no'.	Specifies whether words may be hyphenated.
index-column-count	Positive integer. Default value: '2'.	The number of columns of index pages.
index-column-gap	Length. Default value: '2em'.	The distance which separates columns in index pages.
justified	Allowed values are: 'yes' and 'no'. Default value: 'no'.	Specifies whether text (e.g. in paragraphs) should be justified (that is, flush left and right) or just left aligned (that is, flush left and ragged right).
link-bullet	A string containing a single character.	Specify which character is inserted before the text of a <link> element.

Parameter	Value	Description
	Default value: '•' (BULLET).	Changing the value of this parameter may imply changing the <code>font-family</code> attribute of the attribute-set <code>link-bullet</code> .
<code>menucas-cade-separator</code>	A string containing a single character. Default value: '→' (RIGHTWARDS ARROW).	Specify which character is used to separate the child elements of a <code><menucascade></code> element. Changing the value of this parameter may imply changing the <code>font-family</code> attribute of the attribute-set <code>menucas-cade-separator</code> .
<code>note-icon-height</code>	Length. A length may have a unit. Default is px. Default value: '32'. '7mm' for the XSLT stylesheets that generate XSL-FO.	The height of a note icon.
<code>note-icon-suffix</code>	Default value: ' .png'.	The suffix of a note icon. The root name of a note icon should be identical to the value of the <code>@type</code> attribute it represents. For example, if <code>note-icon-suffix=' .svg'</code> , the default resources directory is expected to contain <code>note.svg</code> , <code>important.svg</code> , <code>caution.svg</code> , etc. In principle, there is no need for an end-user to specify any of the <code>note-icon-suffix</code> , <code>note-icon-width</code> or <code>note-icon-height</code> parameters.
<code>note-icon-width</code>	Length. A length may have a unit. Default is px. Default value: '32'. '7mm' for the XSLT stylesheets that generate XSL-FO.	The width of a note icon.
<code>page-bottom-margin</code>	Length. Default value: '0.5in'.	See Figure 4-1 below.
<code>page-height</code>	Length. Example: '297mm'. Default value: depends on paper-type.	The height of the printed page.

Parameter	Value	Description
page-inner-margin	Length. Default value: if parameter two-sided is specified as 'yes' then '1.25in' otherwise '1in'.	See Figure 4-1 below.
page-orientation	Allowed values are: 'portrait' and 'landscape'. Default value: 'portrait'.	The orientation of the printed page.
page-outer-margin	Length. Default value: if parameter two-sided is specified as 'yes' then '0.75in' otherwise '1in'.	See Figure 4-1 below.
page-ref-after	String. Default value: ''.	Appended after the page number pointed to by an <xref> or <link> element. Ignored unless show-xref-page='yes' or show-link-page='yes'. When both page-ref-after and page-ref-before are specified as the empty string, in fact, this specifies that the generated string must be the localized equivalent of "on page".
page-ref-before	String. Default value: ''.	Separates the text of an <xref> or <link> element from the page number it points to. Ignored unless show-xref-page='yes' or show-link-page='yes'.
page-top-margin	Length. Default value: '0.5in'.	See Figure 4-1 below.
page-width	Length. Example: '8.5in'. Default value: depends on paper-type.	The width of the printed page.
paper-type	Allowed values are: 'Letter', 'Legal', 'Ledger', 'Tabloid', 'A0',	A convenient way to specify the size of the printed page. It is also possible to specify a custom paper type by ignoring the paper-type parameter and directly specifying the page-width and page-height parameters.

Parameter	Value	Description
	'A1', 'A2', 'A3', 'A4', 'A5', 'A6', 'A7', 'A8', 'A9', 'A10', 'B0', 'B1', 'B2', 'B3', 'B4', 'B5', 'B6', 'B7', 'B8', 'B9', 'B10', 'C0', 'C1', 'C2', 'C3', 'C4', 'C5', 'C6', 'C7', 'C8', 'C9', 'C10' (case-insensitive). Default value: 'A4'.	
pdf-outline	Allowed values are: 'yes' and 'no'. Default value: 'no'.	Specifies whether PDF bookmarks should be generated. Supported by the 'XEP', 'FOP' and 'AHF' XSL-FO processors. Not relevant, and thus ignored by 'XFC'.
show-external-links	Allowed values are: 'yes' and 'no'. Default value: 'no'.	Specifies whether the <i>external URL</i> referenced by an <xref> or <link> element should be displayed right after the text contained by this element. Example: show-external-links='yes' causes <xref href="http://www.oasis-open.org/">Oasis</xref> to be rendered as follows: Oasis [http://www.oasis-open.org/].
show-imagemap-links	Allowed values are: 'yes' and 'no'. Default value: 'yes'.	Specifies whether a numbered list should be generated for an <imagemap> element, with one list item per <area> element. A list item contains the link specified by the <area> element. No list items are generated for “dead areas” (<area> elements specifying no link at all).
show-link-page	Allowed values are: 'yes' and 'no'. Default value: 'no'.	Same as show-xref-page but for <link> elements.
show-xref-page	Allowed values are: 'yes' and 'no'. Default value: 'no'.	Specifies whether the page number corresponding to the <i>internal link target</i> referenced by an <xref> element should be displayed right after the text contained by this element. Example: show-xref-page='yes' causes <xref href="introduction.dita">Introduction</xref> to be rendered as follows: Introduction [3].

Parameter	Value	Description
title-color	A string representing a color. Default value: 'black'.	Specifies the color used for the text of topic (of any kind) titles.
title-font-family	A string containing one or more font families separated by commas. Default value: 'sans-serif'.	Specifies the family of the font used for the text of topic (of any kind) titles.
two-sided	Allowed values are: 'yes' and 'no'. Default value: 'no'.	Specifies whether the document should be printed double sided.
ul-li-bullets	A string containing one or more single characters separated by whitespace. Default value: '• –' (BULLET, EN DASH).	Specify which bullet character to use for an / element. Additional characters are used for nested elements. For example, if ul-li-bullets="* - +", "*" will be used for / elements, "-" will be used for / elements contained in a / element and "+" will be used for / elements nested in two / elements. Changing the value of this parameter may imply changing the font-family attribute of the attribute-set ul-li-label.
un-ordered-step-bullets	A string containing one or more single characters separated by whitespace. Default value: '•' (BULLET, EN DASH).	Specify which bullet character to use for a <steps-unordered>/<step> element. Additional characters are used for nested <steps-unordered>/<step> elements. Changing the value of this parameter may imply changing the font-family attribute of the attribute-set un-ordered-step-label.
use-multimedia-extensions	'yes', 'no' or a list of MIME types separated by white-space. Default value: 'no'.	Currently ignored unless RenderX XEP is used to generate PDF. Specifies whether vendor-specific XSL-FO extensions should be used to embed rich media content in the output file. If specified value is 'yes', then XSL-FO extensions are used whatever the MIME types of the media. If specified value is a list of MIME types, then XSL-FO extensions are used only for media having these types. For now, it seems that only 'application/x-shockwave-flash' gives useful results.


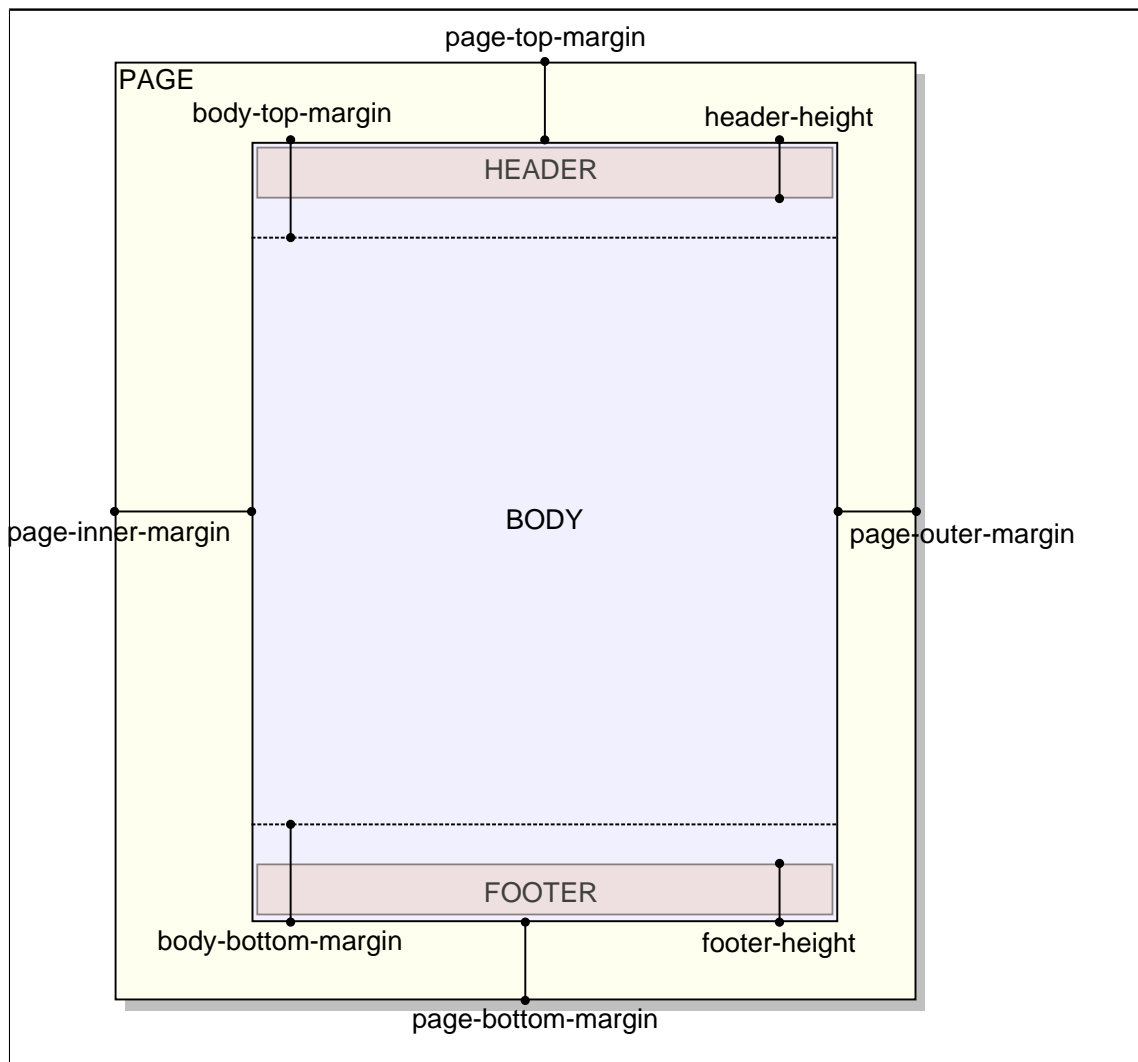
Parameter	Value	Description
		 <p>Note</p> <p>This feature requires generating PDF version 1.5+. By default, XEP generates PDF version 1.4. It seems that there is no way to change this by passing a command-line option to <code>xep.bat</code> or to the <code>xep</code> shell script. However, this can be changed once for all, for example by inserting <code><option name="PDF_VERSION" value="1.5"/></code> into the <code><generator-options format="PDF"></code> element found in the <code>XEP_install_dir/xep.xml</code> configuration file.</p>
watermark	<p>Allowed values are one or more of 'blank', 'title', 'toc', 'booklist', 'frontmatter', 'body', 'backmatter', 'index', 'all' separated by white-space.</p> <p>Default value: 'all'.</p>	<p>Specifies which pages in the output document are to be given a watermark.</p> <p>By default, all pages are given a watermark. If for example, parameter <code>watermark</code> is set to <code>'frontmatter body backmatter'</code>, then only the pages which are part of the front matter, body and back matter of the output document are given a watermark. The title page, TOC pages, etc, are not given a watermark.</p> <p>No effect unless parameter <code>watermark-image</code> is specified.</p>

Figure 4-1. Page areas

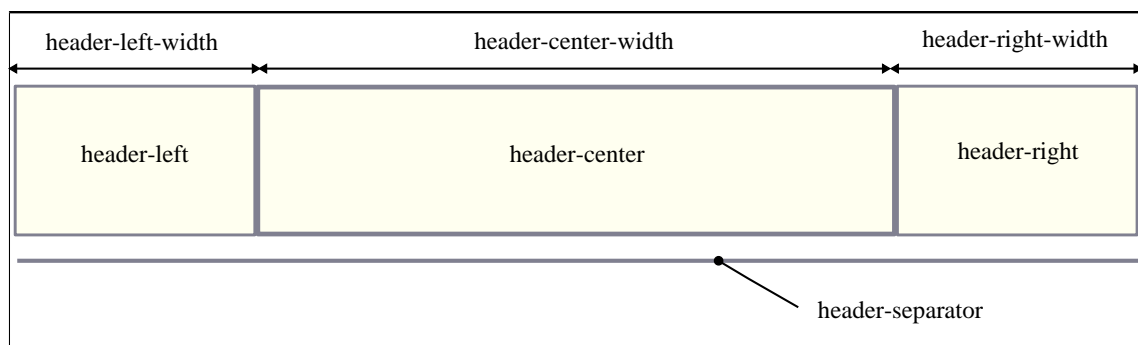


1. Page headers and footers

Specifying a header or a footer

The header or the footer of a generated PDF, RTF, etc, page has 3 columns.

Figure 4-2. Layout of a header



The width of these columns may be specified using the `header-left-width`, `header-center-width`, `header-right-width` parameters for the header and the `footer-left-width`, `footer-center-width`, `footer-right-width` parameters for the footer.

The width of a column is specified as an integer which is larger than or equal to 1. This value is the *proportional width of the column*. For example, if the left column has a width equal to 2 and the right column has a width equal to 4, this simply means that the right column is twice ($4/2 = 2$) as wide as the left column.

The contents of these columns may be specified using the `header-left`, `header-center`, `header-right` parameters for the header and the `footer-left`, `footer-center`, `footer-right` parameters for the footer.

When `header-left`, `header-center`, `header-right` are all specified as the empty string, no header is generated. When `footer-left`, `footer-center`, `footer-right` are all specified as the empty string, no footer is generated.

The content of a column is basically a mix of text and variables. Example: "Page `{{page-number}}` of `{{page-count}}`".

Supported variables are:

`{{document-title}}`

The title of the document.

`{{document-date}}`

The publication date of the document.

The value of the variable comes from the last `created` or `revised` element found in the `topicmeta/critdates` or `bookmeta/critdates` element of the map. More precisely, it comes the value of attribute `golive`, `modified` or `date`, considered in that order. The value of this attribute must be something like *YYYY-MM-DD*, because it is parsed and then formatted according to the `xml:lang` of the map. For example, if `golive="2014-02-23"`, with `xml:lang="en"`, it gives: "February 02, 2014" and with `xml:lang="fr"`, it gives: "02 Février 2014". If the map has no `critdates` element, then the current date is used. If the value of attribute `golive`, `modified` or `date` is not specified as *YYYY-MM-DD*, then this value is used as is.

`{{chapter-title}}`

The title of the current part, chapter, appendices or appendix. Empty if the map being converted is not a bookmap.

`{{section1-title}}`

The title of the current part, chapter, appendices or appendix or *section 1*. A section 1 is specified by a non-typed `topicref` (that is, not a part, chapter, preface, appendix, dedication, etc) which is a direct child of a map or bookmap.

`{{topic-title}}`

The title of the current topic. All topics are guaranteed to have a corresponding `{{topic-title}}`. Even automatically generated topics such as `toc` or `indexlist` have a corresponding `{{topic-title}}`⁽⁶⁾.

`{{page-number}}`

Current page number within the current document division (front matter, body matter or back matter).

`{{page-count}}`

Total number of pages of the current document division (front matter, body matter or back matter).

⁽⁶⁾The `{{topic-title}}` of a `toc` is "**Table of Contents**", properly localized. The `{{topic-title}}` of a `indexlist` is "**Index**", properly localized.

`{{break}}`

A line break.

`{{image(URI)}}`

An image having specified URI. A relative URI is resolved against the current working directory.

Example: "`{{image(artwork/logo.svg)}}`".

`{{page-sequence}}`

Not for production use. Inserts in the header/footer the name of the current page sequence . This allows to learn which name to use in a *conditional header or footer*. See [below](#).

Conditional headers and footers

The default value of `header-center` is '`{{document-title}}`'. This means that each page of the generated PDF, RTF, etc, file will have the document title centered on its top. But what if you want the pages containing the Table of Contents have a "**Contents**" header? Is there a way to specify: use "**Contents**" for the pages containing the Table of Contents and use the title of the document for any other page?

This is done by specifying the following conditional value for parameter `header-center`: '`toc:: Contents; {{document-title}}`'.

A conditional value may contain one or more cases separated by `;;`. Each case is tested against the page being generated. The first case which matches the page being generated is the one which is selected.

*conditional_value --> case [";;" case]**

case --> [condition ":"] value*

*condition --> [test_page_sequence]?
& [S test_page_layout]?
& [S test_page_side]?*

Let's suppose you also want the the pages containing the Index have a "**Index**" header. Specifying '`toc:: Contents; {{document-title}}; indexlist:: Index`' won't work as expected because the second case (having no condition at all) matches any page, including the Index pages. You need to specify: '`toc:: Contents; indexlist:: Index; {{document-title}}`'.

Let's remember that variable `{{topic-title}}` is substituted with the title of the current topic, including automatically generated topics such `toc` and `indexlist`.

Therefore our conditional value is better expressed as: '`toc:: indexlist:: {{topic-title}}; {{document-title}}`'. Notice how a case may have several conditions. Suffice for any of these conditions to match the page being generated for the case to be selected.

Even better, specify '`toc|indexlist:: {{topic-title}}; {{document-title}}`'. String `"|"` may be used to separate alternative values to be tested against the page being generated.

*test_page_sequence --> page_sequence ["|" page_sequence]**

*page_sequence --> "abbrevlist" | "amendments" | "appendices" | "appendix"
| "backmattersection" | "bibliolist" | "bookabstract" | "booklist"
| "chapter" | "colophon" | "dedication" | "draftintro"
| "figurelist" | "glossarylist" | "indexlist" | "notices"
| "part" | "preface" | "section1" | "tablelist"
| "toc" | "trademarklist"*

**Tip**

It's not difficult to guess that the name of the page sequence corresponding to the Table of Contents is `toc` and that the name of the page sequence corresponding to the Index is `indexlist`. However the simplest way to learn what is the name of the page sequence being generated is to reference variable `{{page-sequence}}` in the specification of a header or a footer.

Now let's suppose that we want to suppress the document title on the first page of a part, chapter or appendix. This is specified as follows: `'first part|chapter|appendix:: ;; toc|indexlist:: {{topic-title}};; {{document-title}}'`.

For now, we have only described a condition about the page sequence being generated: TOC, Index, etc. In fact, a condition may test up to 3 facets of the page being generated:

- The page sequence to which belongs the page being generated.
- Whether the page being generated is part of a one-sided or a two-sided document.
- Whether the page being generated is the first page of its sequence. When the the page being generated is *not* the first page of its sequence, if the page being generated has an odd or an even page number.

```
test_page_layout --> page_layout [ "|" page_layout ]*
```

```
page_layout --> "two-sides" | "one-side"
```

```
test_page_side --> page_side [ "|" page_side ]*
```

```
page_side --> "first" | "odd" | "even"
```

**Remember**

When the document has one side, the only possible page side is `odd`. The other values, `first` and `even`, are not supported. For example, something like `'one-side chapter|appendix even:: {{chapter-title}};;'` cannot generate any text.

The order of the tests is not significant. For example, `'first part|chapter|appendix'` is equivalent to `'part|chapter|appendix first'`.

Therefore `'first part|chapter|appendix:: ;; toc|indexlist:: {{topic-title}};; {{document-title}}'` reads as follows:

1. Use the empty string for the first page of a part, chapter or appendix.
2. Use the topic title for the pages containing the Table of Contents. This title is "**Table of Contents**", but localized according to the main language of the DITA document being converted.
3. Use the topic title for the pages containing the Index. This title is "**Index**", but localized according to the main language of the DITA document being converted.
4. For any other page, use the title of the DITA document.

**Note**

Everything explained in this section applies not only to the contents of a column of a header or footer, but also to the proportional width of a column of a header or footer. Example: -
`p footer-right-width "first||odd:: 4;; even:: 1".`

Chapter 5. Controlling the numbering of ordered lists

This chapter explains how you can control the numbering of ordered lists by the means of one or more directives specified in the @outputclass attribute of the element.

By default, the numbering of nested ordered lists automatically alternates between the "1 ." and "a ." formats. If you want more control on the numbering of ordered lists, then you'll have to specify one or more of the following directives in the @outputclass attribute of the element.

lower-alpha

upper-alpha

lower-roman

upper-roman

decimal

Specifies the style of numbering.

start(*positive_integer*)

Numbering begins at specified *positive_integer*.

continue

Numbering begins where the preceding ordered list left off.

Example: <ol outputclass="upper-roman start(10)"> specifies an ordered list which starts with an "X."

Note that it is still possible to specify any class name you want in the @outputclass attribute of the element. Example: <ol outputclass="continue fancy-list">.

Chapter 6. Giving a background color to table cells

This chapter explains how you can give a background color to table cells by adding a `bgcolor(color)` directive to the `@outputclass` attribute of most table elements.

It's possible to give a background color to table cells by adding a `bgcolor(color)` directive, where *color* is any CSS color value, to the `@outputclass` attribute of the following elements:

Inside a `<simpletable>` element

`<simpletable>`, `<sthead>`, `<strow>`, `<stentry>`.

Inside a `<table>` element

`<tgroup>`, `<thead>`, `<tbody>`, `<row>`, `<entry>`.

Example:

```
<table>
  <tgroup cols="2" outputclass="bgcolor(#F0FFFF)">
    <tbody>
      <row>
        <entry>C1,1</entry>
        <entry>C1,2</entry>
      </row>
      <row outputclass="bgcolor(#FFFFF0)">
        <entry>C2,1</entry>
        <entry>C2,2</entry>
      </row>
    </tbody>
  </tgroup>
</table>
```

Note that it is still possible to specify any class name you want in the `@outputclass` attribute of a table element. Example: `<simpletable outputclass="bgcolor(#FFFFF0) fancy-table">`.

Chapter 7. Syntax highlighting

This chapter explains how you can automatically colorize the source code contained in `<pre>`, `<codeblock>` or any other element specializing `<pre>`.

You can automatically colorize the source code contained in `<pre>`, `<codeblock>` or any other element specializing `<pre>`. This feature, commonly called *syntax highlighting*, has been implemented using an open source software component called "XSLT syntax highlighting".

If you want to turn on syntax highlighting in a DITA document, suffice to add attribute `@outputclass` to a `<pre>`, `<codeblock>` or any other element specializing `<pre>`. The value of attribute `@outputclass` must be any of: `language-bourne`, `language-c`, `language-cmake`, `language-cpp`, `language-csharp`, `language-css21`, `language-delphi`, `language-ini`, `language-java`, `language-javascript`, `language-lua`, `language-m2 (Modula 2)`, `language-perl`, `language-php`, `language-python`, `language-ruby`, `language-sql1999`, `language-sql2003`, `language-sql92`, `language-tcl`, `language-upc (Unified Parallel C)`, `language-html`, `language-xml`.

If you want to customize syntax highlighting for an HTML-based output format (XHTML, EPUB, etc), then redefine any of the following CSS styles:

- `.hl-keyword` (keywords of a programming language),
- `.hl-string` (string literal),
- `.hl-number` (number literal),
- `.hl-comment` (any type of comment),
- `.hl-docomment` (comments used as documentation, i.e. javadoc, or xmldoc),
- `.hl-directive` (preprocessor directive or in XML, a processing-instruction),
- `.hl-annotation` (annotations or "attributes" as they are called in .NET),
- `.hl-tag` (XML tag, i.e. element name),
- `.hl-attribute` (XML attribute name),
- `.hl-value` (XML attribute value),
- `.hl-doctype` (`<!DOCTYPE>` and all its content).

Customization of the syntax highlighting of a keyword for HTML-based output formats

```
.hl-keyword {
  font-weight: bold;
  color: #602060;
}
```

How to use a custom CSS stylesheet is explained in [Part II, Chapter 9, Section 1](#).

If you want to customize syntax highlighting for an XSL-FO-based output format (PDF, RTF, etc), then redefine any of the following `<attribute-set>`s: `hl-keyword`, `hl-string`, `hl-number`, `hl-comment`, `hl-docomment`, `hl-directive`, `hl-annotation`, `hl-tag`, `hl-attribute`, `hl-value`, `hl-doctype`.

Customization of the syntax highlighting of a keyword for XSL-FO-based output formats

```
<xsl:attribute-set name="hl-keyword" use-attribute-sets="hl-style">
  <xsl:attribute name="font-weight">bold</xsl:attribute>
  <xsl:attribute name="color">#602060</xsl:attribute>
</xsl:attribute-set>
```

How to use a custom XSLT stylesheet generating XSL-FO is explained in [Part II, Chapter 9, Section 2](#).

Chapter 8. Rich media content

This chapter explains how to add SVG, MathML, audio, video and Flash animations to your DITA topics and how **ditaac** processes this rich media content in the case where the output format supports rich media (e.g. XHTML 5, EPUB 3) and also in the case where the output format does not support rich media (e.g. XHTML 1, PDF, RTF).

SVG

It is possible to include SVG graphics in a DITA document either by reference or by inclusion. Use an `<svg-container>/<svgref>` element pointing to an SVG file to include it by reference. Example:



The XML source code corresponding to the above example is:

```
<p><svg-container><svgref href="media/graphic.svg" /></svg-container></p>
```

It's also possible to use an `<image>` element pointing to an SVG file to include it by reference. Example:

```
<p><image href="media/graphic.svg" /></p>
```

Embedding SVG graphics in a DITA document can be achieved using the same `<svg-container>` element. Example:



The XML source code corresponding to the above example is:

```
<p><svg-container>
  <svg:svg height="64.710144" version="1.1"
    viewBox="0 0 104.28986 51.768115" width="130.36232"
    xmlns:svg="http://www.w3.org/2000/svg">
    ...
  </svg:svg>
</svg-container></p>
```

Notes:

- It is still recommended to include SVG graphics by reference using the `<image>` element rather than `<svg-container>/<svgref>`. The `<image>` element has useful attributes (`@width`, `@height`, `@scale`, `@scalefit`) allowing to adjust the dimension of the image. Moreover this elements permits on the fly conversion between image formats.
- It is not recommended to embed SVG graphics in a DITA document as this is likely to cause many validation problems.
- Only the following screen formats may contain SVG: XHTML 5, XHTML 5 Web Help and EPUB 3. Note that only modern web browsers support XHTML 5 and XHTML 5 Web Help. Very few EPUB readers (e.g. iBooks) support EPUB 3.
- All XSL-FO based formats (PDF, RTF, DOCX, etc) support SVG whatever the XSL-FO processor you may use.

MathML

It is possible to include math in a DITA document either by reference or by inclusion. Use an `<mathml>/<mathmlref>` element pointing to a MathML file to include it by reference. Example:

$$\nabla E = \frac{\rho}{\epsilon_0}$$

The XML source code corresponding to the above example is:

```
<p><mathml><mathmlref href="media/math.mml" /></mathml></p>
```

Embedding MathML in a DITA document can be achieved using the same `<mathml>` element. Example:

$$\left\{ \begin{array}{l} \nabla \times \mathbf{E} = -\frac{\partial \mathbf{B}}{\partial t} \\ \nabla \times \mathbf{B} = \mu_0 \mathbf{J} + \mu_0 \epsilon_0 \frac{\partial \mathbf{E}}{\partial t} \end{array} \right.$$

The XML source code corresponding to the above example is:

```
<p><mathml>
  <m:math display="block"
    xmlns:m="http://www.w3.org/1998/Math/MathML">
    <m:row>
      ...
    </m:mrow>
  </m:math>
</mathml></p>
```

Notes:

- For clarity, it is recommended to wrap `<mathml>` into the following equation elements: `<equation-inline>`, `<equation-block>`, `<equation-figure>`.
- There is an option to number `<equation-figure>` elements having a `<title>`. Example:

Equation 8-1. Gauss's law in its differential form

$$\nabla E = \frac{\rho}{\epsilon_0}$$

`<equation-block>` elements containing an empty `<equation-number>` are automatically numbered. Example:

$$\nabla E = \frac{\rho}{\epsilon_0} \tag{8-1}$$


The counter used to number `<equation-figure>` elements having a `<title>` and the counter used to number `<equation-block>` elements containing an empty `<equation-number>` are different. Therefore mixing numbered `<equation-figure>`s and numbered `<equation-block>`s in the same DITA document may result in a hard to understand equation numbering.

- Only the following screen formats may contain MathML: XHTML 5, XHTML 5 Web Help and EPUB 3. Most modern web browsers (Firefox, Chrome) support XHTML 5 and XHTML 5 Web Help containing MathML. Very few EPUB readers (e.g. iBooks) support EPUB 3.
- XSL-FO based formats (PDF, RTF, DOCX, etc) support MathML depending on the XSL-FO processor you use:
 - Apache FOP requires you to download and install the the [JEUclid FOP plug-in](#).
 - RenderX XEP does not support MathML.

- Antenna House Formatter supports MathML as an option.
- XMLmind XSL-FO Converter supports MathML out of the box.

Audio

Use the `<object>` DITA element to add audio to your DITA topics. Example:

 [audio.mp3 \(audio/mpeg\)](#)

The XML source code corresponding to the above example is:

```
<p><object data="media/audio.mp3" type="audio/mpeg">
  <param name="source" value="media/audio.ogg"
    valuetype="ref" type="audio/ogg"/>

  <param name="source" value="media/audio.m4a"
    valuetype="ref" type="audio/mp4"/>

  <param name="source" value="media/audio.wav"
    valuetype="ref" type="audio/wav"/>

  <param name="controls" value="true"/>
</object></p>
```

Notes:

- The `@data` and `@type` attributes are required. The value of the `@type` attribute must start with "audio/".
- It is strongly recommended to specify *alternate audio files* as modern web browsers, while all supporting the HTML 5 `<audio>` element, vary in their support of audio formats. This is done by adding `<param>` child elements to the `<object>` element. Such `<param>` elements must have a `name="source"` attribute, a `valuetype="ref"` attribute, a `@value` attribute referencing an audio file and preferably, a `@type` attribute specifying the media type of the audio file.
- It is possible to add `<param>` elements corresponding to the attributes supported by the HTML 5 audio element (`<crossorigin>`, `<preload>`, `<autoplay>`, `<mediagroup>`, `<loop>`, `<muted>`, `<controls>`). In the above example, we have added a `<param>` element corresponding to the `@controls` HTML 5 attribute. Note that in the case of HTML 5 *boolean* attributes (`<autoplay>`, `<loop>`, `<muted>`, `<controls>`), the `@value` attribute of a `<param>` is not significant. For example, in the case of the above example, you could have specified "yes", "on", "1", etc, instead of "true".
- If the `<object>` element has a `<desc>` child element, then this `<desc>` element is used to generate fallback content in case audio is not supported. If the object element has no `<desc>` child element, then a simple fallback content is automatically generated by ditac. This automatic fallback content basically consists in a link allowing to download the audio file.
- When ditac is used to generate an XSL-FO based format (PDF, RTF, etc), only the fallback content appears in the output file.

Video

Use the `<object>` DITA element to add video to your DITA topics. Example:



[video.mp4 \(video/mp4\)](#)

The XML source code corresponding to the above example is:

```
<p><object data="media/video.mp4" type="video/mp4">
  <param name="source" value="media/video.ogv"
    valuetype="ref" type="video/ogg; codecs="theora, vorbis" />

  <param name="source" value="media/video.webm"
    valuetype="ref" type="video/webm" />

  <param name="width" value="320" />
  <param name="controls" value="yes" />
  <param name="poster" value="media/video_poster.jpg"
    valuetype="ref" />
</object></p>
```

Notes:

- The @data and @type attributes are required. The value of the @type attribute must start with "video/".
- It is strongly recommended to specify *alternate video files* as modern web browsers, while all supporting the HTML 5 <video> element, vary in their support of video formats. This is done by adding <param> child elements to the <object> element. Such <param> elements must have a name="source" attribute, a valuetype="ref" attribute, a @value attribute referencing a video file and preferably, a @type attribute specifying the media type of the video file.
- It is possible to add <param> elements corresponding to the attributes supported by the HTML 5 <video> element (<crossorigin>, <poster>, <preload>, <autoplay>, <mediagroup>, <loop>, <muted>, <controls>, <width>, <height>). In the above example, we have added a <param> element corresponding to the <width>, <controls> and <poster> HTML 5 attributes. Note that in the case of HTML 5 *boolean* attributes (<autoplay>, <loop>, <muted>, <controls>), the @value attribute of a <param> is not significant. For example, in the case of the above example, you could have specified "true", "on", "1", etc, instead of "yes".
- If the <object> element has a <desc> child element, then this <desc> element is used to generate fallback content in case video is not supported. If the object element has no <desc> child element, then a simple fallback content is automatically generated by ditac. This automatic fallback content basically consists in a link allowing to download the video file. The <param> element corresponding to the <poster> HTML 5 attribute, if present, is used to generate a nicer automatic fallback content.
- When ditac is used to generate an XSL-FO based format (PDF, RTF, etc), only the fallback content appears in the output file.

Flash animation

Use the <object> DITA element to add Adobe® Flash® animations to your DITA topics. Example:

 [animation.swf \(application/x-shockwave-flash\)](#)

(You may have to right-click on the above screenshot and select **Play** from the Flash popup menu to replay the animation.)

The XML source code corresponding to the above example is:

```
<p><object data="animation.swf"
  type="application/x-shockwave-flash"
  width="431" height="123">
```

```

<param name="movie" value="animation.swf"
        valuetype="ref" type="application/x-shockwave-flash"/>

<param name="menu" value="true"/>
<param name="quality" value="low"/>
</object></p>

```

Notes:

- The @data, @type, @width and @height attributes are required. The param name=movie child element having the same value as attribute @data is required too.
- You may add any other <param> child element supported by the Flash object. In the above example, you'll find menu and quality in addition to required movie.
- If the <object> element has a <desc> child element, then this <desc> element is used to generate fallback content in case Flash is not supported. If the object element has no <desc> child element, then a simple fallback content is automatically generated by ditac. This automatic fallback content basically consists in a link allowing to download the .swf file.
- When ditac is used to generate an XSL-FO based format (PDF, RTF, etc), only the fallback content appears in the output file.

Other uses of the <object> element

We have seen in previous sections how the <object> DITA element may be used to add audio, video and Adobe® Flash® animations to your DITA topics. In any case other than those described in previous sections, the <object> DITA element is converted to the equivalent <object> XHTML element. For example, if you want to add a YouTube video to your DITA topics, simply do it in DITA as you would do it in XHTML using the <object> element.



Watch this [test video](#) on YouTube.

The XML source code corresponding to the above example is:

```

<p><object data="https://www.youtube.com/embed/C0DPdy98e4c"
        width="640" height="360">
    <desc><img href="media/youtube_icon.png"/> Watch this <xref format="html"
        href="https://youtu.be/C0DPdy98e4c" scope="external">test video</xref> on
    YouTube.</desc>
</object></p>

```

Notes:

- If the <object> element has a <desc> child element, then this <desc> element is used to generate fallback content in case the media object is not supported. If the object element has no <desc> child element, then a simple fallback content is automatically generated by ditac. This automatic fallback content basically consists in a link allowing to download the media file.
- When ditac is used to generate an XSL-FO based format (PDF, RTF, etc), only the fallback content appears in the output file.

Actions

Unless you add param name="controls" (see [above](#)), you'll not be able to play audio or video. Even worse, without the controls <param>, an audio object is not rendered on screen (that is, it is invisible).

A simple solution for this problem is to insert a `<?onclick?>` processing-instruction in a DITA element (typically an *inline* element such as `<xref>` or `<ph>`). The `<?onclick?>` processing-instruction allows to specify an number of actions:

play

Play the associated resource from the beginning. Only applicable to video or audio targets.

pause

Pause playing . Only applicable to video or audio targets.

resume

Resume playing . Only applicable to video or audio targets.

mute

Mute sound . Only applicable to video or audio targets.

unmute

Unmute sound . Only applicable to video or audio targets.

show

Set the visibility property of the target element to visible.

hide

Set the visibility property of the target element to hidden.

The above actions are exactly those supported by EPUB 3's `<epub:trigger>`.


The `<?onclick?>` processing-instruction is processed by **ditaac** for the following output formats: XHTML 5, XHTML 5 Web Help and EPUB 3. It is discarded for any other output format.

The syntax for the content of `<?onclick?>` is:

```
onclick_data -> action (S action)*
action -> op '(' target_id? ')'
op -> 'play' | 'pause' | 'resume' | 'mute' | 'unmute'
      'show' | 'hide'
```

When *target_id* is not specified, it is taken from the `@href` attribute of the element containing the `<?onclick?>` processing-instruction. For example, `<xref href="#media/target_audio"><?onclick play()?>` is equivalent to: `<xref href="#media/target_audio"><?onclick play(media/target_audio)?>`.

Example 1: Say: "*Viens Hubble!*", which, in French, means: "Come here Hubble!".

 No audio. Say: "*Viens Hubble!*", which, in French, means: "Come here Hubble!".

The XML source code corresponding to the above example is:

```
<p>Example 1: <xref href="#media/target_audio"><?onclick play()?>
Say " <ph xml:lang="fr">Viens Hubble!</ph>"</xref>
...
<object data="media/audio.wav" id="audio_sample" type="audio/wav">
  <desc> ... </desc>
</object></p>
```

Example 2: Hide Hubble. Show Hubble.

Figure 8-1. My name is Hubble. I'm a 7-month old Golden Retriever.



The XML source code corresponding to the above example is:

```
<p>Example 2:  
<xref href="#media/target_image"><?onclick hide()?>Hide Hubble</xref>.  
<xref href="#media/target_image"><?onclick show()?>Show Hubble</xref>.</p>
```

Part II. Customizing the output of XMLmind DITA Converter

Chapter 9. Simple customization

1. Customize the look of the HTML pages generated by ditac

We'll explain how to customize the look of the HTML pages generated by ditac by using an example. Let's suppose we want to render topic titles in a nice dark blue color rather than in black.



Restriction

The customization method described below will not work for formats for which the generated HTML pages are automatically compiled or archived. This is the case for HTML Help, Java™ Help and EPUB.

About this task

The easiest way to customize the look of the HTML pages generated by ditac is to use a custom CSS stylesheet rather than the stock one.

Procedure

1. Create a custom CSS stylesheet importing the stock CSS stylesheet.

The stock CSS stylesheet is found in:

`ditac_install_dir/xsl/xhtml/resources/basic.css`

Used for the XHTML 1.0, XHTML 1.1, HTML 4.01 and XHTML 5 output formats.

`ditac_install_dir/xsl/webhelp/resources/webhelp.css`

Used for the Web Help output format.

`ditac_install_dir/xsl/htmlhelp/resources/basic.css`

Used for the HTML Help output format.

`ditac_install_dir/xsl/eclipsehelp/resources/basic.css`

Used for the Eclipse Help output format.

`ditac_install_dir/xsl/javahelp/resources/javahelp.css`

Used for the Java™ Help output format.

`ditac_install_dir/xsl/epub/resources/basic.css`

Used for the EPUB output format.

Initial contents of the custom CSS stylesheet (a copy of this file is found in `customize/custom.css`).

```
@import url(basic.css);
```

2. Add one or more rules to the custom CSS stylesheet.

The XSLT stylesheets generating XHTML/HTML pages make extensive use of the class attribute. Generally the XHTML element generated for a DITA element has a `class` attribute bearing the name of the DITA element. Example: a DITA `<p>` is converted to a XHTML `<div class="p">`.

For more information, you'll have to refer to the stock CSS stylesheet or even to the HTML pages generated by ditac.


```
@import url(basic.css);

.part-title,
.chapter-title,
.appendix-title,
.section1-title,
.section2-title,
.section3-title,
.section4-title,
.section5-title,
.section6-title,
.section7-title,
.section8-title,
.section9-title,
.topic-title {
    color: #403480;
    border-bottom: 1px solid #403480;
}
```

3. Specify the "-p `css res/custom.css`" option when running `ditac`.

```
$ ditac -images img -p xsl-resources-directory res \
-p css res/custom.css \
out/manual/_html manual.ditamap
```

4. Last but not least, do not forget to copy `custom.css` to the resources subdirectory of the output directory (`out/manual/res/` in the case of the above example). The `ditac` command-line cannot do that automatically for you.

```
$ cp customize/custom.css out/manual/res
```

2. Customizing the look of the PDF files generated by ditac

We'll explain how to customize the look of the PDF files generated by `ditac` by using an example. Let's suppose we want to render topic titles in a nice dark blue color rather than in black.

About this task

A PDF file is created by converting the XSL-FO file generated by the `ditac` XSLT 2.0 stylesheet by the means of an XSL-FO processor such as Apache FOP, RenderX XEP or Antenna House Formatter. Therefore we need to generate a custom XSL-FO file. This is done by creating a very simple variant of the stock XSLT stylesheet which generates XSL-FO.

Procedure

1. Create a custom XSLT stylesheet importing the stock one.

This stock XSLT stylesheet is found in `ditac_install_dir/xsl/fo/fo.xsl`. It is used to generate an intermediate XSL-FO file. After that, the XSL-FO file is converted to PDF, PostScript, RTF, WordprocessingML, Office Open XML (.docx) or OpenOffice/LibreOffice (.odt) by the means of an XSL-FO processor.

Initial contents of the custom XSLT stylesheet (a copy of this file is found in `customize/custom_fo.xsl`).

```
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  xmlns:fo="http://www.w3.org/1999/XSL/Format"
  version="2.0">

  <xsl:import href="ditac-xsl:fo/fo.xsl"/>

</xsl:stylesheet>
```

Notice the funny looking URI "ditac-xsl:fo/fo.xsl". "ditac-xsl:" is an easy way to refer to *ditac_install_dir/xsl/*. This works because the XML catalog used by the **ditac** command-line utility (found in *ditac_install_dir/schema/catalog.xml*) contains:

```
<rewriteURI uriStartString="ditac-xsl:" rewritePrefix="../xsl/" />
```

2. Redefine one or more named `xsl:attribute-sets` in your custom XSLT stylesheet.

Named `xsl:attribute-sets` are not documented yet. For more information, you'll have to refer to the XSLT stylesheets found in *ditac_install_dir/xsl/fo/*.

```
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  xmlns:fo="http://www.w3.org/1999/XSL/Format"
  version="2.0">

  <xsl:import href="ditac-xsl:fo/fo.xsl"/>

  <xsl:attribute-set name="topic-title" use-attribute-sets="topic-title">
    <xsl:attribute name="color">#403480</xsl:attribute>
    <xsl:attribute name="font-size">160%</xsl:attribute>
    <xsl:attribute name="padding-bottom">0.05em</xsl:attribute>
    <xsl:attribute name="border-bottom">0.5pt solid #403480</xsl:attribute>
    <xsl:attribute name="space-before.optimum">1.5em</xsl:attribute>
    <xsl:attribute name="space-before.minimum">1.2em</xsl:attribute>
    <xsl:attribute name="space-before.maximum">1.8em</xsl:attribute>
  </xsl:attribute-set>

</xsl:stylesheet>
```

3. Specify the `-t customize/custom_fo.xsl` option when running **ditac**.

```
$ ditac -t customize/custom_fo.xsl \
  out/manual.pdf manual.ditamap
```

Alternatively, package your custom XSLT stylesheet as a **plug-in** and then specify the name of this plug-in using the `-plugin` command-line option. By doing this, your custom XSLT stylesheet will be automatically used whatever the output format which uses XSL-FO as its intermediate format (PDF, RTF, .odt, .docx, etc).

Chapter 10. Using ditac to convert documents conforming to a DITA specialization

We'll explain by example how to use ditac to convert documents conforming to a DITA specialization. Let's suppose we have a DITA specialization which adds `<time>` and `<kbd>` elements (similar to `<time>` and `<kbd>` HTML5 elements) topic contents. These elements are modeled as follows (see `sample_plugin/dtd/sampleDomain.mod`)

```
<!ENTITY % time "time">

<!ELEMENT time (#PCDATA | %text;)*>

<!ATTLIST time %univ-atts;
              outputclass CDATA #IMPLIED>

<!ATTLIST time %global-atts;
              class CDATA "+ topic/ph sample-d/time ">

<!ATTLIST time %univ-atts;
              datetime CDATA #IMPLIED>

<!ENTITY % kbd "kbd">

<!ELEMENT kbd (#PCDATA | %text;)*>

<!ATTLIST kbd %univ-atts;
              outputclass CDATA #IMPLIED>

<!ATTLIST kbd %global-atts;
              class CDATA "+ topic/ph sample-d/kbd ">
```

All the example files of this tutorial have been packaged as a **plug-in** called "sample_plugin". They are found in directory `sample_plugin/`. In order to give this plug-in a try, you'll have to copy directory `sample_plugin/` to `ditac_install_dir/plugin/`.

About this task

Using ditac to convert documents conforming to a DITA specialization basically requires customizing the output of the tool using the same techniques as those explained in [Chapter 9, Section 1](#) and [Chapter 9, Section 2](#).

Procedure

1. Create an XML catalog pointing to a local copy of your custom DTD. This file must be named `catalog.xml` and must be found in your plug-in directory.

File `sample_plugin/catalog.xml`:

```
<catalog xmlns="urn:oasis:names:tc:entity:xmlns:xml:catalog"
         prefer="public">

  <public publicId="-//OASIS//DTD DITA Concept//EN"
          uri="dtd/concept.dtd"/>
```

```

<public publicId="-//OASIS//DTD DITA Composite//EN"
        uri="dtd/ditabase.dtd"/>

<public publicId="-//OASIS//DTD DITA General Task//EN"
        uri="dtd/generalTask.dtd"/>

...

</catalog>

```

2. Create a customization of `ditac_install_dir/xsl/xhtml/xhtml.xsl` as explained in [Chapter 9, Section 2](#). This file must be found in `your_plugin_dir/xsl/xhtml/xhtml.xsl` in order to be used by ditac.

File `sample_plugin/xsl/xhtml/xhtml.xsl`:

```

<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
                xmlns="http://www.w3.org/1999/xhtml"
                version="2.0">

  <xsl:import href="ditac-xsl/xhtml/xhtml.xsl"/>

  <xsl:template match="*[contains(@class,' sample-d/kbd ')]">
    <tt>
      <xsl:call-template name="commonAttributes"/>
      <xsl:apply-templates/>
    </tt>
  </xsl:template>

  ...

</xsl:stylesheet>

```

Note that the XSLT template called `commonAttributes` adds a `class="kbd"` attribute to the generated `<tt>` element. This makes it easy styling the generated `<tt>` element using the technique explained in [Chapter 9, Section 1](#).

3. Create a customization of `ditac_install_dir/xsl/fo/fo.xsl` as explained in [Chapter 9, Section 2](#). This file must be found in `your_plugin_dir/xsl/fo/fo.xsl` in order to be used by ditac.

File `sample_plugin/xsl/fo/fo.xsl`:

```

<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
                xmlns:fo="http://www.w3.org/1999/XSL/Format"
                version="2.0">

  <xsl:import href="ditac-xsl/fo/fo.xsl"/>

  <xsl:attribute-set name="kbd" use-attribute-sets="monospace-style">
    <xsl:attribute name="border">1px solid #C0C0C0</xsl:attribute>
    <xsl:attribute name="background-color">#F0F0F0</xsl:attribute>
    <xsl:attribute name="padding">0.25em</xsl:attribute>
  </xsl:attribute-set>

```

```
<xsl:template match="*[contains(@class,' sample-d/kbd ')]">
  <fo:inline xsl:use-attribute-sets="kbd">
    <xsl:call-template name="commonAttributes"/>
    <xsl:apply-templates/>
  </fo:inline>
</xsl:template>

...

</xsl:stylesheet>
```

4. Pass command-line option `-plugin plugin_name` to `ditac` in order to use the DTDs (or schemas) and the XSLT stylesheets found in your plug-in subdirectory, preferably to those found in `ditac_install_dir/schema/` and in `ditac_install_dir/xsl/`.

You'll find a sample DITA document making use of the custom `<time>` and `<kbd>` elements in `sample_plugin/sample/sample.ditamap`. You can convert this sample document to single-page XHTML and to PDF by running `sample_plugin/sample/run.sh` (`sample_plugin\sample\run.bat` on Windows):

```
../../../../bin/ditac -plugin sample_plugin \
  out/sample.pdf sample.ditamap
```

Chapter 11. Extensive customization

In order to extensively customize the output of ditac, you need to learn how it works.

Basically, this means that you'll have to understand the contents of the `ditac_lists.ditac_list` file and the `.ditac` files, which are generated by the ditac *preprocessor*.

An extensive customization works exactly like a simple one:

1. Create a custom XSLT 2.0 stylesheet which imports the stock one.
2. Redefine one or more attribute sets and/or one or more templates in the custom XSLT 2.0 stylesheet.

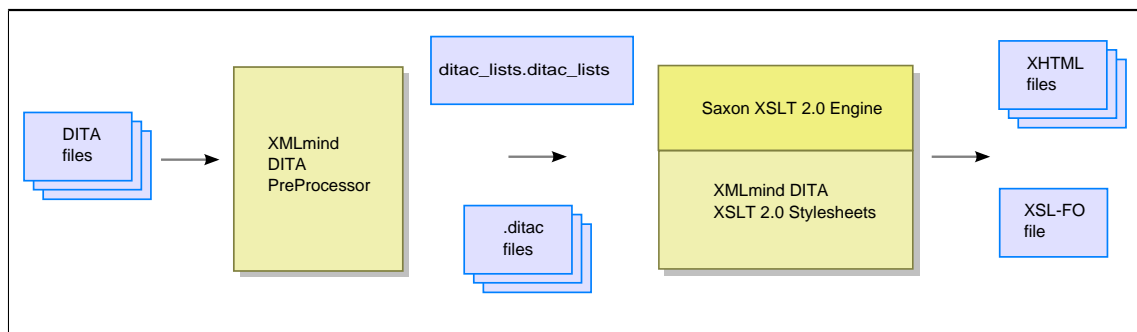
The only difference is that this time, you need to know exactly what is the format of the files you are going to transform. The bad news first: the ditac XSLT 2.0 stylesheets do not transform plain DITA files. They transform `.ditac` files, which are *fully preprocessed DITA files*. Now, the good news: `.ditac` files mainly contains DITA elements and because the ditac preprocessor performs all the grunt work beforehand, `.ditac` files are really straightforward to transform.

In fact, transforming `.ditac` files rather than plain DITA files allows to concentrate on creating great-looking output.

How it works

The ditac preprocessor generates a single `ditac_lists.ditac_list` file and one or more `.ditac` files⁽⁷⁾ out of the source DITA files.

Figure 11-1. The intermediate files generated by the ditac preprocessor



Then, each `.ditac` file, which mainly contains fully preprocessed DITA topics, is transformed in turn by the ditac XSLT 2.0 stylesheets.

The `ditac_lists.ditac_list` file, which contains useful information about the overall DITA document being converted, is not directly transformed by the ditac XSLT 2.0 stylesheets. Instead, when needed to, the ditac XSLT 2.0 stylesheets *query* the `ditac_lists.ditac_list` file in order to generate optional items. Example: number topics, tables, figures, etc, when parameter `number='XXX'` has been specified.

It is possible to examine the contents of the `ditac_lists.ditac_list` file and those of the `.ditac` files by specifying the `-preprocess` command-line option. Example:

```
$ ditac -preprocess \
-v -chunk single \
-images img -p xsl-resources-directory res \
out/manual.html manual.ditamap
```

⁽⁷⁾A single `.ditac` file for a print output; one or more `.ditac` files for a screen output.

Contents of a .ditac file

The root element of a .ditac file is `<ditac:chunk>`. A `<ditac:chunk>` element may have the following child elements (in any order and in any number):

<ditac:titlePage>

This empty placeholder element means: generate a "title page" section here.

<ditac:toc>

This empty placeholder element means: generate a **Table of Contents** section here.

<ditac:figureList>

This empty placeholder element means: generate a **List of Figures** section here.

<ditac:tableList>

This empty placeholder element means: generate a **List of Tables** section here.

<ditac:exampleList>

This empty placeholder element means: generate a **List of Examples** section here.

<ditac:equationList>

This empty placeholder element means: generate a **List of Equations** section here.

<ditac:indexList>

This empty placeholder element means: generate an **Index** section here.

A DITA topic of any kind

A fully preprocessed topic. This topic is guaranteed not to contain nested topics.

<ditac:flags>

A wrapper element for a DITA topic of any kind. A `<ditac:flags>` element is used to wrap any DITA element which has been flagged by the means of a conditional processing profile (a .ditaval file). See the [-filter](#) command-line option.

More formally, the content model of `<ditac:chunk>` is specified by the `schema/ditac.rnc` RELAX NG grammar.

Example:

```
<ditac:chunk xmlns:ditac="http://www.xmlmind.com/ditac/schema/ditac"
  xmlns:ditaarch="http://dita.oasis-open.org/architecture/2005/">
  <ditac:titlePage/>
  <ditac:toc/>
  <topic class="- topic/topic "
    domains="(topic ui-d) (topic hi-d) (topic pr-d) (topic sw-d)
      (topic ut-d) (topic indexing-d)"
    id="introduction" ditaarch:DITAArchVersion="1.1">
    <title class="- topic/title ">Introduction</title>
    .
    .
    .
  </topic>
  <topic class="- topic/topic " id="I_2yl4p_">
    <title class="- topic/title ">Using XMLmind DITA Converter</title>
  </topic>
  <task class="- topic/topic task/task "
    domains="(topic ui-d) (topic hi-d) (topic pr-d) (topic sw-d)
      (topic ut-d) (topic indexing-d)"
```

```

    id="install" ditaarch:DITAArchVersion="1.1">
    <title class="- topic/title ">Installing XMLmind DITA Converter</title>
    .
    .
    .
  </task>
  .
  .
  .
  <ditac:indexList/>
</ditac:chunk>

```



Important

The DITA topics contained in a `.ditac` file are fully preprocessed. What does this mean? Basically that they are ready to be transformed without further efforts:

- Conref inclusions have been processed.
- Unspecified attributes having default values have been added to the elements. Example: a `<p>` element becomes `<p class="- topic/p ">`.
- Elements now have a ``flat'', globally unique, ID. Example: the `@id` attribute of this `<title>` element `<topic id="introduction"><title id="start">` becomes `id="introduction__start"`.
- The `@href` attribute of `<xref>`, `<link>`, `<image>`, `<svgref>`, `<mathmlref>` elements now point to the (future) output files. Example: the `@href` attribute of this `<xref>` element `<xref href="intro.dita#introduction/start">` becomes `href="userguide-1.html#introduction__start"`.
- Some text may have been added to empty `<xref>` and `<link>` elements.
- The `<reltable>` elements of the DITA map have been converted to `<related-links>` sections or to extra `<link>` elements.
- Filtered elements have been removed. Flagged elements have been wrapped in a `<ditac:flags>` element.

Contents of the `ditac_lists.ditac_list` file

The root element of the `ditac_lists.ditac_list` file is `<ditac:lists>`. A `<ditac:lists>` element may have the following child elements (in this exact order and in this exact number):

`<ditac:chunkList>`

A `<ditac:chunkList>` contains a `<ditac:chunk>` element for each `.ditac` file. A `<ditac:chunk>` element may be seen as the *manifest* of a `.ditac` file.

Example:

```

<ditac:chunkList>
  <ditac:chunk file="manual.html">
    <ditac:titlePage/>
    <ditac:toc/>
    <ditac:topic id="introduction" number="bookabstract.1"
      role="bookabstract" title="Introduction"/>

```



```

.
.
.
  <ditac:topic id="limitations" number="appendix.1" role="appendix"
              title="Limitations and implementation specificities"/>
  <ditac:indexList/>
</ditac:chunk>
</ditac:chunkList>

```

<ditac:titlePage>

Contains all the DITA elements needed to generate the ``title page" section of a document. This element exists but is empty if the DITA document being converted has no title and no metadata (e.g. <topicmeta>/<autor>, <bookmeta>/<publisherinformation>, etc).

Example:

```

<ditac:titlePage>
  <title class="- topic/title ">XMLmind DITA Converter Manual</title>
  <bookmeta class="- map/topicmeta bookmap/bookmeta ">
    <authorinformation class="+ topic/author xnal-d/authorinformation ">
      .
      .
      .
    <critdates class="- topic/critdates ">
      <created class="- topic/created " date="September 17, 2009"/>
    </critdates>
  </bookmeta>
</ditac:titlePage>

```

<ditac:toc>

Contains all the information needed to generate **Table of Contents** section of a document.

Example:

```

<ditac:toc>
  <ditac:tocEntry file="manual.html" id="I_2yl4p_" number="part.1"
                 role="part" title="Using XMLmind DITA Converter">
    <ditac:tocEntry file="manual.html" id="install"
                   number="part.1 chapter.1" role="chapter"
                   title="Installing XMLmind DITA Converter">
      .
      .
      .
    </ditac:tocEntry>
  </ditac:tocEntry>
  <ditac:tocEntry file="manual.html" id="limitations" number="appendix.1"
                 role="appendix"
                 title="Limitations and implementation specificities"/>
</ditac:toc>

```

<ditac:figureList>

Contains all the information needed to generate the **List of Figures** section of a document. This element exists but is empty if the DITA document being converted contains no <fig> elements having a <title> child element.

Example:

```
<ditac:figureList>
  <ditac:figure file="manual.html" id="xsltParams__page_areas"
    number="part.1 chapter.4 figure.1" title="Page areas"/>
  <ditac:figure file="manual.html" id="howItWorks__I_6gb2s_"
    number="part.2 chapter.3 figure.1"
    title="The intermediate files generated by
    the ditac preprocessor"/>
</ditac:figureList>
```

<ditac:tableList>

Contains all the information needed to generate the **List of Tables** section of a document. This element exists but is empty if the DITA document being converted contains no <table> elements having a <title> child element.

Example:

```
<ditac:tableList>
  <ditac:table file="manual.html"
    id="quickStart__supported_filename_extensions"
    number="part.1 chapter.2 table.1"
    title="Supported filename extensions"/>
  <ditac:table file="manual.html"
    id="quickStart__supported_output_formats"
    number="part.1 chapter.2 table.2"
    title="Supported output formats"/>
</ditac:tableList>
```

<ditac:exampleList>

Contains all the information needed to generate the **List of Examples** section of a document. This element exists but is empty if the DITA document being converted contains no <example> elements having a <title> child element.

Example:

```
<ditac:exampleList>
  <ditac:example file="topicAAA.htm" id="topicAAA__I_6wcr3_"
    number="part.1 example.1" title="Example AAA.1"/>
  .
  .
  .
  <ditac:example file="topicHHH.htm" id="topicHHH__exampleHHH.2"
    number="appendix.2 example.2" title="Example HHH.2"/>
</ditac:exampleList>
```

<ditac:equationList>

Contains all the information needed to generate the **List of Examples** section of a document. This element exists but is empty if the DITA document being converted contains no <equation-figure> elements having a <title> child element.

Example:

```
<ditac:equationList>
  <ditac:equation file="part222.html" id="part222__first_equation"
    number="part.2 equation.1" title="First equation">
```

```

<ditac:description>This is a short description of the first equation.
It should be displayed in the <i class="+ topic/ph hi-d/i ">
<b class="+ topic/ph hi-d/b ">List of
Equations</b></i>.</ditac:description>
</ditac:equation>
.
.
.
<ditac:equation file="trademarks.html" id="trademarks__I_qa9vmk_"
number="equation.4" title="Second equation"/>
</ditac:equationList>

```

<ditac:indexList>

Contains all the information needed to generate the **Index** section of a document. This element exists but is empty if the DITA document being converted contains no <indexterm> elements.

Example:

```

<ditac:indexList>
  <ditac:div title="A">
    <ditac:indexEntry term="appendix-number-format, parameter"
      xml:id="I_hd1wr_">
      <ditac:indexAnchor file="manual.html"
        id="xsltParams__I_8bona_"
        number="1" />
    </ditac:indexEntry>
    <ditac:indexEntry sortAs="automap" term="-automap, option"
      xml:id="I_2gud9_">
      <ditac:indexAnchor file="manual.html"
        id="commandLine__I_5x8va_"
        number="1" />
    </ditac:indexEntry>
  </ditac:div>
  .
  .
  .
  <ditac:div title="X">
    .
    .
    .
    <ditac:indexEntry sortAs="xslt" term="-xslt, option"
      xml:id="I_atn9k_">
      <ditac:indexAnchor file="manual.html"
        id="commandLine__I_cu3ew_"
        number="1" />
      <ditac:indexAnchor file="manual.html"
        id="customAttributeSet__I_gis5b_" number="2" />
      <ditac:indexAnchor file="manual.html"
        id="specialize__I_11514_"
        number="3" />
      <ditac:indexSeeAlso ref="I_bhy05_" term="-t, option" />
      <ditac:indexSeeAlso ref="I_fljh_" term="-xslt, option" />
    </ditac:indexEntry>

```

```
</ditac:div>  
</ditac:indexList>
```

More formally, the content model of `<ditac:lists>` is specified by the `schema/ditac_lists.rnc` RELAX NG grammar.

Currently the `ditac_lists.ditac_list` file is used to generate:

- the "title page" section of a document;
- the **Table of Contents** section of a document;
- the **List of Figures**, **List of Tables**, **List of Examples**, **List of Equations** sections of a document;
- the **Index** section of a document;
- the navigation icons in a multi-page HTML document;
- all the files (`project.hhp`, `toc.hhc`, etc) required by the HTML Help system;
- all the files (`jhelpset.hs`, `jhelpmap.jhm`, etc) required by the Java™ Help system.

Part III. Embedding XMLmind DITA Converter in a Java™ application

Chapter 12. High-level method: embedding `com.xmlmind.ditac.convert.Converter`

Quick and easy embedding: embed `com.xmlmind.ditac.convert.Converter`, the Java™ class which is used to implement the `ditac` command-line utility.

`Converter` is the object which is at the core of the `ditac` command-line utility. Its `run` method accepts the same string arguments as the `ditac` command-line utility.

The full source code of the `Embed1` sample is found in `Embed1.java`.

1. Create the `Converter`.

```
StyleSheetCache cache = new StyleSheetCache();

Console console = new Console() {
    public void showMessage(String message, MessageType messageType) {
        System.err.println(message);
    }
};

Converter converter = new Converter(cache, console);
```

- `StyleSheetCache` is a simple cache for the `ditac` XSLT 2.0 stylesheets. It is a thread-safe object which is intended to be shared by several `Converters`.

Unlike `StyleSheetCache`, `Converter` is not thread-safe. Each thread must own its `Converter`. However, the `run` method of a `Converter` may be invoked several times.

- `Console` is a very simple interface. Implementing this interface allows to do whatever you want with the messages reported by a `Converter`.

2. Configure the `Converter`.

```
if (!converter.registerFOP(path("/opt/fop/fop"))) {
    return 1;
}

File xslDir = new File(path("../..../xsl"));
System.setProperty("DITAC_XSL_DIR", xslDir.getAbsolutePath());
```

- There are several methods which allows to register an XSL-FO processor with a `Converter`. From high-level ones to low-level ones, these methods are: `registerFOP`, `registerXEP`, `register-AHF`, `registerXFC`, `registerExternalFOConverter`, `registerFOConverter`.

- A `Converter` can automatically determine where to find the directory containing all the `ditac` XSLT 2.0 stylesheets. It assumes that `ditac.jar` is in the class path of Java™. It also assumes that `ditac.jar` is contained in a `lib/` directory. The `xsl/` directory is assumed to be a sibling of the `lib/` directory.

However, the situation may be very different in applications other than the `ditac` command-line. That's why you may need to set the `DITAC_XSL_DIR` system property to the absolute path of the `xsl/` directory containing the XSLT stylesheets.

3. Invoke the `run` method.

```
String[] args = {
```

```
        "-v",
        "-p", "number", "all",
        outFile.getPath(),
        inFile.getPath(),
    };

    return converter.run(args);
```

The `run` method returns 0 if the conversion is successful and an integer greater than 0 otherwise. When the conversion fails, error messages are displayed on the `Console`.

Compiling and executing the `Embed1` sample

Compile the `Embed1` sample by running `ant` in `ditac_install_dir/doc/manual/embed/`.

Execute the `Embed1` sample by running `ant embed1` in `ditac_install_dir/doc/manual/embed/`. This will convert `ditac_install_dir/docsrc/manual/manual.ditamap` to `ditac_install_dir/doc/manual/embed/manual.pdf`, using Apache FOP.

Note that `Embed1.java` contains "hardwired filenames". This means that this sample cannot be run from elsewhere than `ditac_install_dir/doc/manual/embed/` and that you'll almost certainly need to modify the source code in order to specify the actual location of the `fop` (`fop.bat`) script.

Related information

- Chapter 13. Low-level method: embedding `com.xmlmind.ditac.preprocess.PreProcessor`

Chapter 13. Low-level method: embedding `com.xmlmind.ditac.preprocess.PreProcessor`

Advanced embedding method: first invoke a preprocessor which will generate intermediate `.ditac` files, then invoke the XSLT 2.0 engine in order to transform all these `.ditac` files.

This method consists in first invoking the `PreProcessor` in order to pre-process the DITA source files into a `ditac_lists.ditac_lists` file and one or more `.ditac` files; then invoking the [Saxon XSLT 2.0](#) engine in order to transform all the `.ditac` files.

For some output formats, PDF, RTF, etc, the final third step consists in invoking an XSL-FO processor such as [Apache FOP](#) in order to convert the XSL-FO generated by the XSLT stylesheets to the desired output format.

The full source code of the `Embed2` sample is found in `Embed2.java`.

1. Invoke the `ditac PreProcessor` to pre-process the DITA source files into a `ditac_lists.ditac_lists` file and one or more `.ditac` files.
 - a. Create and configure the `PreProcessor`.

```

Console console = new Console() {
    public void showMessage(String message, MessageType messageType) {
        System.err.println(message);
    }
};

PreProcessor preProc = new PreProcessor(console);
preProc.setChunking(Chunking.SINGLE);
preProc.setMedia(Media.SCREEN);

ResourceCopier resourceCopier = new ResourceCopier();
resourceCopier.parseParameters("img");
preProc.setResourceHandler(resourceCopier);

```

- `Console` is a very simple interface. Implementing this interface allows to do whatever you want with the messages reported by a `PreProcessor`.
- Specifying `preProc.setChunking(Chunking.SINGLE)` allows to generate a single HTML page using a DITA map designed to generate multiple HTML pages.
- A `PreProcessor` is not concerned about the *exact* output format. However its behaves differently depending on the target `Media`.
- A `PreProcessor` handles to an `ResourceHandler` all the resource files, typically image files, referenced in the DITA source using relative URLs. An `ResourceHandler` is registered with a `PreProcessor` using method `setResourceHandler`.

In the case of the `Embed2` sample, we use the simplest possible `ResourceHandler` which is `ResourceCopier`.

- b. Pre-process the DITA source files.

```

URL inFileURL = null;
try {
    inFileURL = inFile.toURI().toURL();
} catch (MalformedURLException cannotHappen) {}

```



```

File[] preProcFiles = null;
try {
    preProcFiles = preProc.process(new URL[] { inFileURL }, outFile);
} catch (IOException e) {
    console.showMessage(e.toString(), Console.MessageType.ERROR);
}
if (preProcFiles == null) {
    return false;
}

```

The `process` method of a `PreProcessor` returns `null` if an error other than an `IOException` has caused the pre-processing to fail. When this is the case, error messages are displayed on the Console.

Note that a `PreProcessor` is not thread-safe. Each thread must own its `PreProcessor`. However, the `process` method of a `PreProcessor` may be invoked several times.

2. Invoke the Saxon XSLT 2.0 engine, in order to transform all the `.ditac` files. Note that this is done using the standard `JAXP` API.
 - a. Pass *required system parameters* to the XSLT stylesheets, in addition to the normal, user, parameters.

```

String ditacListsURI = "";

int count = preProcFiles.length;
for (int i = 0; i < count; ++i) {
    File ditacFile = preProcFiles[i];

    if (ditacFile.getPath().endsWith(".ditac_lists")) {
        ditacListsURI = ditacFile.toURI().toASCIIString();
        break;
    }
}

String[] params = {
    "ditacListsURI", ditacListsURI,
    "xsl-resources-directory", "res",
    "use-note-icon", "yes",
    "default-table-width", "100%"
};

```

These required system parameters are:

- `ditacListsURI`, always required.
 - `foProcessor`, required by the XSLT stylesheets that generate XSL-FO.
 - `chmBasename`, `hhpBasename`, required by the XSLT stylesheets that generate HTML Help.
- b. Use the Saxon XSLT 2.0 engine to create a `TransformerFactory`, then configure this `TransformerFactory`.

```

private static
TransformerFactory createTransformerFactory(URIResolver uriResolver,

```

```

                                                    ErrorListener errorListener)
    throws Exception {
    Class<?> cls = Class.forName("net.sf.saxon.TransformerFactoryImpl");
    TransformerFactory transformerFactory =
        (TransformerFactory) cls.newInstance();

    ExtensionFunctions.registerAll(transformerFactory);

    transformerFactory.setURIResolver(uriResolver);
    transformerFactory.setErrorListener(errorListener);

    return transformerFactory;
}

```

- Creating an instance of Saxon 9.2+ is absolutely needed. XMLmind DITA Converter is not designed to work with any other XSLT engine (e.g. the Xalan XSLT 1.0 engine, which is part of the Java™ runtime).
- The ditac XSLT 2.0 stylesheets make use of a few XSLT extension functions written in Java™. These extension functions must be registered with Saxon. This is done using `ExtensionFunctions.registerAll`.

c. Create and configure a Transformer.

```

private static Transformer createTransformer(String[] params,
                                           Console console)
    throws Exception {
    URIResolver uriResolver = Resolve.createURIResolver();
    ErrorListener errorListener = new ConsoleErrorListener(console);

    TransformerFactory factory = createTransformerFactory(uriResolver,
                                                         errorListener);

    File xslFile = new File(path("../.../xsl/xhtml/html.xsl"));
    Transformer transformer =
        factory.newTransformer(new StreamSource(xslFile));

    transformer.setURIResolver(uriResolver);
    transformer.setErrorListener(errorListener);

    for (int i = 0; i < params.length; i += 2) {
        transformer.setParameter(params[i], params[i+1]);
    }

    return transformer;
}

```

- `Resolve` is a helper class making it easy to use the services of XML Catalog resolvers.

By default, `Resolve` automatically loads all the XML catalogs specified using the `xml.catalog.files` Java™ system property. Excerpts of the `ant build.xml` file:

```

<target name="embed2" depends="compile,clean_embed2">
    <java classpathref="cp" fork="yes" classname="Embed2">

```

```

<sysproperty key="xml.catalog.files"
              value="${ditac.dir}/schema/catalog.xml" />
<arg value="${ditac.dir}/docsrc/manual/manual.ditamap" />
<arg value="manual.html" />
</java>
</target>

```

However, static method `setResolverFactory` allows to configure this thread-safe utility class (used by ditac in many places) differently.

- `ConsoleErrorListener` is an implementation of `ErrorListener` which displays its messages on a Console.

d. Invoke the Transformer to transform each `.ditac` file.

```

for (int i = 0; i < count; ++i) {
    File ditacFile = preProcFiles[i];

    String ditacFilePath = ditacFile.getPath();
    if (ditacFilePath.endsWith(".ditac")) {
        File transformedFile = new File(
            ditacFilePath.substring(0, ditacFilePath.length()-5) +
            "html");

        try {
            transformer.transform(new StreamSource(ditacFile),
                                 new StreamResult(transformedFile));
        } catch (Exception e) {
            console.showMessage(e.toString(),
                               Console.MessageType.ERROR);
            cleanUp(preProcFiles);
            return false;
        }
    }
}

```

In the case of `Embed2`, the above loop is not strictly needed. We specified `preProc.setChunking(Chunking.SINGLE)` and therefore the `PreProcessor` generates a single `.ditac` file.

3. Copy the resources of the XSLT stylesheets (CSS stylesheets, icons, etc) to output subdirectory `res/`. Note that the images referenced in the DITA source, if any, have already been copied to output subdirectory `img/` by the `ImageCopier`.

```

File dstDir = new File("res");
if (!dstDir.exists()) {
    File srcDir = new File(path("../..../xsl/xhtml/resources"));
    try {
        FileUtil.copyDir(srcDir, dstDir, false);
    } catch (IOException e) {
        console.showMessage(e.toString(), Console.MessageType.ERROR);
        cleanUp(preProcFiles);
        return false;
    }
}

```

4. Delete the `ditac_lists.ditac_lists` and `.ditac` files.

```
cleanUp(preProcFiles);
```

Compiling and executing the `Embed2` sample

Compile the `Embed2` sample by running **ant** in `ditac_install_dir/doc/manual/embed/`.

Execute the `Embed2` sample by running **ant** `embed2` in `ditac_install_dir/doc/manual/embed/`. This will convert `ditac_install_dir/docsrc/manual/manual.ditamap` to single HTML 4.01 page `ditac_install_dir/doc/manual/embed/manual.html`.

Note that `Embed2.java` contains ``hardwired filenames". This means that this sample cannot be run from elsewhere than `ditac_install_dir/doc/manual/embed/`.

Related information

- [Part II, Chapter 11. Extensive customization](#)
- [Chapter 12. High-level method: embedding `com.xmlmind.ditac.convert.Converter`](#)

Appendix A. About DITA support in XMLmind DITA Converter

DITA 1.3 support

As of version 3.0, XMLmind DITA Converter (ditac for short) fully supports DITA 1.3 and as such, allows to convert DITA documents conforming to the DITA 1.3 DTD, W3C XML Schema or *RELAX NG schema*. However, there are still limitations, deemed minor, and implementation specificities which are documented in [Appendix B](#).

In fact, when ditac v2.6+ is used, DITA 1.2 documents are automatically “upgraded” to DITA 1.3. This is caused by the fact that the following `<!DOCTYPE>` means “use latest version of the DITA DTD”:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE topic PUBLIC "-//OASIS//DTD DITA Topic//EN" "topic.dtd">
<topic id="MyTopic">
...
</topic>
```

This should not be a problem as DITA 1.3 is a superset of DITA 1.2.

Technical content only

Ditac only supports “Technical content elements”. However *Classification elements* (e.g. subject scheme maps) are still not supported.

DITA 1.3 RELAX NG schema

Ditac has no problem processing a DITA document pointing to a RELAX NG schema, rather than to a DTD or W3C XML Schema:

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-model href="urn:oasis:names:tc:dita:rng:topic.rng"?>
<topic id="MyTopic">
...
</topic>
```

The `<?xml-model?>` processing-instruction used in the above example is the standard way to associate a document to a RELAX NG schema. See *“Associating Schemas with XML documents 1.0”*.

The DTDToSchema facility

The `DTDToSchema` facility can be used to “upgrade” your documents conforming to a DITA 1.3 DTD to the equivalent DITA 1.3 W3C XML Schema or RELAX NG schema. Command-line example showing how to invoke the `DTDToSchema` facility:

```
$ java -cp ditac_install_dir/lib/ditac.jar com.xmlmind.ditac.tool.DTDToSchema-
-rng MyTopic.dita
```

Before invoking the `DTDToSchema` facility, `MyTopic.dita` contained:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE topic PUBLIC "-//OASIS//DTD DITA Topic//EN" "topic.dtd">
<topic id="MyTopic">
```

```
...  
</topic>
```

After invoking the `DTDTToSchema` facility, `MyTopic.dita` contains:

```
<?xml version="1.0" encoding="UTF-8"?>  
<?xml-model href="urn:oasis:names:tc:dita:rng:topic.rng"?>  
<topic id="MyTopic">  
...  
</topic>
```

The `DTDTToSchema` facility is auto-documented:

```
$ java -cp ditac_install_dir/lib/ditac.jar com.xmlmind.ditac.tool.DTDTToSchema
```

```
Usage: java -cp ditac.jar com.xmlmind.ditac.tool.DTDTToSchema  
-rng|-xsd [ in_dita_file|in_dir_containing_dita_files ]+
```

"Upgrades" specified DITA documents conforming to a standard DITA 1.3 DTD to the corresponding W3C XML schema or RELAX NG schema.

Processes files or directories. Files are modified in place. Directories are recursively processed. All the '.ditamap', '.dita' and '.ditaval' files found in specified directories are processed.

Options:

```
-rng Upgrade to RELAX NG schema.  
-xsd Upgrade to W3C XML schema.
```

Related information

- [Appendix B. Limitations and implementation specificities](#)

Appendix B. Limitations and implementation specificities

Conversion to XHTML and XSL-FO

- The following elements are ignored:
 - the `<syntaxdiagram>` element and all its descendant elements;
 - `<ux-window>`;
 - `<sort-as>`.
- The `<titlealts>/<navtitle>` element of topic is ignored.
- Layout of `<simpletable>` elements:
 - Attribute `@frame` is ignored.
 - Conversion to XHTML:
 - Attribute `@expanse` is partially supported. Its value is considered to always be 100%.
 - Conversion to XSL-FO:
 - Attribute `@expanse` is ignored. The width of a `<simpletable>` is always 100% and thus, you cannot center a `<simpletable>` using the `center` parameter.
- Layout of (CAL S) `<table>` element:
 - Attributes `<table>/@orient` and `<entry>/@rotate` are not supported.
 - Conversion to XHTML:
 - Attribute `@pgwide` is partially supported. Its value is considered to always be 100%.
 - Something like `colwidth="2*+3pt"` is treated as if it were `colwidth="2*"`. Moreover, because no Web browser seems to support relative lengths, a relative length is approximated to a percentage.
 - Conversion to XSL-FO:
 - Attribute `@pgwide` is ignored. The width of a `<table>` is always 100% and thus, you cannot center a `<table>` using the `center` parameter.
- The qualified ID of a descendant element of a topic is transformed as follows: `topicID/descendantID` becomes `topicID__descendantID` in the generated content. (The separator string being used comprises *two* underscore characters.)

Example: let's suppose a topic having "parameters" as its `@id` attribute, containing a table having "default_values" as its `@id` attribute, has been converted to HTML. The generated HTML file which contains the topic is called `userguide.html`.

- URL `"userguide.html#parameters"` allows to address the topic.
- URL `"userguide.html#parameters__default_values"` allows to address the table.

Booklists

- Contents corresponding to the following empty `<bookmap>` elements: `<toc>`, `<tablelist>`, `<figurelist>`, `<indexlist>` can be automatically generated by ditac.
- Ditac supports `<examplelist>` and `<equationlist>` in addition to `<toc>`, `<tablelist>`, `<figurelist>`, `<indexlist>`.

- Contents corresponding to the following empty `<bookmap>` elements: `<trademarklist>`, `<abbrevlist>`, `<bibliolist>`, `<glossarylist>` *cannot* be automatically generated by ditac.
- Entries automatically generated by ditac for `<toc>`, `<tablelist>`, `<figurelist>`, `<examplelist>`, `<equationlist>` and `<indexlist>` only contain plain text. For example, if a topic title is "`<title>The JavaTM Spring framework</title>`", then the corresponding TOC entry contains "The JavaTM Spring framework".
- About the automatically generated `<indexlist>`:
 - Specifying an `<indexterm>` element in the `<topicmeta>/<keywords>` element of a `<topicref>` element is equivalent to specifying it in the `<prolog>/<metadata>/<keywords>` element of the corresponding topic. Any other `<indexterm>` element found in a map is ignored.
 - In a topic, the implicit end of an index range is always after the last child of the topic, not including nested topics.
 - Overlapping index ranges are not supported.
 - The markup possibly contained in an `<indexterm>` (`<option>`, `<parname>`, `<apiname>`, etc) is ignored.
 - Because we consider this feature to be truly useful, we'll generate page references and "see also" redirections even for non-leaf index terms. No warnings will be reported in this case. If you don't like this specificity, simply do not author such `<indexterm>` elements.
 - Unless specified using the `-lang` command-line option, the language of the document is taken from the `@xml:lang` attribute of the root element of the topic map. If there is no such attribute, the language defaults to "en". Knowing the language of the document is required to be able to generate localized text (e.g. "Kapitel") and to sort and group the index entries.

Keyref processing

- Matching element content taken from a key definition is limited to the following cases:
 - A `<link>` element gets its `<linktext>` child from `key_definition/topicmeta/linktext` and its `<desc>` child from `key_definition/topicmeta/shortdesc`.
 - An `<xref>` element gets its contents from `key_definition/topicmeta/linktext`.
 - Elements `<ph>`, `<cite>`, `<keyword>`, `<dt>` and `<term>` all get their content from `key_definition/topicmeta/keywords/keyword`, if any. Otherwise the contents of `key_definition/topicmeta/linktext` is used as a fallback.
- Key-based, cross-deliverable addressing is not implemented.
- Topics which are not directly or indirectly referenced by the root map are automatically added to the root key scope. Such topics typically contain common content which is included by other topics using `@conref`.

If you don't want this to happen, please explicitly reference such common content topics in your maps and mark these references as being resource-only. Example:

```
<topigroup keyscope="MyKeycope">
  <topicref href="commonContent.dita" processing-role="resource-only"/>
```

Transclusion

- During a conref transclusion, ditac does not check the compatibility of the domains of the referencing document with the domains of the referenced document. This can be changed by defining system property `DITAC_CHECK_DOMAINS` (that is, adding `-DDITAC_CHECK_DOMAINS=1` to the `bin/ditac` shell

script or to `bin/ditac.bat`). However, the verifications performed by `ditac` are almost certainly not conforming as we have not really understood the spec.

- Transclusion does not implement automatic generalization. For example, transcluding `<li conref="foo.dita#foo/item3"/>` will report a fatal error if `"foo/item3"` is a `<step>` element.

A `<step>` element is a specialization of a `` element. Some DITA processors are capable of automatically converting a `<step>` element to an `` element. This is not the case of `ditac`.

Cascading of attributes and metadata

- Filtering and flagging may be performed using any attribute. However only the following attributes: `<audience>`, `<platform>`, `<product>`, `<otherprops>`, `<props>`, specializations of attributes `<props>` and `<rev>` properly cascade with a map, within the `<related-links>` element of a topic and from a `<topicref>` element to the referenced `<topic>` element.
- Both attribute (e.g. `@audience`) and element (e.g. `<audience>`) metadata are copied from a `<topicref>` element to the referenced `<topic>` element.
- Unless `topicref/topicmeta/@lockmeta=no`, `topicref/topicmeta/searchtitle` supplements or overrides `topic/titlealts/searchtitle`.
- In the following case, `<topicref href="foo.dita"/>`, the `<topicref>` metadata is copied only to the first topic found in `foo.dita`. An alternative would be to copy metadata to all topics found in `foo.dita`.

Conditional processing

- Conditional processing is also applied to the information (e.g. `<title>`, `<metadata>`) contained in a map. However, only the `exclude` action will work. The `flag` action does not work in this context.
- Any attribute (that is, not only `@audience`, `@platform`, `@product`, `@rev`, `@otherprops`, `@deliveryTarget` and attributes specialized from `@props`) may be used to filter or flag an element. For example, the `@status` attribute may be used to highlight changes. See [below](#).
- Subject scheme maps, which should be used to validate attribute values and also to implement smarter conditional processing, are currently ignored.
- If a map directly contains multiple `<ditavalref>` elements, all `<ditavalref>` elements but the first one are ignored. When this is the case, a warning is reported, though.
- The externally specified DITaval file is combined with the `<ditavalref>` element, if any, which is a direct child of a map.
- `<ditavalref>` error conditions are not detected.
- In a DITaval file, `action="passthrough"` is not supported.

Flagging contents

- Only the following elements (and, of course, their specializations) can be flagged *without restrictions*: `<topic>`, `<p>`, `<lq>`, `<note>`, `<dl>`, ``, ``, `<sl>`, `<pre>`, `<lines>`, `<fig>`, `<object>`, `<table>`, `<simpletable>`, `<section>`, `<example>`, `<ph>`, `<term>`, `<xref>`, `<cite>`, `<q>`, `<boolean>`, `<state>`, `<keyword>`, `<tm>`, `<image>`, `<foreign>`.

Any other element (``, `<dlentry>`, `<step>`, `<stentry>`, etc) is just given *some* of the colors and font styles, if any, specified by the flagging elements and attributes found in the `.ditaval` file.

- In a `.ditaval` file, attribute `style="double-underline"` is processed as if it were `underline`.

- In a .ditaval file, attribute `style="line-through"` is supported in addition to underline and overline.
- The `@status` attribute may be used to highlight changes. Example:

```
$ ditac -filter status.ditaval doc.pdf doc.ditamap
```

where file `status.ditaval` contains:

```
<val>
  <prop action="flag" att="status" bgcolor="#FFFF99" style="underline"
    val="new" />

  <prop action="flag" att="status" bgcolor="#99FF99" val="changed" />

  <prop action="flag" att="status" bgcolor="#FF7F7F" style="line-through"
    val="deleted" />
</val>
```

and where `doc.ditamap` references a topic containing for example:

```
<p>A paragraph containing <ph status="new">new text</ph>,
<ph status="changed">changed text</ph>,
<ph status="deleted">deleted text</ph>.</p>
...
<p status="new">New paragraph.</p>
...
<ul status="changed">
  <li>First item in changed <tt>ul</tt>.</li>
  <li><p>Second item.</p>
  <p status="deleted">Deleted paragraph.</p></li>
  <li>Third item.</li>
</ul>
```

- In a .ditaval file, the value of the `@changebar` attribute of the `<revprop>` element, has the following syntax:

```
changebar -> prop [ S ';' S prop ]+
```

```
prop -> prop_name ':' S prop_value
```

The style properties supported there are:

Name	Value	Default	Description
color	<code><color></code>	The value of the color property.	See XSL 1.1 property <code>change-bar-color</code> .
offset	<code><length></code>	6pt	See XSL 1.1 property <code>change-bar-offset</code> .
placement	start end left right inside outside alternate	start	See XSL 1.1 property <code>change-bar-placement</code> .

Name	Value	Default	Description
style	<code><border-style></code>	solid	See XSL 1.1 property <code>change-bar-style</code> .
width	<code><border-width></code>	medium	See XSL 1.1 property <code>change-bar-width</code> .

Example:

```
$ ditac -filter changebar.ditaval doc.pdf doc.ditamap
```

where file `changebar.ditaval` contains:

```
<val>
  <revprop action="flag" val="2.1"
    changebar="style: double; width: 3px; placement: start;" ></revprop>
</val>
```

and where `doc.ditamap` references a topic containing for example:

```
...
<fig rev="2.1">
  <title>The logo of ACME corp.</title>
  <image href="acme_logo.png" />
</fig>
...
```

- Change bars are implemented by only a few XSL-FO 1.1 processors, namely [RenderX XEP](#) and [Antenna House Formatter](#). For any other XSL-FO processor (e.g. [Apache FOP](#), [XMLmind XSL-FO Converter](#)) and also for all XHTML-based output formats (e.g. EPUB, Web Help), change bars are emulated using left or right borders. This emulation may give poor results when a change bar is added to a table.

Generating links

- Attribute `@collection-type`, whatever its value, is ignored inside the `<reltable>` element.
- Ditac cannot generate “smart labels” for related links. The label is always “Related Links”. It could have been “Related Concepts”, “Related Reference” or even something determined using what is specified in the `<title>` child element of a `<relcolspec>` element.

Chunking

- The “to-navigation” chunk value is ignored.
- When the `@copy-to` attribute is used to specify an URI, the parent path part (e.g. “foo” in “foo/bar.htm”) and the extension part (e.g. “.htm” in “foo/bar.htm”) are ignored. Only the “root name” (e.g. “bar” in “foo/bar.htm”) is taken into account during the processing of the map.
- The default chunking policy is `by-document`.
- When the deliverable targets a print media, all chunk specifications are removed and a `chunk="to-content"` attribute is added to the root element of the map.

Other limitations and specificities

- `<topicref>` elements found inside a `<reliable>` do not “pull” the corresponding topics. In other words, a `<reliable>` cannot be used to add some content to a deliverable. With `ditac`, a `<reliable>` is just used to create links between topics which are already part of the deliverable.
- There are several limitations and inconsistencies when working with files containing multiple topics and/or nested topics.

For example, let's suppose a map contains the following `<topicref>`s, where `multi.dita` contains multiple topics (first topic being `t1`, second topic being `t2`), each topic possibly containing nested topics.

```
<topicref href="multi.dita" />
<topicref href="multi.dita" />
<topicref href="multi.dita#t1" />
<topicref href="multi.dita#t2" />
```

- As expected, the first `<topicref>` pulls all the topics, including nested ones, contained in `multi.dita`. However parent, child, sibling, etc, related links will *not* be automatically generated for these topics.
- The second `<topicref>` pulls a copy of all the topics, including nested ones, contained in `multi.dita`. The third `<topicref>` pulls a copy of topic `t1` (excluding nested topics). The fourth `<topicref>` is *not* detected as pulling a copy of topic `t2`. Therefore the fourth `<topicref>` does nothing at all, as topic `t2` has already been pulled into the deliverable (by the first `<topicref>`).
- The following `<topicref>` elements are not treated differently from the others: `<topicset>`, `<topicsetref>`.
- The following `<bookmap>` elements: `<abbrevlist>`, `<amendments>`, `<appendices>`, `<appendix>`, `<bibliolist>`, `<bookabstract>`, `<booklist>`, `<chapter>`, `<colophon>`, `<dedication>`, `<draftintro>`, `<figurelist>`, `<examplelist>`, `<equationlist>`, `<glossarylist>`, `<indexlist>`, `<notices>`, `<part>`, `<preface>`, `<tablelist>`, `<toc>`, `<trademarklist>`, are considered to have an *implicit title* when
 - they have no `@href` attribute,
 - and they have no explicit title,
 - and they contain one or more `<topicref>` (of any type) child elements.

For example:

```
<glossarylist>
  <topicref href="term1.dita" />
  <topicref href="term2.dita" />
  <topicref href="term3.dita" />
</glossarylist>
```

is processed as if it was:

```
<glossarylist navtitle="Glossary">
  <topicref href="term1.dita" />
  <topicref href="term2.dita" />
  <topicref href="term3.dita" />
</glossarylist>
```

- All attributes and elements — `map/@anchorref`, `<anchorref>`, `<anchor>`, `<navref>` — related to runtime integration of maps are ignored.
- Ditac reports a "*topicB*, href points outside processed topics" warning when *topicA* references *topicB* and *topicB* is not referenced in the map. In order to suppress this warning, add to the map a `<topicref>` having attribute `toc="no"` and pointing to *topicB*.
- Convenience element `<glossref>` cannot be used with ditac without setting some of its attributes. Example:

```
<glossref href="ONE.dita" keys="key_ONE" />
```

is strictly equivalent to:

```
<topicref href="ONE.dita" keys="key_ONE" linking="none" print="no"
  toc="no" search="no" />
```

Notice default attribute `print="no"`. Therefore, when generating PDF, such `<glossref>` is discarded at a very early stage by ditac. The consequence is that each occurrence of `<abbreviated-form keyref="key_ONE" />` will cause ditac to report a "cannot resolve keyref" warning. The workaround is to simply avoid using `<glossref>` and to stick to `<topicref>` with a `@keys` attribute.

Related information

- [Appendix A. About DITA support in XMLmind DITA Converter](#)

Appendix C. Translating the messages generated by ditac

About this task

The messages generated by ditac (**Table of Contents**, **List of Figures**, **Chapter**, **Appendix**, etc) are available in English (*en*), French (*fr*), German (*de*), Spanish (*es*), etc. Now let's suppose that you are routinely authoring Portuguese documents and that you want ditac to also support this language.

Procedure

1. Go to the `ditac_install_dir/xsl/common/messages/` directory.

```
~$ cd /opt/ditac/xsl/common/messages/
```

2. Copy `en.xml` to `pt.xml`.

Note that "pt" is the ISO 639-1 two-letter code of the Portuguese language.

Country variants of a language are supported too. Example: `pt-BR` (Brazilian Portuguese). However, when this is the case, make sure that the name of the file containing your messages use lower-case characters. Example: `pt-br.xml` should be fine, while `pt-BR.xml` or `pt_br.xml` would not work.

```
/opt/ditac/xsl/common/messages$ cp en.xml pt.xml
```

3. Open `pt.xml` in a text or XML editor and translate to Portuguese all the messages found in this file.

```
<?xml version="1.0" encoding="UTF-8"?>
<messages xml:lang="pt">
  <!-- Task sections -->
  <message name="prereq">Pré-requisito</message>
  ...
```

For some languages, like CJK languages, you'll have to insert variable `%{N}` in the localized text corresponding to numbered elements (Chapter, Appendix, Table, Figure, etc). This variable is replaced by the number of the element.

Japanese example: excerpts of a possible `ditac_install_dir/xsl/common/messages/ja.xml`:

```
<message name="chapter">#%{N}#</message>
```

For the first chapter of the document, this gives "#1#", which means "The 1 Chapter".

4. Open `ditac_install_dir/xsl/common/commonUtil.xsl` in a text or XML editor and add string 'pt' at the end of the `messageFileNames` list:

```
<!-- localize =====>
<xsl:param name="messageFileNames" select="'en', 'fr', 'de', 'es', 'pt'"/>
```

5. When done, please send us (ditac-support@xmlmind.com) your translation (e.g. `pt.xml`) so we can add to the distribution of XMLmind DITA Converter.

Index

A

- addindex, option, 16
- add-index-toc, parameter, 28
- add-toc-root, parameter, 33
- ahf, option, 17
- AHF, XSL-FO processor, 7, 17, 93
- animation, 61
- Antenna House Formatter. *See* AHF, XSL-FO processor
- Apache FOP. *See* FOP, XSL-FO processor
- appendix-number-format, parameter, 22
- audio, 60
- automap, option, 18

B

- backmatter, option, 16
- base-font-size, parameter, 40
- body-bottom-margin, parameter, 40
- body-font-family, parameter, 40
- body-start-indent, parameter, 40
- body-top-margin, parameter, 41
- {{break}}, page header/footer variable, 51

C

- c, option, 14
- cause-number-format, parameter, 22
- center, parameter, 22
- chain-pages, parameter, 28
- chain-topics, parameter, 29. *See also* ignore-navigation-links, parameter
- changes
 - bars, 92
 - highlighting, 92
- {{chapter-title}}, page header/footer variable, 50
- .chm, filename extension, 7. *See also* HTML Help, output format
- chmBasename, parameter, 38
- choice-bullets, parameter, 41
- chunk, option, 14
- cover-image, parameter, 39
- CSS, custom, 66
- css, parameter, 29, 67
- css-name, parameter, 29

D

- default-table-width, parameter, 30
- ditac.options, options file, 3, 18

- ditacListsURI, parameter, 22
- DITA specialization, 69
- DITAVAL, conditional processing, 15, 91, 92
- .doc, filename extension, 7. *See also* RTF, output format
- {{document-date}}, page header/footer variable, 50
- {{document-title}}, page header/footer variable, 50
- .docx, filename extension, 7, 26. *See also* Office Open XML, output format
- docx, output format name. *See* Office Open XML, output format
- dryrun, option, 18

E

- Eclipse Help, output format, 5, 9, 12, 20, 66
- eclipsehelp, output format name. *See* Eclipse Help, output format
- .epub, filename extension, 7. *See also* EPUB 2, output format
- epub, output format name. *See* EPUB 2, output format
- epub:trigger, 63
- EPUB 2, output format, 5, 7, 9, 12, 20, 23, 66, 66
- epub2-compatible, parameter, 40
- EPUB 3, output format, 5, 9, 13, 13, 20, 23, 40, 40, 40, 58, 59, 63, 63, 66
- epub3, output format name. *See* EPUB 3, output format
- epub-identifier, parameter, 40
- equation-block-equation-width, parameter, 41
- equation-block-number-width, parameter, 41
- equation-number-after, parameter, 22
- equation-number-before, parameter, 23
- errout, option, 18
- extended-toc, parameter, 23
- external-href-after, parameter, 41
- external-href-before, parameter, 41
- external-link-icon-height, parameter, 30
- external-link-icon-name, parameter, 30
- external-link-icon-width, parameter, 30
- external-resource-base, parameter, 23

F

- f, option, 15
- filter, option, 15, 92, 93
- Flash, 61
- fo, output format name. *See* XSL-FO, output format

-foconverter, option, 17
 footer-center, parameter, 41
 footer-center-width, parameter, 42
 footer-height, parameter, 42
 footer-left, parameter, 42
 footer-left-width, parameter, 42
 footer-right, parameter, 42
 footer-right-width, parameter, 42
 footer-separator, parameter, 42
 -fop, option, 17
 FOP, XSL-FO processor, 4, 7, 7, 17, 93
 foProcessor, parameter, 41
 -format, option, 15
 format-to-type, parameter, 30
 -frontmatter, option, 15

G

Generated messages. *See* Translation
 generate-epub-trigger, parameter, 40
 generator-info, parameter, 30

H

header-center, parameter, 42
 header-center-width, parameter, 43
 header-height, parameter, 43
 header-left, parameter, 43
 header-left-width, parameter, 43
 header-right, parameter, 43
 header-right-width, parameter, 43
 helpset-basename, parameter, 38
 -hhc, option, 17
 hhc-basename, parameter, 38
 hhpBasename, parameter, 38
 hhp-template, parameter, 38
 hhx-basename, parameter, 39
 highlight-source, parameter, 23
 .htm, filename extension, 7. *See also* XHTML
 1.0, output format
 .html, filename extension, 7. *See also* XHTML
 1.0, output format
 html, output format name. *See* HTML 4.01,
 output format
 HTML 4.01, output format, 5, 9, 9, 20, 66
 HTML Help, output format, 5, 7, 9, 11, 20, 66,
 66
 htmlhelp, output format name. *See* HTML
 Help, output format
 hyphenate, parameter, 43

I

-i, option, 15

ignore-navigation-links, parameter, 31. *See
 also* chain-topics, parameter
 -ignoreoptionsfile, option, 18
 {{image(URI)}}, page header/footer variable,
 51
 imagemap, 46
 -images, option, 15
 -index, option, 15
 index-basename, parameter, 38
 index-column-gap, parameter, 43
 indexer-directory-basename, parameter, 38
 index-range-separator, parameter, 23

J

Java Help, output format, 5, 7, 10, 11, 20, 66,
 66
 javahelp, output format name. *See* Java Help,
 output format
 javascripts, parameter, 31
 jhindexer, 38
 -jhindexer, option, 17
 justified, parameter, 43

K

-keepfo, option, 18

L

-lang, option, 16
 link-auto-text, parameter, 24
 link-bullet, parameter, 43
 Localization. *See* Translation

M

map-basename, parameter, 38
 mark-external-links, parameter, 32
 mark-important-steps, parameter, 24
 MathJax, 31, 32
 mathjax, parameter, 31
 mathjax-url, parameter, 32
 MathML, 31, 32, 59
 menucascade-separator, parameter, 44

N

navigation-icon-height, parameter, 32
 navigation-icon-suffix, parameter, 32
 navigation-icon-width, parameter, 32
 note-icon-height, parameter, 44
 note-icon-list, parameter, 24
 note-icon-suffix, parameter, 44
 note-icon-width, parameter, 44

number, parameter, 8, 24
 number-separator1, parameter, 24
 number-separator2, parameter, 24
 number-toc-entries, parameter, 33

O

-o, option, 16
 .odt, filename extension, 7, 27. *See also* OpenOffice, output format
 OpenOffice, output format
 odt, output format name. *See* OpenOffice, output format
 Office Open XML, output format, 4, 7, 8, 10, 20, 67
 onclick, processing-instruction, 63
 OpenOffice, output format, 4, 7, 8, 10, 20, 67
 -options, option, 16
 ordered list, numbering, 54
 out of memory error, 19
 outputclass, attribute, 23

P

-p, option, 14
 page-bottom-margin, parameter, 44
 {{page-count}}, page header/footer variable, 50
 page-height, parameter, 44
 page-inner-margin, parameter, 45
 {{page-number}}, page header/footer variable, 50
 page-orientation, parameter, 45
 page-outer-margin, parameter, 45
 page-ref-after, parameter, 45
 page-ref-before, parameter, 45
 {{page-sequence}}, page header/footer variable, 51, 52
 page-top-margin, parameter, 45
 page-width, parameter, 45
 paper-type, parameter, 45
 -param, option, 14
 Parameters. *See* XSLT stylesheets parameters
 part-number-format, parameter, 25
 .pdf, filename extension, 7, 7. *See also* Java Help, output format; PDF, output format
 PDF, output format, 4, 7, 7, 10, 20, 67
 pdf, output format name. *See* PDF, output format
 pdf-outline, parameter, 46
 -plugin, option, 18
 plugin-id, parameter, 39
 plugin-index-basename, 39
 plugin-name, parameter, 39

plugin-provider, parameter, 39
 plugin-toc-basename, parameter, 39
 plugin-version, parameter, 39
 PostScript, output format, 4, 7, 7, 10, 20, 67
 prepend-chapter-to-section-number, parameter, 25
 -preprocess, option, 18, 72
 .ps, filename extension, 7. *See also* PostScript, output format
 ps, output format name. *See* PostScript, output format

R

-r, option, 15
 remedy-number-format, parameter, 25
 RenderX XEP. *See* XEP, XSL-FO processor
 -resourcehandler, option, 15
 -resources, option, 15
 .rtf, filename extension, 7, 26. *See also* RTF, output format
 RTF, output format, 4, 7, 8, 10, 20, 67
 rtf, output format name. *See* RTF, output format

S

screen-resolution, parameter, 32
 {{section1-title}}, page header/footer variable, 50
 show-draft-comments, parameter, 25
 show-external-links, parameter, 46
 show-imagemap-links, parameter, 46
 show-link-page, parameter, 46
 show-xref-page, parameter, 46
 SVG, 58
 .swf, 61
 syntax highlighting, 23, 56

T

-t, option, 14, 68
 table, background color, 55
 text-file-encoding, parameter, 25
 title-after, parameter, 25
 title-color, parameter, 47
 title-font-family, parameter, 47
 title-page, parameter, 25
 title-prefix-separator1, parameter, 26
 -toc, option, 8, 15
 toc-basename, parameter, 38
 {{topic-title}}, page header/footer variable, 50, 51

Translation, 96
 troubleSolution-number-format, parameter, 25
 two-sided, parameter, 47

U

ul-li-bullets, parameter, 47
 unordered-step-bullets, parameter, 47
 use-multimedia-extensions, parameter, 47
 use-note-icon, parameter, 26

V

-v, option, 8, 16
 -validate, option, 18
 -version, option, 18
 video, 60, 62
 -vv, option, 16
 -vvv, option, 16

W

watermark, parameter, 48
 watermark-image, parameter, 26
 Web Help, output format, 5, 5, 9, 10, 20, 20, 66
 webhelp, output format name. *See* Web Help, output format
 Web Help 5, output format, 9, 11, 58, 59, 63
 webhelp5, output format name. *See* Web Help 5, output format
 whc-index-basename, parameter, 37
 wh-collapse-toc, parameter, 34
 whc-toc-basename, parameter, 37
 wh-index-numbers, parameter, 34
 wh-jquery, parameter, 34
 wh-jquery-css, parameter, 34
 wh-jquery-custom-theme, parameter, 34
 wh-jquery-theme, parameter, 34
 wh-jquery-ui, parameter, 34
 wh-layout, parameter, 35
 wh-local-jquery, parameter, 35
 wh-user-css, parameter, 35
 wh-user-footer, parameter, 36
 wh-user-header, parameter, 36
 wh-user-resources, parameter, 37
 wh-use-stemming, parameter, 35
 .wml, filename extension, 7, 26. *See also*
 WordprocessingML, output format
 wml, output format name. *See*
 WordprocessingML, output format
 WordprocessingML, output format, 4, 7, 8, 10, 20, 67

X

-xep, option, 17
 XEP, XSL-FO processor, 7, 17, 19, 93
 -xfc, option, 17
 XFC, XSL-FO Converter, 8
 XFC, XSL-FO processor, 8, 17, 27, 93
 .xhtml, filename extension, 7. *See also*
 XHTML 1.0, output format
 xhtml, output format name. *See* XHTML 1.0, output format
 XHTML 1.0, output format, 4, 7, 8, 9, 20, 66
 XHTML 1.1, output format, 4, 9, 9, 20, 66
 xhtml1.1, output format name. *See* XHTML 1.1, output format
 XHTML 5, output format, 5, 9, 9, 20, 58, 59, 63, 66
 xhtml5, output format name. *See* XHTML 5, output format
 xhtml-mime-type, parameter, 32
 .xml, filename extension, 7. *See also*
 WordprocessingML, output format
 XML catalog, 69
 XMLmind XSL-FO Converter. *See* XFC, XSL-FO processor
 -Xmx, Java option, 19
 xref-auto-text, parameter, 27
 XSL-FO, output format, 10
 xsl-resources-directory, parameter, 27
 -xslt, option, 14
 XSLT stylesheets parameters, 22

Y

YouTube, 62