
XMLmind XSL Utility - Online Help

Hussein Shafie, XMLmind Software <xfc-support+xmlmind.com>

December 27, 2024

Table of Contents

| | |
|---|----|
| 1. Overview | 1 |
| 2. Running XMLmind XSL Utility | 3 |
| 2.1. System requirements | 3 |
| 2.2. Installation | 3 |
| 2.3. Contents of the installation directory | 3 |
| 2.4. Starting XMLmind XSL Utility | 4 |
| 2.5. XMLmind XSL Utility as a command-line tool | 5 |
| 3. Converting an XML document to another format | 6 |
| 3.1. Canceling the current conversion process | 8 |
| 4. Changing the parameters of a conversion | 9 |
| 5. Specifying a conversion | 11 |
| 5.1. Specifying the conversion of a DITA topic or map | 16 |
| 5.1.1. Reference of the fields found in the DITA tab | 16 |
| 5.2. Specifying the conversion of an XHTML page | 18 |
| 5.3. Specifying the conversion of DocBook v5.1+ assembly | 20 |
| 5.4. Customizing a stock XSLT stylesheet | 21 |
| 5.4.1. Using an existing XSLT stylesheet customization | 21 |
| 5.4.2. Creating an XSLT stylesheet customization | 21 |
| 5.4.3. The "XMLmind XSL Customizer" application | 22 |
| 6. User preferences | 29 |
| 6.1. The <code>-p</code> command-line option | 30 |
| 6.2. General preferences | 30 |
| 6.3. Helper applications preferences | 31 |
| 6.3.1. The "Helper Application Editor" dialog box | 32 |
| 6.4. FOP preferences | 35 |
| 6.5. XEP preferences | 37 |
| A. Variables | 38 |
| B. XSL-FO processor parameters | 39 |
| 1. XMLmind XSL-FO Converter (XFC) | 39 |
| 2. Apache FOP | 42 |
| 3. RenderX XEP | 42 |
| C. Using Antenna House Formatter as your XSL-FO processor | 42 |
| D. Using XMLmind XSL Utility to convert a DocBook 4 document to HTML Help (.chm file) | 44 |
| E. Support of XInclude in XMLmind XSL Utility | 47 |

1. Overview



About Evaluation Edition

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!

XMLmind XSL Utility is a graphical application (see screenshot [7]) written in Java™ integrating two XSLT engines:

- Saxon 6, an XSLT 1 transformation engine,
- Saxon 12, an XSLT 3 transformation engine,

and three XSL-FO processors:

- XMLmind XSL-FO Converter (XFC for short), converts XSL-FO to RTF (Word 2000+), WordprocessingML (Word 2003+), Office Open XML (.docx, Word 2007+) and OpenOffice (.odt, OpenOffice.org 2+),
- Apache FOP, renders XSL-FO as PDF and PostScript®,
- RenderX XEP, renders XSL-FO as PDF and PostScript®. (More information about the integration of RenderX XEP in XMLmind XSL Utility [2].)

Out of the box, it allows to convert DocBook 4.x, 5.0, 5.1 and 5.2 including assemblies and XHTML documents to PDF, RTF (can be opened in Word 2000+), WordprocessingML (can be opened in Word 2003+), Office Open XML (.docx, can be opened in Word 2007+) and OpenOffice (.odt, can be opened in OpenOffice/LibreOffice 2+) formats. DITA 1.0, 1.1, 1.2 and 1.3 documents can be converted to even more formats: XHTML 1.0, XHTML 1.1, HTML 4.01, XHTML 5, Web Help, Java™ Help, HTML Help, Eclipse Help, EPUB 2, EPUB 3, PDF, RTF, WordprocessingML, Office Open XML, OpenDocument.

A dialog box allows to modify the specifications of existing conversions (example: change the `paper.type` parameter of the XSLT stylesheet from `A4` to `USletter`) or to add more conversion specifications (example 1: convert DocBook 4 to PostScript® using FOP; example 2: convert TEI to PDF using PassiveTeX).

This dialog box also allows to specify the command (Windows example: `start "" "%0"`) which is to be used to preview the result of the conversion.

A conversion is basically a two-step process¹:

1. Transform the XML input document using the XSLT engine.
2. Process the temporary files created by first step to generate the desired output format.

Because step #1 is optional, you may use XMLmind XSL Utility to transform XSL-FO files (generated by your external tool chain) to PDF, PostScript®, RTF, WordprocessingML, Office Open XML and OpenOffice formats.

Because step #2 is optional, you may use XMLmind XSL Utility to transform your XML documents to formats such as HTML or Eclipse Help.

Because step #2 can be performed using an external command (HTML Help example: `hhc.exe`) rather than an XSL-FO processor, you may use XMLmind XSL Utility to transform your XML documents to formats such as HTML Help (.chm) or Java™ Help (.jar). Appendix D, *Using XMLmind XSL Utility to convert a DocBook 4 document to HTML Help (.chm file)* [44] explains how to do this by taking the HTML Help format (.chm) as an example.



About the integration of RenderX XEP in XMLmind XSL Utility

Unlike XMLmind XSL-FO Converter and Apache FOP, RenderX XEP is just *pre-installed* in XMLmind XSL Utility.

¹Conversion of DITA documents works differently.

The first time you'll try to use this commercial XSL-FO processor, XMLmind XSL Utility will prompt you for the directory where you have installed the RenderX product. This directory must contain RenderX XEP (`xep`, `xep.xml`, `lib/xep.jar`, etc) as well as a valid licence file (`license.xml`).

If you didn't purchase RenderX XEP and wants to give it a try (highly recommended), close the dialog box allowing to finish the installation of XEP in XMLmind XSL Utility, quit XMLmind XSL Utility, then download and install RenderX XEP Trial Edition or RenderX XEP Personal Edition (free to use under certain conditions).

2. Running XMLmind XSL Utility

2.1. System requirements

A Java™ 1.8+ runtime is required to run the XMLmind XSL Utility. Both Oracle Java and OpenJDK are officially supported.

XMLmind XSL Utility is officially supported is officially supported on Windows 7/8/10/11 (32-bit or 64-bit), on Linux and on macOS (Intel® or Apple® Silicon processor) 14.x (Sonoma) and on macOS 15.x (Sequoia).

2.2. Installation

Simply unzip the distribution somewhere. Linux/macOS example:

```
~$ cd /opt
/opt$ unzip /tmp/xslutil-6_5_0.zip
/opt$ ls xslutil-6_5_0
addon/
bin/
doc/
legal.txt
legal/
```

This means that uninstalling XMLmind XSL Utility simply consists in deleting the directory created by unzipping its distribution.

2.3. Contents of the installation directory

addon/

Contains XMLmind XML Editor configurations (DITA 1.3, DocBook 4.x, DocBook 5.0, DocBook 5.1+, XHTML) and plug-ins (FOP, Batik, JEuclid, XEP, XFC).

bin/xslutil.exe, xslutil-c.bat

Executable file and .bat file used to run XMLmind XSL Utility on Windows. More information about `xslutil-c.bat` in Section 2.5, “XMLmind XSL Utility as a command-line tool” [5].

bin/xslutil

Shell script used to run XMLmind XSL Utility on the Mac and on Linux.

bin/*.jar

All the (non-system) Java™ class libraries needed to run XMLmind XSL Utility.

bin/icon/

Contains desktop icons for XMLmind XSL Utility.

doc/index.html

Points to copies of this online help in HTML, PDF, RTF, WordprocessingML, Office Open XML and OpenOffice formats.

legal.txt, legal/

Contains XMLmind XSL Utility licenses as well as the licenses and notices attached to the software components used to build XMLmind XSL Utility.

2.4. Starting XMLmind XSL Utility

XMLmind XSL Utility is intended to be used directly from the directory created by unzipping its distribution. That is, you can start XMLmind XSL Utility by typing the following command in a command prompt and then, by pressing Enter:

```
C:\> xslutil-6_5_0\bin\xslutil
```

After testing that it works, you may want to add a shortcut to C:\xslutil-6_5_0\bin\xslutil.exe on your desktop.

On the Mac and on Linux, please type the following command in a terminal, then press Enter:

```
/opt$ xslutil-6_5_0/bin/xslutil &
```

Note that it's possible to specify the file to be converted as a command-line argument. Linux example:

```
/opt$ xslutil-6_5_0/bin/xslutil userguide/doc.ditamap &
```

In such case, when the file extension of the input file is well known (dita, ditamap, html, xhtml, fo, etc), the corresponding conversion to DOCX is automatically selected as well as an output file having the same name and directory as the input file, but with a "docx" extension. In the case of the above example, the "ditaToDocx" conversion is automatically selected with a "userguide/doc.docx" output file.



Running XMLmind XSL Utility on a computer having a very high resolution (HiDPI) screen

XMLmind XSL Utility works fine on computers having *very high resolution* (HiDPI) screens. For example, it works fine on a Mac having a Retina® screen and a Windows computer having an UHD (“4K”) screen.

However, on some Linux computers having HiDPI screens, HiDPI is not automatically detected. You'll have to specify the display scaling factor you prefer using the `-putpref` command-line option:

```
xslutil -putpref displayScaling 200
```

Preference key `displayScaling` may be used to globally change the size of all the items comprising the user interface of XMLmind XSL Utility. Its value is a percentage between 100 and 400 or integer -1 which means use system settings.

Note that using option `-putpref` updates the user preferences file. Therefore suffice to specify `-putpref` once and you are done.



FlatLAF as the default Look&Feel on Linux

On Linux, **FlatLAF** and its light theme (called "FlatLight") is now used as the default Look & Feel. This is needed because on Linux, the "system" Look & Feel (called "Metal") looks rather outdated.

If, for any reason, you prefer to use the "system" Look & Feel, please start **xslutil** by running

```
xslutil -putpref lookAndFeelClassName fallback
```

This setting is done once for all. If after doing that, you finally prefer to revert to **FlatLaf**, simply run

```
xslutil -delpref lookAndFeelClassName
```

The **FlatLAF** Look&Feel is available on all platforms. For example, if you prefer a dark theme on your Mac, simply start **xslutil** as follows.

```
xslutil -putpref lookAndFeelClassName com.formdev.flatlaf.themes.FlatMacDarkLaf
```

As explained above, this setting is done once for all. No need to specify `-putpref lookAndFeelClassName LAF_class_name` after that.

| Description | Value of <i>LAF_class_name</i> |
|---|---|
| FlatLaf Light | <code>com.formdev.flatlaf.FlatLightLaf</code> |
| FlatLaf Dark | <code>com.formdev.flatlaf.FlatDarkLaf</code> |
| FlatLaf IntelliJ (based on FlatLaf Light) | <code>com.formdev.flatlaf.FlatIntelliJLaf</code> |
| FlatLaf Darcula (based on FlatLaf Dark) | <code>com.formdev.flatlaf.FlatDarculaLaf</code> |
| FlatLaf macOS Light | <code>com.formdev.flatlaf.themes.FlatMacLightLaf</code> |
| FlatLaf macOS Dark | <code>com.formdev.flatlaf.themes.FlatMacDarkLaf</code> |
| "System" Look & Feel | <code>fallback</code> |
| Default Look & Feel | <code>default</code> |

2.5. XMLmind XSL Utility as a command-line tool

XMLmind XSL Utility may also be used a command-line tool.

- Without any command-line arguments², XMLmind XSL Utility starts as a desktop application.

²Or just with the file to be converted as the *single* command-line argument of XMLmind XSL Utility.

- If you pass it the following command-line arguments, XMLmind XSL Utility will perform the conversion without displaying its main window:

```
xslutil conversion_specification_name input_xml_file output_file_or_directory
```

Windows example corresponding to the figure below [7]:

```
C:\xslutil-6_5_0\bin> xslutil-c dbToDocx E:\tmp\help.xml E:\tmp\help.docx
```



On Windows, make sure to use `xslutil-c.bat` and not `xslutil.exe`.

Linux/Mac example:

```
/opt/xslutil-6_5_0/bin$ xslutil dbToDocx /tmp/help.xml /tmp/help.docx
```

The basic idea here is to use the dialog box to add or edit conversion specifications and then to use the XMLmind XSL Utility command-line to actually perform the conversion. This way you get the best of both worlds.

See also Section 6.1, “The `-p` command-line option” [30].

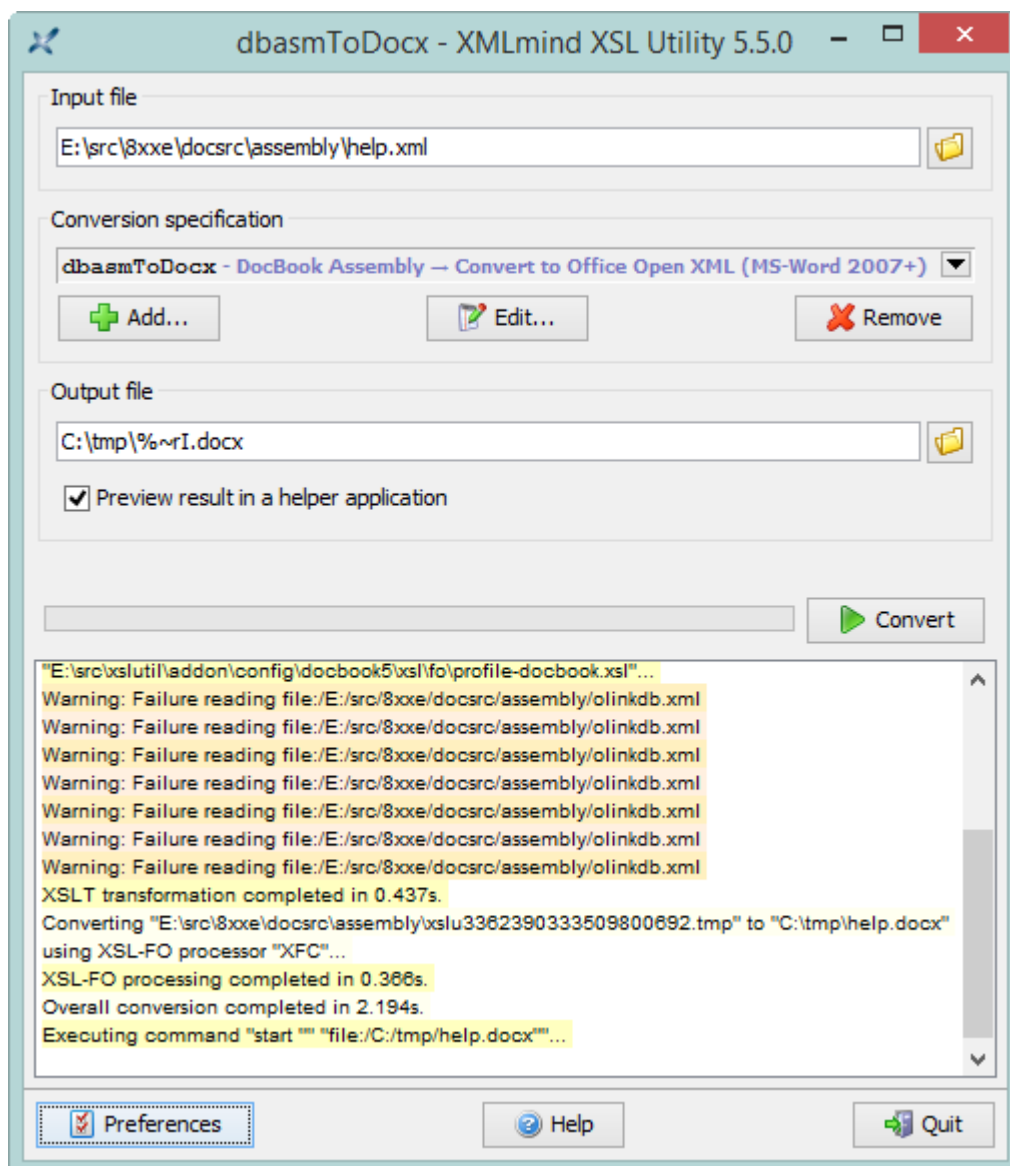
3. Converting an XML document to another format



About Evaluation Edition

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!

Figure 1. The main window of XMLmind XSL Utility



Procedure 1. How to convert an XML document to another format

1. Specify the XML document to be converted in the **Input file text** field.

The button next to the text field allows you to choose this XML document using the standard file chooser dialog box.

2. Choose the conversion using the **Conversion specification** combobox.



Make sure to choose the conversion which matches the document type of your input file. For example, make sure to choose `dbToRTF` if you want to convert a DocBook 4.x document to RTF. Do not choose `db5ToRTF` because this will give unexpected results.



DocBook 5.1+ support

Note that `db5Toxxx` conversions work for DocBook 5.0 input files as well as for DocBook 5.1+ input files. However if the DocBook 5.1+ input file contains an *assembly* (i.e. not a book, chapter, article, etc), then you must use `dbasmToxxx` conversions instead.



Do not hesitate to remove all the conversion specifications you don't need. This will unclutter the popup menu displayed by **Conversion specification** combobox. In order to do this, select the unwanted conversion and then click the **Remove** button.

3. Specify the output file or directory in the **Output file** text field.

The button next to the text field allows you to choose a save file or a save directory, depending on the conversion selected in step #2. Examples: generating a PDF file requires you to specify a save file; generating Eclipse Help requires you to specify a save directory.



The filename specified in **Output file** text field may reference the `%I` variable. (Appendix A, *Variables* [38] describes all the variables supported by XMLmind XSL Utility.)

For example, let's suppose the **Output file** text field contains `%~pI/rtf/%~rI.rtf`. When you'll convert `/home/john/docs/manual.xml` to RTF, the generated file will be found in `/home/john/docs/rtf/manual.rtf`. When you'll convert `/home/john/docs/primer.xml` to RTF, the generated file will be found in `/home/john/docs/rtf/primer.rtf`.

4. Optionally, check the "**Preview result in a helper application**" checkbox if you want to preview the result of the conversion in a helper application such as Acrobat Reader, Microsoft Word, OpenOffice/LibreOffice, etc. Such helper applications are specified once for all using **Preferences**, "**Helper Applications**" section [31].
5. Click the **Convert** button.

3.1. Canceling the current conversion process

After the conversion process is started, the **Convert** button becomes the **Cancel** button and allows you to stop the conversion.

A conversion can be stopped only after the current conversion step (example: the XSLT transformation step) is complete, therefore clicking the **Cancel** button generally has not an immediate effect.

If you think that the current conversion step has entered an endless loop, click the **Cancel** button while pressing the Shift key. This will abruptly stop the overall conversion process. After doing that, it is recommended to restart XMLmind XSL Utility as the application may become unstable.

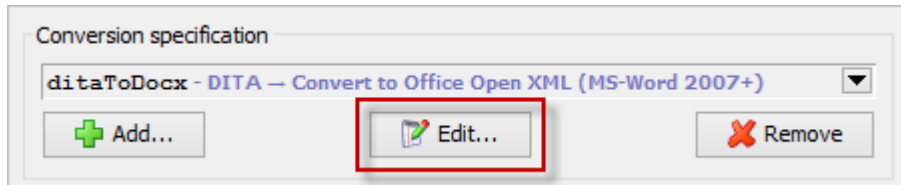
4. Changing the parameters of a conversion

Almost all conversions leverage an XSLT transformation, also called *an XSLT stylesheet*. This XSLT stylesheet almost always supports a large number of parameters. For example, some of such parameters are documented in the following documents:

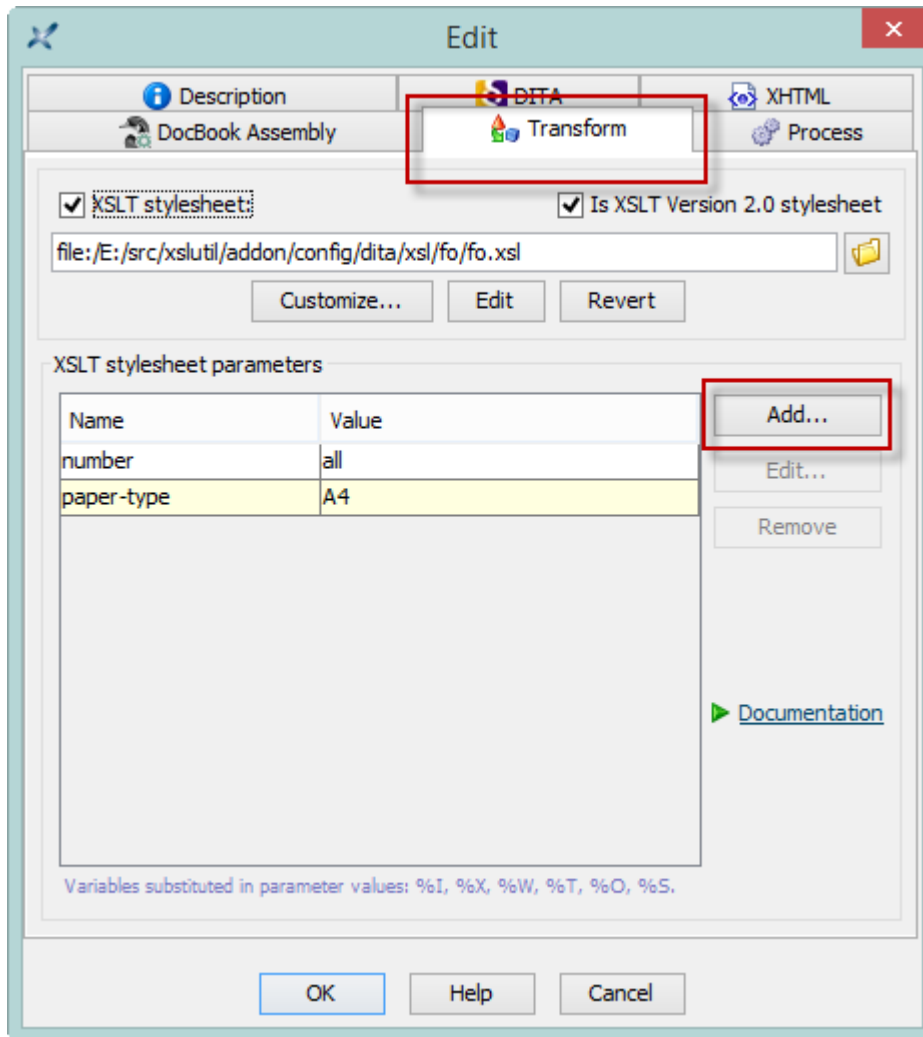
| Input document | Documentation of the XSLT stylesheet parameters |
|----------------|--|
| DITA | XMLmind DITA Converter Manual, XSLT stylesheets parameters |
| DocBook | DocBook XSL Stylesheet Reference Documentation by Norman Walsh |
| XHTML | Parameters of the XSLT stylesheets used to convert XHTML to XSL-FO |

Procedure 2. How to add an XSLT stylesheet parameter

1. Select the conversion you want to parameterize. Let's suppose you have selected `ditaToDocx`.
2. Click **Edit**.



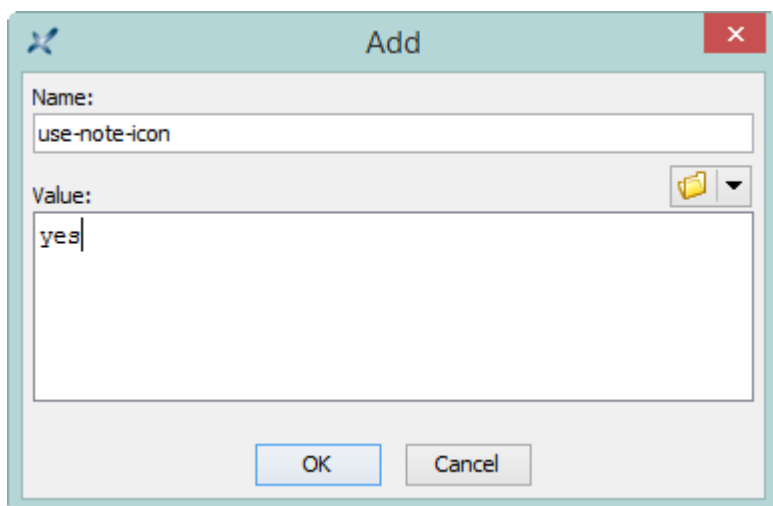
3. Click the **Transform** tab.



Parameter "paper-type" is set by default to "A4". If you want to change it to "Letter", suffice to select the table row containing "paper-type", click **Edit** and then type the parameter name and its value using the same dialog box as the one shown below.

Now let's suppose to want to *add* parameter use-note-icon=yes.

4. Click **Add**.
5. Type the parameter name and its value.

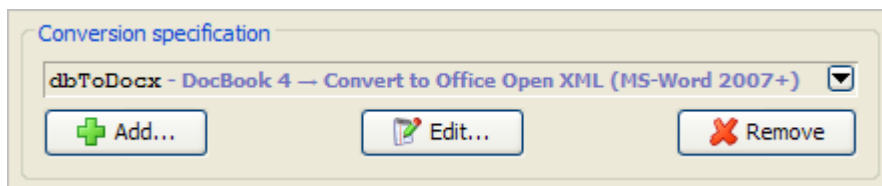


6. Click **OK** twice to close the two dialog boxes.

Note that conversion `ditaToDocx` has been modified once for all. You'll not have to add parameter `use-note-icon=yes` the next time you'll use XMLmind XSL Utility.

5. Specifying a conversion

Figure 2. Buttons allowing to add, edit and remove conversion specifications



Procedure 3. How to specify a conversion

1. Click the **Add** button.



You may also select a conversion close to the one you intend to add and then click the **Edit** button to modify it. In such case, do not forget to change the name of the edited conversion specification. By doing so, you'll add a new conversion rather than modify the existing one.

2. Give a name to your conversion specification and specify whether it creates a file or a directory.

The screenshot shows the 'Edit' dialog box with the following fields and options:

- Name:** ditaToDocx
- Description:** Convert to Office Open XML (MS-Word 2007+)
- Icon:** icon:microsoft
- Category:** DITA
- Category icon:** icon:oasis
- Output type:** File, Directory
- Extension:** docx

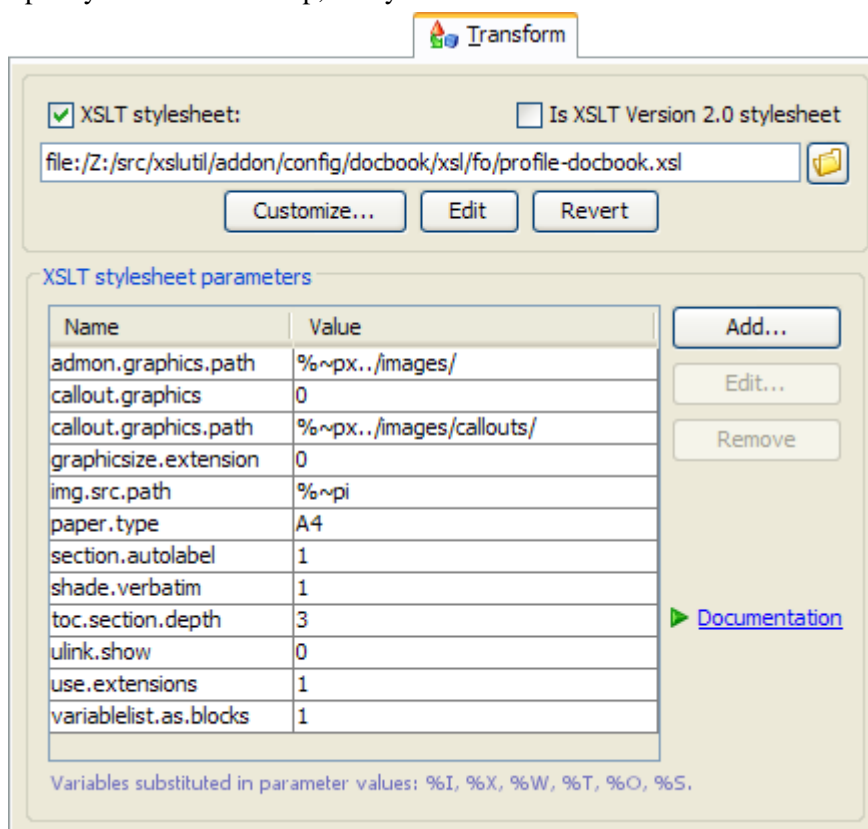
- a. Specify the name of your conversion in the **Name** text field. This name must be an *XML Name*, that is, to make it simple, it cannot start with a digit and it cannot contain space characters.
- b. Optionally describe what does your conversion in the **Description** text field.
- c. Optionally associate an icon to your conversion.

This icon may be selected from a predefined list (button ) or found in an external file (button )

- d. Optionally associate a category to your conversion. Generally, the type of the input document (DocBook, DITA, XHTML, etc) is used as a category.
- e. Optionally associate an icon to the category.
- f. Check the **File** radio button if your conversion creates a output file and in such case, specify the standard extension used for such file in the **Extension** text field.

If your conversion requires/creates an output directory, check the **Directory** radio button. In this case, note that:

- The output directory will be created if it does not exist. If it exists, the output directory will be made empty before performing the conversion (you'll have to confirm this interactively).
 - You'll need to specify the output directory to the XSLT stylesheet (see the transform step [13]) and/or to the process command (see the process step [14]) by referencing the %o variable in an XSLT style parameter value and/or in the process command.
3. Optional step:
- If your input document is a DITA topic, map or bookmap, you'll have to check the "**Input file is a DITA topic or map**" checkbox and fill some of the fields of the **DITA** tab. More information about this tab in Section 5.1, "Specifying the conversion of a DITA topic or map" [16].
 - If your input document is is an XHTML 1.0, 1.1 or 5.0 file, you'll have to check the "**Input file is XHTML**" checkbox and optionally, fill some of the fields of the **XHTML** tab. More information about this tab in Section 5.2, "Specifying the conversion of an XHTML page" [18].
 - If your input document is a DocBook v5.1+ *assembly* (i.e. not a book, chapter, article, etc), you'll have check to the "**Input file is a DocBook Assembly**" checkbox and optionally, fill some of the fields of the "**DocBook Assembly**" tab. More information about this tab in Section 5.3, "Specifying the conversion of DocBook v5.1+ assembly" [20].
4. Specify the transform step, if any.



Transform

XSLT stylesheet: Is XSLT Version 2.0 stylesheet

file:/Z:/src/xslutil/addon/config/docbook/xsl/fo/profile-docbook.xsl

Customize... Edit Revert

XSLT stylesheet parameters

| Name | Value |
|------------------------|-------------------------|
| admon.graphics.path | %~px../images/ |
| callout.graphics | 0 |
| callout.graphics.path | %~px../images/callouts/ |
| graphicsize.extension | 0 |
| img.src.path | %~pi |
| paper.type | A4 |
| section.autolabel | 1 |
| shade.verbatim | 1 |
| toc.section.depth | 3 |
| ulink.show | 0 |
| use.extensions | 1 |
| variablelist.as.blocks | 1 |

Add... Edit... Remove

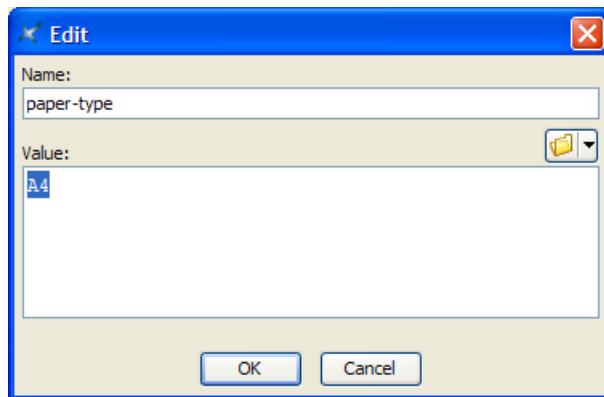
[Documentation](#)

Variables substituted in parameter values: %I, %X, %W, %T, %O, %S.

- Check the **XSLT stylesheet** checkbox if your conversion requires the XML input file to be transformed using an XSLT stylesheet.
- Specify the *URL* of the XSLT stylesheet in the corresponding text field.

The button next to the text field allows you to choose this XSLT stylesheet using the standard file chooser dialog box.

- c. Check the **"Is XSLT Version 2.0 stylesheet"** checkbox if the stylesheet you have chosen conforms to the XSLT Version 2 standard.
- d. Optionally, specify one or more XSLT stylesheet parameters by clicking the **Add** button.



Note that an XSLT stylesheet parameter value may reference one or more of the following variables: %I, %X, %W, %T, %O, %S. (Appendix A, *Variables* [38] describes all the variables supported by XMLmind XSL Utility.)



Select a parameter in the **XSLT stylesheet parameters** table by clicking on it and then, click the **Documentation** link to display its online documentation in your Web browser.

Unfortunately, the **Documentation** link is often disabled ("grayed") because many XSLT stylesheets are not documented and/or do not support any parameter.

5. Specify the process step, if any.

Process

Run XSL-FO processor: XFC

XSL-FO processor parameters

| Name | Value |
|---------------------|------------------------------------|
| genericFontFamilies | serif=Times·New·Roman,sans-seri... |
| imageResolution | 120 |
| outputEncoding | UTF-8 |
| outputFormat | docx |
| prescaleImages | false |

Variables substituted in parameter values: %I, %X, %W, %T, %O, %S.

Execute process command:

Variables substituted in the process command: %I, %X, %W, %T, %O, %S.

No processing

- a. Check the **Run XSL-FO processor** radio button and select this XSL-FO processor using the corresponding combobox if your conversion requires processing an XSL-FO file generated by the transform step (or specified as the XML input file when the transform step is omitted).

Alternatively, check the **Execute process command** radio button when your XSL-FO processor is not one of the three processors supported by XMLmind XSL Utility (example: Antenna House Formatter; see Appendix C, *Using Antenna House Formatter as your XSL-FO processor* [42]) or when your conversion requires a specific post-transformation processing (see Appendix D, *Using XMLmind XSL Utility to convert a DocBook 4 document to HTML Help (.chm file)* [44] for an example of such case).



About the process command

The process command is evaluated (after all the % variables have been substituted with their values) by `cmd.exe` on Windows and by `/bin/sh` on Linux/Mac.

The process command may reference one or more of the following variables: %I, %X, %W, %T, %O, %S. (Appendix A, *Variables* [38] describes all the variables supported by XMLmind XSL Utility.)

Finally, there are many cases where the transform step is all what you need (examples: conversion to HTML, Eclipse Help). In such case, simply check the **No processing** radio button.

- b. When the **Run XSL-FO processor** radio button has been checked, optionally specify one or more XSL-FO processor parameters by clicking the **Add** button.

These parameters are documented in Appendix B, *XSL-FO processor parameters* [39].

Note that an XSL-FO processor parameter value may reference one or more of the following variables: %I, %X, %W, %T, %O, %S. (Appendix A, *Variables* [38] describes all the variables supported by XMLmind XSL Utility.)

5.1. Specifying the conversion of a DITA topic or map

Figure 3. The *DITA* tab

Procedure 4. Minimal specification

1. Check the "**Input file is a DITA topic or map**" checkbox.
2. If you want to generate an HTML-based format (XHTML, HTML Help, etc), specify a relative path in the "**Copy images to**" field [17]. For example, specify "images".
3. Select the format of the generated document from the "**Target format**" combobox [18].

5.1.1. Reference of the fields found in the DITA tab

Default language

Specifies the main language of the document. Examples: en, en-US, fr, fr-CA. This information is needed in order 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).

Conditional processing profile

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

The buttons found at the top/right of this field allow respectively to:

- clear this field;
- edit, or simply view, the `.ditaval` file specified in this field;
- specify the URL of a `.ditaval` file by selecting it using a file chooser dialog box.

Table of Contents

Specifies whether to automatically generate a **Table of Contents** and, if a **Table of Contents** is to be generated, where to generate it. *Frontmatter* means at the beginning of the document. *Backmatter* means at the end of the document.

This option, like **List of Figures**, **List of Tables**, **List of Examples** and **Index**, is mainly useful when working with maps or individual topics. When working with a bookmap, the preferred way to specify the location, if any, of a **Table of Contents** is to do it in the bookmap itself. In all cases, what's specified in the bookmap has priority over the value of this option.

List of Figures

Specifies whether to automatically generate a **List of Figures** and, if a **List of Figures** is to be generated, where to generate it.

List of Tables

Specifies whether to automatically generate a **List of Tables** and, if a **List of Tables** is to be generated, where to generate it.

List of Examples

Specifies whether to automatically generate a **List of Examples** and, if a **List of Examples** is to be generated, where to generate it.

Index

Specifies whether to automatically generate an **Index** and, if an **Index** is to be generated, where to generate it.

Copy images to

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

When this field is left empty, the generated document will reference the image files using absolute URLs. This is harmless for PDF, RTF, etc, files because at the end of the conversion process, such files will *embed* a copy of the image files. However, this is rarely what is wanted for HTML-based formats (XHTML, Java Help, HTML Help, etc).

Chunk mode

Allowed values are **Automatic**, **Single** and **None**.

Chunk **Automatic** means: ignore the chunk specification found in the topic map and output a single chunk for the print medium; honor the chunk specification for the screen medium.

Chunk **None** means ignore the chunk specification found in the topic map and output a single chunk. As explained above, chunk **None** is implicit for some formats (PostScript, PDF, RTF, etc),

Both the **None** and **Single** values may be used to force the generation of a single output file. Chunk **Single** allows to reuse a map designed to output multiple HTML pages in order to generate a single HTML file or a PDF file.

Target format

The format of the generated document. This format must match the XSLT stylesheet specified in the "XSLT stylesheet" field of the **Transform** tab:

| Format | XSLT Stylesheet |
|---|--|
| XHTML 1.0 | <code>xslutil_install_dir/addon/config/dita/xsl/xhtml/xhtml1.xsl</code> |
| XHTML 1.1 | <code>xslutil_install_dir/addon/config/dita/xsl/xhtml/xhtml1_1.xsl</code> |
| HTML 4.01 | <code>xslutil_install_dir/addon/config/dita/xsl/xhtml/html.xsl</code> |
| XHTML 5 | <code>xslutil_install_dir/addon/config/dita/xsl/xhtml/xhtml5.xsl</code> |
| Web Help | <code>xslutil_install_dir/addon/config/dita/xsl/webhelp/webhelp.xsl</code> |
| Web Help containing XHTML 5 pages | <code>xslutil_install_dir/addon/config/dita/xsl/webhelp/webhelp5.xsl</code> |
| HTML Help (.chm) | <code>xslutil_install_dir/addon/config/dita/xsl/htmlhelp/htmlhelp.xsl</code> |
| Eclipse Help | <code>xslutil_install_dir/addon/config/dita/xsl/eclipsehelp/eclipsehelp.xsl</code> |
| EPUB 2 | <code>xslutil_install_dir/addon/config/dita/xsl/epub/epub.xsl</code> |
| EPUB 3 | <code>xslutil_install_dir/addon/config/dita/xsl/epub/epub3.xsl</code> |
| PostScript | <code>xslutil_install_dir/addon/config/dita/xsl/fo/fo.xsl</code> |
| PDF | <code>xslutil_install_dir/addon/config/dita/xsl/fo/fo.xsl</code> |
| OpenOffice (.odt, OpenOffice.org 2+) | <code>xslutil_install_dir/addon/config/dita/xsl/fo/fo.xsl</code> |
| RTF (MS-Word 2000+) | <code>xslutil_install_dir/addon/config/dita/xsl/fo/fo.xsl</code> |
| WordprocessingML (MS-Word 2003+) | <code>xslutil_install_dir/addon/config/dita/xsl/fo/fo.xsl</code> |
| Office Open XML (.docx, MS-Word 2007+) | <code>xslutil_install_dir/addon/config/dita/xsl/fo/fo.xsl</code> |
| XSL-FO | <code>xslutil_install_dir/addon/config/dita/xsl/fo/fo.xsl</code> |

Of course, it is also possible to specify XSLT stylesheets which specialize the above ones.

Preprocessor messages

Specifies the level of verbosity of the DITA preprocessor. Allowed values are (from not verbose to very verbose): **None**, **Information**, **Verbose**, **Debug**.

5.2. Specifying the conversion of an XHTML page

If the document being converted is an XHTML 1.0, 1.1 or 5.0 file then you must check the "Input file is XHTML" checkbox. Note that if you forget to do this, the conversion will work, but the images ref-

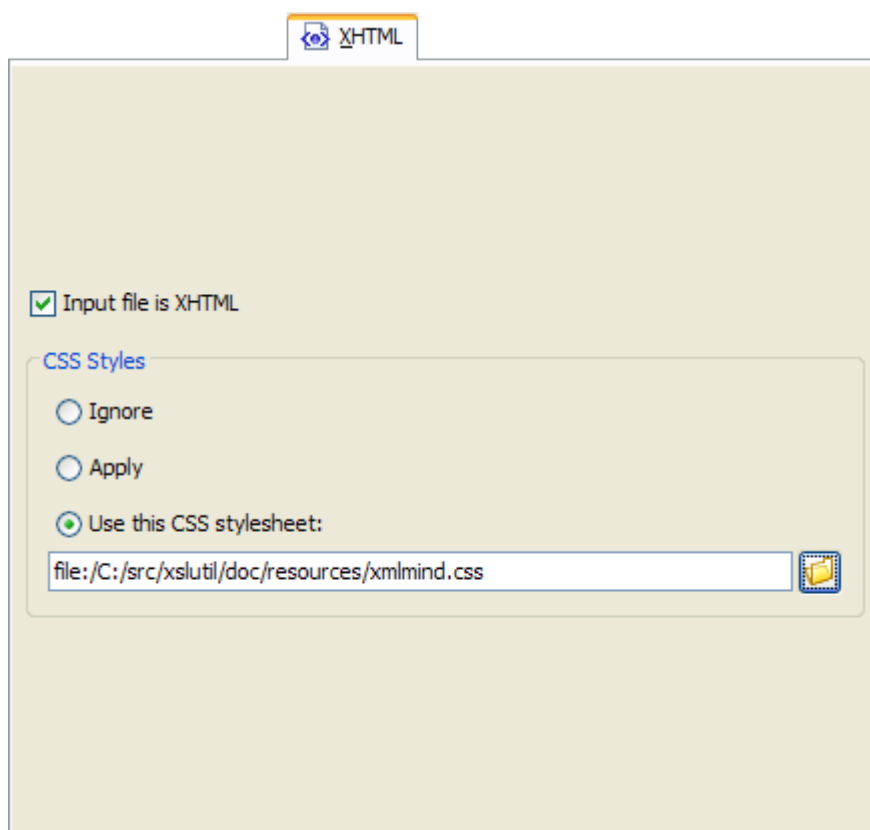
erenced in the XHTML input file will all be omitted in the output file and also, all the CSS styles possibly specified in the XHTML input file will be ignored.



Converting plain HTML (that is, not XHTML) files

XMLmind XSL Utility does not support converting plain HTML (that is, not XHTML) files. However XMLmind XSL Server does. Therefore when XMLmind XSL Utility is used to configure the conversions of XMLmind XSL Server, possibly using the `-p` command-line option [30], then also check "**Input file is XHTML**" when the input file is plain HTML.

Figure 4. The XHTML tab



The **CSS styles** options allow to specify what to do with the CSS styles possibly found in the XHTML input file. (These CSS styles are found in `style` attributes, `style` elements and `link` elements pointing to CSS stylesheets.)

Ignore

Ignore all the CSS styles specified in the XHTML input file.

Apply

Apply to the intermediate XSL-FO file generated by the XHTML XSLT 2 stylesheets all the CSS styles specified in the XHTML input file.

This is the default option. It emulates —to a certain extent— how a Web browser typically renders an HTML page.

Use this CSS stylesheet

Ignore all the CSS styles specified in the XHTML input file. Instead apply to the intermediate XSL-FO file generated by the XHTML XSLT 2 stylesheets all the styles found in specified CSS stylesheet.

5.3. Specifying the conversion of DocBook v5.1+ assembly

If the document being converted is DocBook v5.1+ *assembly* (i.e. not a book, chapter, article, etc) then you must check the "**Input file is a DocBook Assembly**" checkbox.

Figure 5. The *DocBook Assembly* tab

An assembly may contain several `structure` elements. A `structure` element may specify several output formats (`web`, `print`, etc)³. When this is the case, you'll want to specify which structure you want to convert and also which output format you target.

Structure ID

The value of the `xml:id` attribute of the `structure` element to be converted. By default, it's the first `structure` found in the assembly.

Output format

One of the output formats specified in the `structure` selected using the above "**Structure ID**" field. By default, it's the value of the `defaultformat` attribute of the `structure` if any, the "*implicit format*" otherwise.

The "implicit format" matches `output`, `filterin`, `filterout` elements without any `outputformat` attribute.

³The `outputformat` attribute set on `output`, `filterin`, `filterout` elements allows to specify "classes" of output formats rather than actual output formats (PDF, DOCX, EPUB, etc).

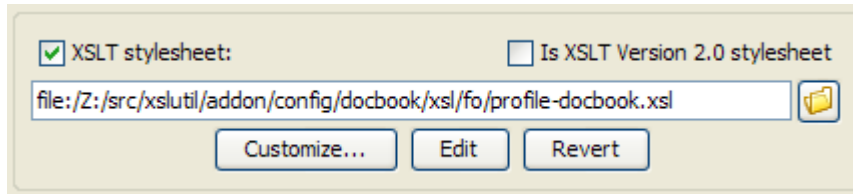
Multiple output formats separated by ";" may be specified. For example, "web;print" means output format is "web" *OR* "print".

Preprocessor messages


Specifies the level of verbosity of the assembly processor. Allowed values are (from not verbose to very verbose): **None**, **Information**, **Verbose**, **Debug**.

5.4. Customizing a stock XSLT stylesheet

The "**XSLT stylesheet**" frame found at the top of the **Transform** tab has buttons allowing to easily customize the XSLT stylesheet currently selected.



5.4.1. Using an existing XSLT stylesheet customization

If you already have an XSLT stylesheet customization, simply specify its location by clicking , the "**Choose XSLT stylesheet**" button.

However if you do this, please make sure that your XSLT stylesheet customization imports the stock XSLT stylesheet by using an URL starting with "xslutil-config:". DITA example:

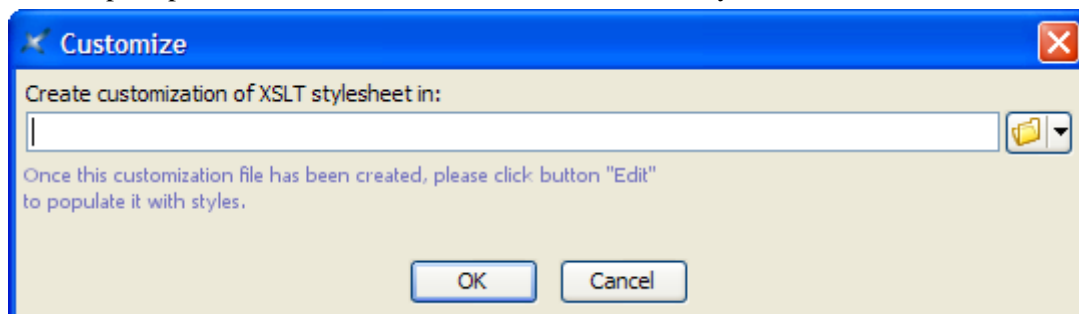
```
xsl:import href="xslutil-config:dita/xsl/fo/fo.xsl"/>
```

5.4.2. Creating an XSLT stylesheet customization

If you want to create an XSLT stylesheet customization, please proceed as follows:

1. Click **Customize**.

You are prompted for the .xsl save file which is to contain your customization.



Clicking **OK** creates a custom XSLT stylesheet based on the currently selected stock XSLT stylesheet.

A custom XSLT stylesheet created this way merely imports the original XSLT stylesheet. Example:

```
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
                version="1.0">
  <xsl:import href="xslutil-config:docbook/xsl/fo/docbook.xsl"/>
  <!-- REDEFINE PARAMETERS AND ATTRIBUTE-SETS HERE -->
</xsl:stylesheet>
```

Note that it's not possible to create a customization of a custom XSLT stylesheet.

2. Click **Edit**.

If the custom XSLT stylesheet has been created using the **Customize** button, this button starts the "XMLmind XSL Customizer" application [22] in order to edit it. Otherwise, this button starts a helper application [31] allowing to edit XSLT files.



Clicking **Edit** while keep the Shift key pressed allows to start a helper application rather than the "XMLmind XSL Customizer" application.

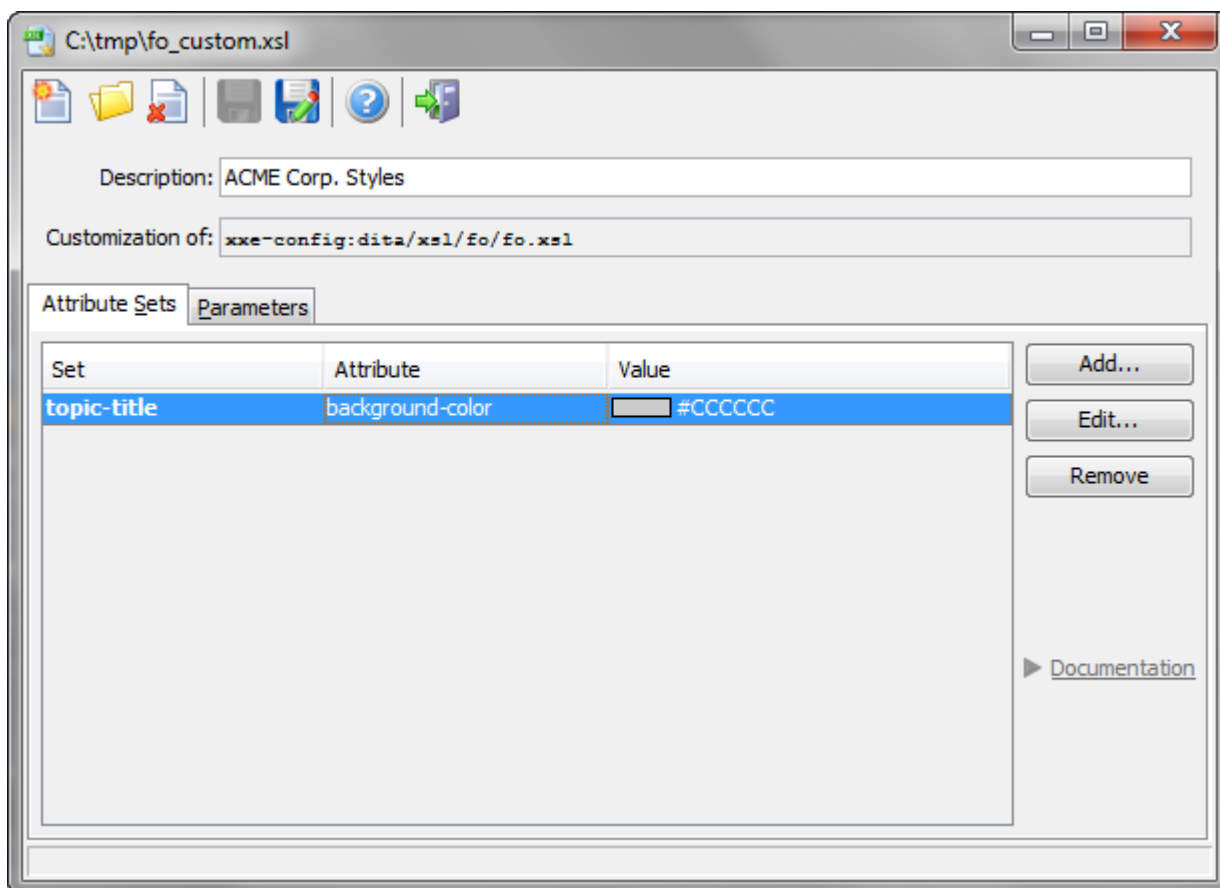
If you have customized an XSLT stylesheet and want to revert to the original XSLT stylesheet, simply click **Revert**.

5.4.3. The "XMLmind XSL Customizer" application

5.4.3.1. Introduction

The "XMLmind XSL Customizer" application is a companion application embedded in XMLmind XML Editor and XMLmind XSL Utility.

Figure 6. "XMLmind XSL Customizer" main window



This application is invoked by clicking the **Edit XSLT** stylesheet button found in XMLmind XML Editor and XMLmind XSL Utility. It allows to modify a custom XSLT stylesheet created by clicking the **Customize XSLT** stylesheet button found in XMLmind XML Editor and XMLmind XSL Utility.

A custom XSLT stylesheet created this way merely imports the stock XSLT stylesheet. Example:

```
<?xml version="1.0" encoding="UTF-8"?>
<?stylesheet-label ACME Corp. Styles?>
<xsl:stylesheet version="2.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:import href="dita-config:xsl/fo/fo.xml" />
  <!-- REDEFINE PARAMETERS AND ATTRIBUTE-SETS HERE -->
</xsl:stylesheet>
```

"XMLmind XSL Customizer" is not an XSLT editor. However it allows to:

- add or change attributes in some of the *attribute sets* supported by the XSLT stylesheet,
- change the value of some of the *parameters* supported by the XSLT stylesheet,

and this, *without prior knowledge of XSLT*.

What are attribute sets and parameters?

XSLT stylesheets are often (but not always) parameterized by the means of:

- Parameters. DocBook XSL stylesheet example:

```
<xsl:param name="paper.type"❶>USletter❷</xsl:param>
```

- ❶ The name of the parameter is "paper.type".
- ❷ The value of the parameter is literal string "USletter".

- Attribute sets. DocBook XSL stylesheet example:

```
<xsl:attribute-set name="monospace.verbatim.properties"❶
    use-attribute-sets="verbatim.properties monospace.properties"❷>
  <xsl:attribute name="text-align">start</xsl:attribute>❸
  <xsl:attribute name="wrap-option">no-wrap</xsl:attribute>❹
</xsl:attribute-set>
```

- ❶ The name of the attribute set is "monospace.verbatim.properties".
- ❷ This attribute set includes two other attribute sets: `verbatim.properties` and `monospace.properties`.
- ❸ This attribute set directly contains attribute `text-align="start"`.
- ❹ This attribute set directly contains attribute `wrap-option="no-wrap"`.

An attribute set can contain any attribute. However attribute sets are mainly used in XSLT stylesheets which generate XSL-FO. The XSL-FO intermediate file generated by the XSLT stylesheet is then processed by programs such as Apache FOP, RenderX XEP, Antenna House XSL Formatter, XMLmind XSL-FO Converter, etc, in order to create the deliverable: PDF, PostScript, RTF, .docx, .odt, etc. The attribute sets are used in this case because they are the only way to influence the look of the deliverable. Such attribute sets contain standard XSL-FO presentation attributes (very similar to the CSS properties): `color`, `font-family`, `line-height`, `margin-left`, etc.

5.4.3.2. Reference

Toolbar buttons



New

Create a customization of an existing XSLT stylesheet. This button displays the standard file chooser dialog box allowing to choose the XSLT stylesheet for which a customization is to be created.



Open

Open a custom XSLT stylesheet previously created by clicking the **New** button. This button displays the standard file chooser dialog box allowing to choose the custom XSLT stylesheet to be opened.

**Close**

Close currently opened XSLT stylesheet.

**Save**

Save the changes made to currently opened XSLT stylesheet.

**Save As**

Save currently opened XSLT stylesheet to a different file.

**Help**

Display this online help.

**Quit**

Close "XMLmind XSL Customizer" main window.



Because "XMLmind XSL Customizer" is an (embedded) application and not a modal dialog box, you can keep it open while converting an XML document in XMLmind XML Editor or in XMLmind XSL Utility. This allows to experiment with attribute sets and parameters until you are satisfied with the result of the conversion.

Information fields about the currently opened XSLT stylesheet

Description:

Short description of the currently opened custom XSLT stylesheet. XMLmind XML Editor requires a custom XSLT stylesheet to have such description.

Customization of:

Read-only text field: URI of the stock XSLT stylesheet for which the currently opened XSLT stylesheet is a customization.

Attribute Sets tab buttons

Add

Add an attribute to one of the attribute sets supported by currently opened XSLT stylesheet.

This button displays the **Add/Edit** attribute dialog box. How to use this dialog box is described in the example below [28].

Figure 7. The **Add/Edit** attribute dialog box



XMLmind XSL Customizer is designed for users who cannot “program in XSLT”. These users are expected to enter literal values, not XSLT elements, in the **Custom value** field. For example, they are expected to enter something like:

```
20pt
```

as the value of the `font-size` attribute, and not something like:

```
<xsl:value-of select="$body.font.master * 2" />
<xsl:text>pt</xsl:text>
```

However, if the user happens to know what she/he is doing and nevertheless enters one or more XML nodes in the **Custom value** field, then XMLmind XSL Customizer will ask her/him to confirm that this is really what she/he wants and make it work.

Edit

Modify currently selected attribute.

This button displays the **Add/Edit** attribute dialog box.

Remove

Remove currently selected attribute.

Documentation

Start the web browser and make it display the page containing the documentation about currently selected attribute. This button is disabled (grayed) when such documentation is not available. For now, only the DocBook XSL Stylesheets provide some documentation for their attribute sets.

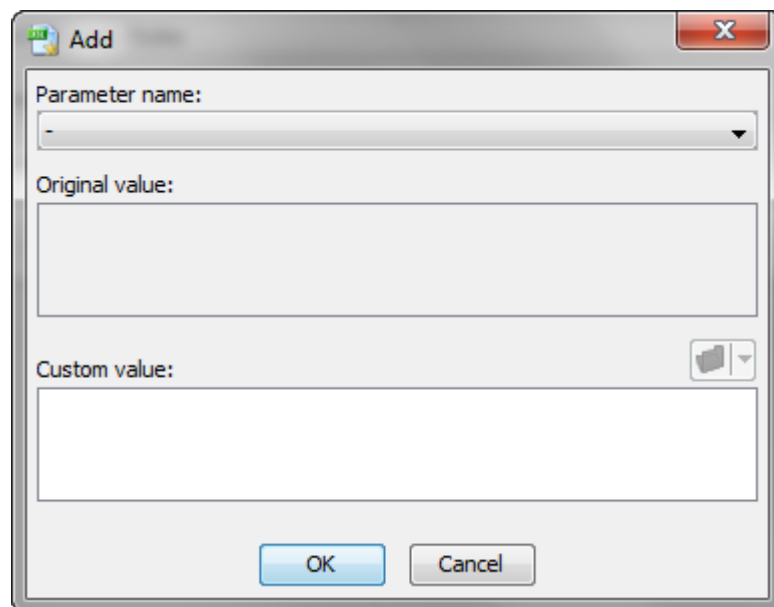
Parameters tab buttons

Add

Set one of the parameters supported by currently opened XSLT stylesheet.

This button displays the **Add/Edit** parameter dialog box. How to use this dialog box is described in the example below [29].

Figure 8. The **Add/Edit** parameter dialog box



XMLmind XSL Customizer is designed for users who cannot “program in XSLT”. These users are expected to enter literal values, not XSLT elements, in the **Custom value** field. For example, they are expected to enter something like:

```
40pt
```

as the value of the `body.start.indent` parameter, and not something like:

```
<xsl:value-of select="$body.font.master * 4"/>
<xsl:text>pt</xsl:text>
```

However, if the user happens to know what she/he is doing and nevertheless enters one or more XML nodes in the **Custom value** field, then XMLmind XSL Customizer will ask her/him to confirm that this is really what she/he wants and make it work.

Edit

Modify currently selected parameter.

This button displays the **Add/Edit** parameter dialog box.

Remove

Remove currently selected parameter.

Documentation

Start the web browser and make it display the page containing the documentation about currently selected parameter. This button is disabled (grayed) when such documentation is not available.

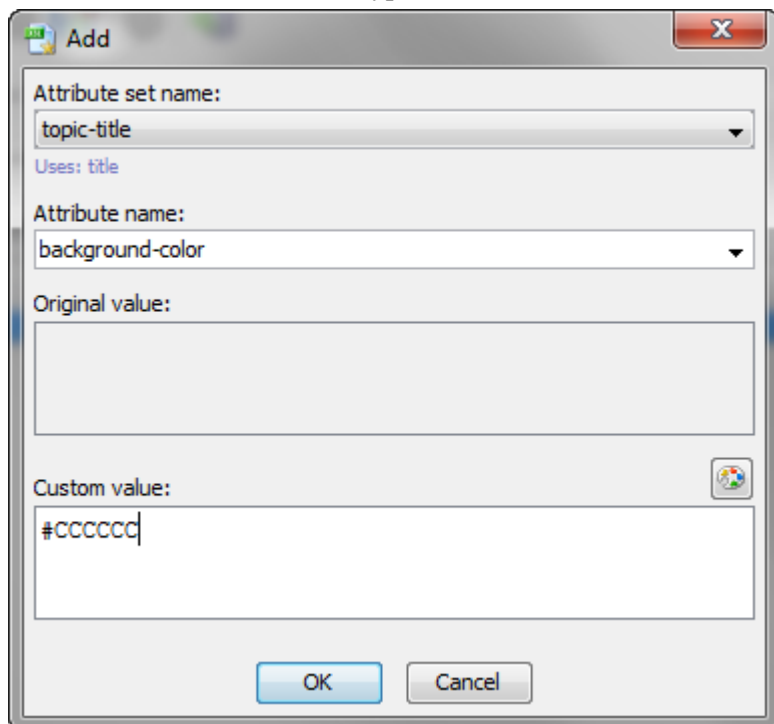
Example 1. When converting a DITA map to PDF, give a light gray background to all the topic titles


This is specified by adding attribute `background-color=#CCCCCC` to the attribute-set called `topic-title`.

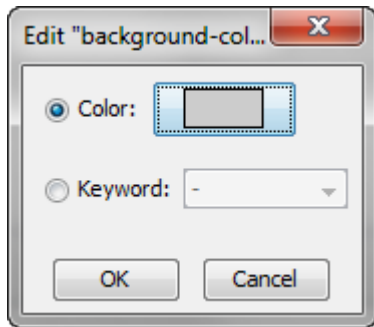
1. Select the **Attribute Sets** tab.
2. Click **Add**. This displays the **Add/Edit** attribute dialog box.
3. In the "Attribute set name:" combobox, select `topic-title`.
4. In the "Attribute:" combobox, type "background-color" or select this commonly used attribute from the drop down list.

The "Original value:" read-only text field remains empty, indicating that the stock XSLT stylesheet does not specify attribute `topic-title/background-color`.

5. In the "Custom value:" field, type "#CCCCCC".



Or more simply, click  "Edit style attribute" and use the `background-color` editor to specify a light gray color.



6. Click **OK** to close the dialog box.

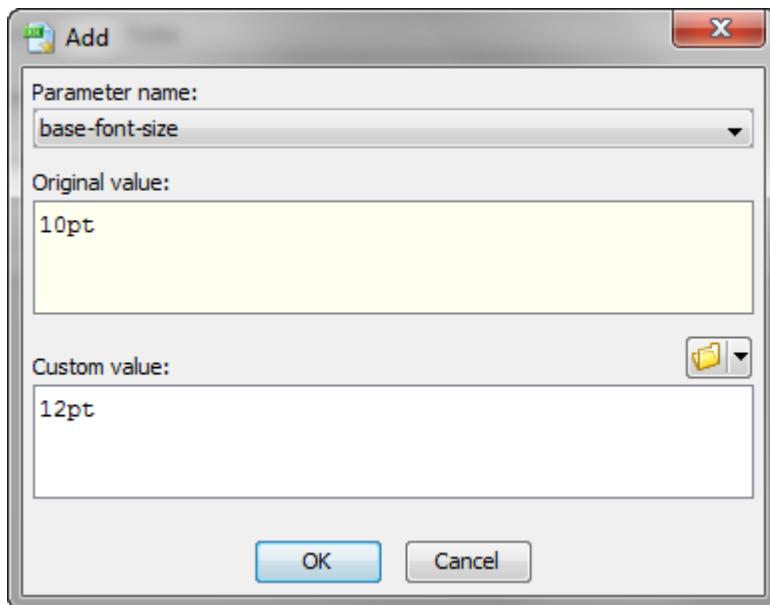
Example 2. When converting a DITA map to PDF, use a 12pt base font size

This is specified by setting parameter `base-font-size` to 12pt.

1. Select the **Parameters** tab.
2. Click **Add**. This displays the **Add/Edit** parameter dialog box.
3. In the "**Parameter name:**" combobox, select `base-font-size`.

The "**Original value:**" read-only text field changes to "10pt", which is the value of parameter `base-font-size` specified in the stock XSLT stylesheet.

4. In the "**Custom value:**" field, type "12pt".



5. Click **OK** to close the dialog box.

6. User preferences

By default, the user preferences are stored in the following directory:

- `$HOME/.xfC/` on Linux.
- `$HOME/Library/Application Support/XMLmind/FOConverter/` on the Mac.

- %APPDATA%\XMLmind\FOConverter\ on Windows. Example: C:\Users\john\AppData\Roaming\XMLmind\FOConverter\.

You may delete this directory if you want to restore the “factory settings”.

File `user_preferences_dir/xslutil.conversions` contains the modifications made to the stock conversion specifications and the original conversion specifications personally created by the user. Deleting just this file will have the effect to restore all the stock conversion specifications (`db5Toxxx`, `dbToxxx`, `xhtmlToxxx`, etc) and to delete all the user's original conversion specifications.

6.1. The `-p` command-line option

The `-p` command-line option allows to specify a custom user preferences directory. This option may be used when XMLmind XSL Utility is used as a graphical application:

```
/opt$ xslutil-6_5_0/bin/xslutil -p /usr/local/share/xslutil &
```

or when XMLmind XSL Utility is used as a command-line tool:

```
/opt/xslutil-6_5_0/bin$ xslutil -p /usr/local/share/xslutil \
dbToDocx /tmp/help.xml /tmp/help.docx
```

Specifying a custom user preferences directory allows to use XMLmind XSL Utility, a convenient graphical application, to configure the conversions of XMLmind XSL Server. For example, if `xslsrv/WEB-INF/web.xml` contains:

```
<init-param>
  <param-name>customizeDir</param-name><param-value>/etc/xslutil</param-value>
</init-param>
```

then run `xslutil -p /etc/xslutil` to configure the conversions of XMLmind XSL Server.

6.2. General preferences

Use the native file chooser in preference to the multi-platform file chooser

If this toggle is checked, the native file chooser will be used in preference to Java's multi-platform file chooser.

Note that file extension filters are not supported when the native file chooser is invoked by Java™.

Default: not checked on Windows and on the Mac, disabled (grayed out) on the other platforms.

Warn me when the output directory is not empty

Some conversion (example: conversion to multi-page HTML, to Eclipse Help) require an output directory rather than an output file. This output directory will be created if it does not exist. However, if it exists, the output directory will be made empty before performing the conversion, which is potentially dangerous. When this option is turned on, you'll have to confirm that you really want to delete all the files contained in the output directory before proceeding with the conversion. We do not recommend to turn this option off.

Default: checked.

Cache compiled stylesheets

Compiling an XSLT stylesheet may take a few seconds. When this option is turned on, an XSLT stylesheet is compiled the first time it is used and then, it is cached in compiled form for all subsequent uses. Unless you are developing a XSLT stylesheet and testing it with XMLmind XSL Utility, there is no need to turn this option off.

Default: checked.

Restore stock conversion specifications

Clicking this button allows to restore all the stock conversion specifications. The user's original conversion specifications are, of course, left untouched.

6.3. Helper applications preferences



Converting DITA documents to HTML Help

In addition to letting you specify which external applications are launched when "**Preview result in a helper application**" [8] is checked, this preferences sheet also lets you specify the location of `hnc.exe`, an external tool, which is required to convert DITA documents to HTML Help.

File types

List of file types. Each file type has an associated helper application. This helper application is assumed to be able to open files detected as having this type. A helper application may be a viewer or an editor.

The following buttons may be used to modify this list:

Add

Displays the "Helper Application Editor" dialog box [32] in order to add a new file type to the list.

Edit

Displays the "Helper Application Editor" dialog box [32] in order to view or modify selected file type.

Remove

Removes selected file type from the list.

Default viewer

Specifies which default viewer to use in case the type of the file to be opened has not been detected. In practice, commands making use of the default viewer typically assumes that it is in fact *a Web browser*. This implies that these commands assume that a default viewer can open URLs as well as filenames and that it can open text, HTML, GIF, PNG and JPEG files.

This field must contain a command line interpreted by the native shell of the platform: `cmd.exe` on Windows and `/bin/sh` on the Mac and on Unix.

This command line must reference one of these two substituted variables: `%U` and `%F`. In principle, `%U` is replaced by the URL of the file to be opened by the helper application and `%F` is replaced by a filename. In practice, `%U` is just a *hint* meaning: the helper application can open URLs as well as filenames.

Default: depends on the platform:

- On Windows: `start "" "%U"`
- On the Mac: `open "%U"`
- On Unix: dynamically detected. For example: `firefox "%U"`

The following buttons may be used to modify this field:

Reset

Resets the field to its default value (see above).

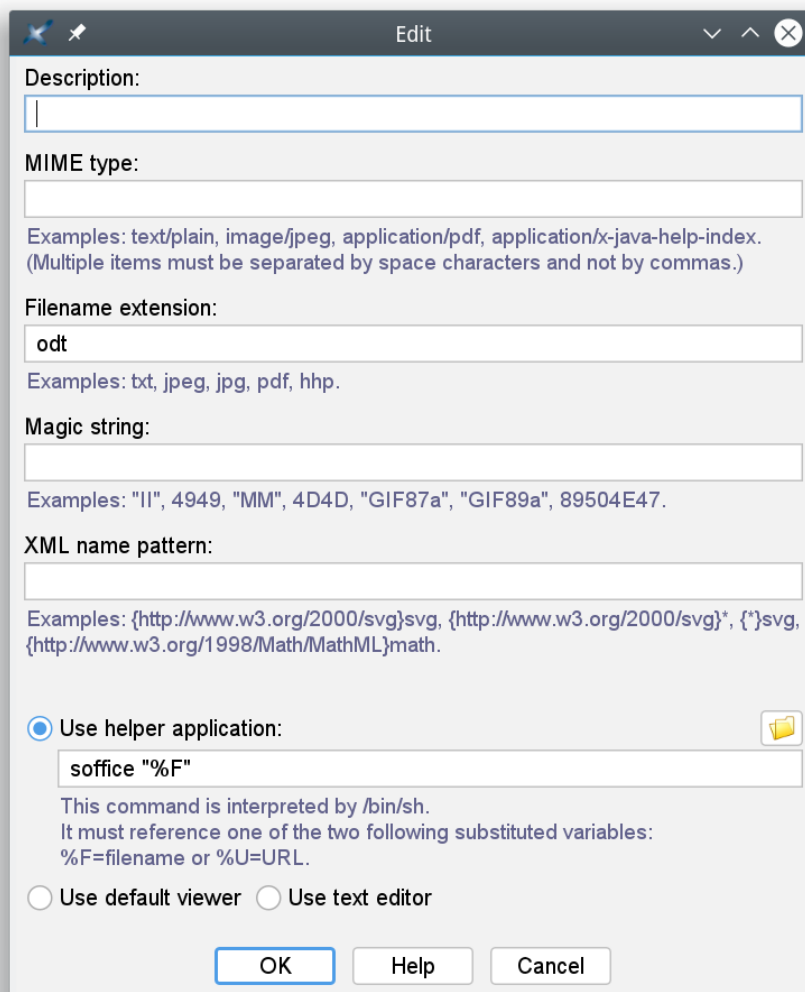
Choose

Displays the standard file chooser in order to specify an application (e.g. a `.exe` or a `.bat` file on Windows). String `"%F"` is automatically appended to the chosen application.

See also Helper applications on the Mac [34].

6.3.1. The "Helper Application Editor" dialog box

This dialog box allows to view or modify a file type listed in the "**Helper applications**" preferences sheet [31].



A file type is specified by *at least one* of the following characteristics:

MIME type

The official (or just well-known) formal name of the file type. Generally returned by Web servers to their client Web browsers. Non-registered MIME types typically start with string "application/x-".

A MIME type may end with a wildcard. Example: "image/*" matches "image/gif", "image/jpeg", etc.

Examples: text/plain, image/jpeg, application/excel, application/x-java-help-index.

Filename extension

If the filename or URL of a file ends with specified ".*extension*", then this file is detected as having this file type.

An extension may or may not start with a dot. This is unimportant because, in all cases, a leading dot would be automatically stripped.

Examples: txt, jpeg, jpg, xls.

Magic string

For some file formats, the first bytes of a file are always the same and therefore, can be considered as being the *signature* of this file type.

If a file starts with specified first bytes, then this file is detected as having this file type. This type of detection is supposed to work like magic, hence the name: "magic string".

A magic string may be specified by a the hexadecimal representation of a sequence of bytes (example, one of the two TIFF magic strings: 4949) or by a *quoted* sequence of ASCII characters (same example, one of the two TIFF magic strings: "II").

Examples: TIFF: "II" or 4949, "MM" or 4D4D; GIF: "GIF87a", "GIF89a"; PNG: 89504E47; PDF: "%PDF-".

XML name pattern

If the root element of an XML file has a name which matches specified pattern, then this XML file is detected as having this file type.

An XML name pattern follows this syntax:

```
( '{' namespace_URI? '}' )? local_part
```

One of *local_part* or *namespace_URI* may be equal to wildcard "*".

Examples: {*}svg, {http://www.w3.org/1998/Math/MathML}:math.

Each file type has an associated helper application. This helper application is assumed to be able to open files detected as having this type. A helper application may be a viewer or an editor.

Description

Description of the file type. Not mandatory, just recommended. This text is displayed in the **File types** list of the "**Helper applications**" preferences sheet [31].

MIME type

One or more MIME types (see definition [33] above) separated by spaces.

Filename extension

One or more extensions (see definition [33] above) separated by spaces.

Magic string

One or more magic strings (see definition [33] above) separated by spaces.


XML name pattern

One or more XML name patterns (see definition [33] above) separated by spaces.

Use helper application

This field must contain a command line interpreted by the native shell of the platform: `cmd.exe` on Windows and `/bin/sh` on the Mac and on Unix.

This command line must reference one of these two substituted variables: `%U` and `%F`. In principle, `%U` is replaced by the URL of the file to be opened by the helper application and `%F` is replaced by a filename. In practice, `%U` is just a *hint* meaning: the helper application can open URLs as well as filenames.

The  **Choose Helper Application** button displays the standard file chooser in order to specify an application (e.g. a `.exe` or a `.bat` file on Windows). String " `%F`" is automatically appended to the chosen application.



Helper applications on the Mac

When an *application*, that is, a folder having a name ending with hidden suffix `.app`, containing a package bundle, has been selected by the user, the **Choose Helper Application** button automatically prepends `"open -w -n -a "` to chosen `.app` folder. Example: `"open -w -n -a /Applications/Inkscape "%F"`.

Options `"-w -n"` mean: start a new instance of the application and wait until this instance has exited. These options are required when the helper application is used to edit the content of an element, the content of an attribute or the whole document.

Use default viewer

Files having specified type are to be opened using the default viewer. The default viewer is specified in the "**Helper applications**" section of the "**Preferences**" dialog box [31].

Use text editor

Files having specified type are to be opened using the "default text editor".

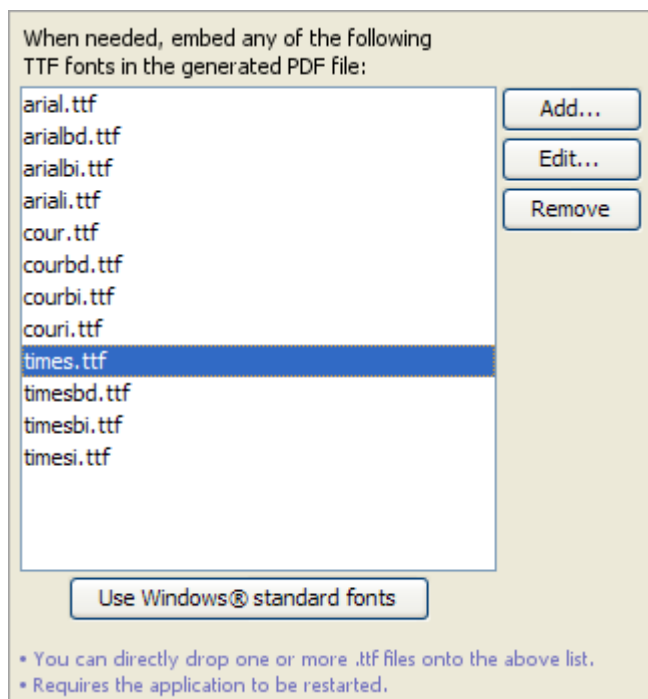


Which text editor?

This text editor to be used is *the helper application associated to the "text/plain" MIME type*. If such association has not been defined, the standard text editor of the platform is automatically used (e.g. on Windows, it's `notepad`).

6.4. FOP preferences

Figure 9. The "FOP" preferences sheet



By default, only the 14 built-in fonts: Times, Helvetica, Courier, Symbol and ZapfDingbats are used in the generated PDF. The above preferences sheet allows to specify which custom TrueType (.ttf) fonts are to be *embedded* in the generated PDF.

This facility is useful in the following two cases:

- The 14 PDF standard fonts (Helvetica, Times, Courier, etc), which are used by default by FOP, have glyphs only for the Western languages. If, for example, you convert a DocBook document written in Russian to PDF, the generated PDF will mainly contain the '#' placeholder character. Fortunately, widely available TTF fonts such as Microsoft® Arial, Times New Roman and Courier New or the DejaVu fonts have glyphs for almost all the languages of the world.
- Use fonts nicer than the 14 PDF standard fonts.

Procedure 5. How to use Times New Roman, Arial and Courier New instead of Times, Helvetica, Courier

1. Click **Use Windows® standard fonts**.

Note that the **Use Windows® standard fonts** button is grayed if the Arial font is not found in the standard fonts folder of your system.

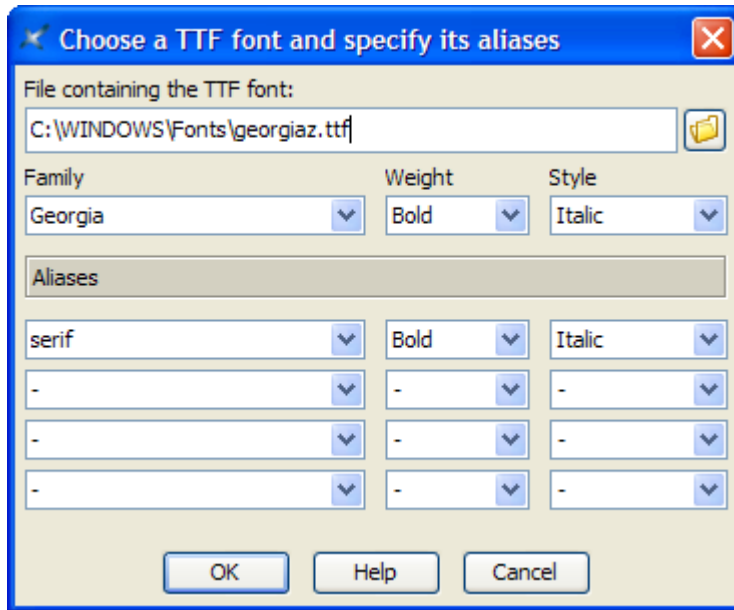
2. Click **OK**.
3. Restart XMLmind XSL Utility.

Procedure 6. How to choose specific fonts (for example, you want to replace Times fonts by Georgia fonts)

1. Click **Add**.

This displays the following dialog box:

Figure 10. The "Choose a TTF font and specify its aliases" dialog box



- a. Choose the .ttf file containing font "Georgia".



On Windows, for permissions reasons, there is no way to pick a font file from the "C:\Windows\Fonts\" folder using the standard file chooser. Therefore the only way to register with Apache FOP a font found in "C:\Windows\Fonts\" is to drag its file from the Windows file manager and to drop it onto the list found in the "FOP" preferences sheet (see above figure [36]).

However, after you do this, do not forget to select each entry added by the drop action and then click **Edit** to possible change or complement what's have been automatically specified there.

- b. Specify the following alias: serif.
- c. Click **OK**.
2. Click **Add**.
 - a. Choose the .ttf file containing font "Georgia Bold".
 - b. Specify the following alias: serif, Bold.
 - c. Click **OK**.
3. Click **Add**.
 - a. Choose the .ttf file containing font "Georgia Italic".
 - b. Specify the following alias: serif, Italic.
 - c. Click **OK**.

4. Click **Add**.
 - a. Choose the `.ttf` file containing font "Georgia Bold Italic".
 - b. Specify the following alias: serif, Bold, Italic.
 - c. Click **OK**.

5. Click **OK**.

Doing this automatically creates a standard FOP configuration file in `user_preferences_dir/fop/fop.conf`. User preferences directory `user_preferences_dir` is documented in Section 6, "User preferences" [29].

6. It is recommended to repeat the above steps in order to specify fonts replacing Helvetica, that is, fonts having a sans-serif alias and fonts replacing Courier, that is fonts having a monospace alias.
7. Restart XMLmind XSL Utility.



Using in XSL Utility an existing FOP configuration file

The location of an existing Apache FOP configuration file may be specified to XSL Utility by the means of Java™ system property or environment variable `XXE_FOP_CONFIG`. The value of this variable is an URL or an absolute file path. Examples: `-DXXE_FOP_CONFIG=http://localhost/~john/fop/fop.conf` (Java™ system property), `set XXE_FOP_CONFIG=C:\Users\john\misc\fop\fop.conf` (Windows environment variable).

6.5. XEP preferences

XEP preferences are identical to FOP preferences [35]. There are two minor differences though:

- Some fonts have licensing restrictions that forbid embedding them in a PDF file. RenderX XEP enforces these licensing restrictions, not Apache FOP. XMLmind XSL Utility has currently no way to detect these licensing restrictions, therefore you may follow the above procedure and end up with glyphs still missing in the generated PDF.
- The standard XEP configuration file is created in `user_preferences_dir/xep/xep.conf`.



Using in XSL Utility an existing XEP configuration file

The location of an existing RenderX XEP configuration file may be specified to XSL Utility by the means of Java™ system property or environment variable `XXE_XEP_CONFIG`. The value of this variable is an URL or an absolute file path. Examples: `-DXXE_XEP_CONFIG=http://localhost/~john/xep/xep.conf` (Java™ system property), `set XXE_XEP_CONFIG=C:\Users\john\misc\xep\xep.conf` (Windows environment variable).

A. Variables

| Variable | Substituted in | Value |
|----------|--|---|
| %I | Output file field, XSLT stylesheet parameter value, XSL-FO processor parameter value, process command. | The input file. |
| %X | XSLT stylesheet parameter value, XSL-FO processor parameter value, process command. | <p>The XSLT stylesheet file.</p> <p>When there is no transform step or when the XSLT stylesheet is not located on the local file system, the value of this variable is the empty string.</p> <p>Note that the value of variable %x may be the empty string and that, at the same time, the value of variable %x may be a valid URI. See the URI counterparts below.</p> |
| %W | XSLT stylesheet parameter value, XSL-FO processor parameter value, process command. | <p>A temporary working directory always created by a conversion, just in case it could be useful.</p> <p>Not available when the "Input file is a DITA topic or map" checkbox is checked.</p> |
| %T | XSLT stylesheet parameter value, XSL-FO processor parameter value, process command. | <p>The temporary file generated by the transform step.</p> <p>This temporary file is created in the directory of the input file, that is %~pT is equal to %~pI (see modifiers below). This implies that you must have the write permissions on the directory containing the input file.</p> <p>Remarks:</p> <ul style="list-style-type: none"> • When there is no transform step, the value of this variable is the empty string. • When there is no process step and when the conversion does not require/create an output directory, the transform step directly generates the output file (variable %O). In such case, variable %T is still defined, but it is not actually used. <p>This variable is always equal to %~pO%S%~rO.fO when the "Input file is a DITA topic or map" checkbox is checked.</p> |
| %O | XSLT stylesheet parameter value, XSL-FO processor parameter value, process com- | The output file or directory. |

| Variable | Substituted in | Value |
|----------|---|--|
| | mand, preview command. | |
| %S | Everywhere: output file field, XSLT stylesheet parameter value, XSL-FO processor parameter value, process command, preview command. | Portable filename separator: '\' on Windows, '/' on all the other platforms. |

The following variables: %i, %x, %w, %t, %o are the *URI counterparts* of the variables representing filenames: %I, %X, %W, %T, %O. Examples:

- If the value of variable %I is C:\Users\john\Documents\doc src>manual.xml, then the value of variable %i is file:/C:/Users/john/Documents/doc%20src/manual.xml.
- If the value of variable %x is http://docbook.sourceforge.net/release/xsl-ns/1.79.1/fo/docbook.xsl, then the value of variable %x is the empty string.

When a variable represents a filename or an URI, the following *modifiers* allow to refer to a part of this filename or URI. Let's suppose that the value of variable %I is C:\Users\john\Documents\doc src>manual.xml.

| Modifier | Description | Example |
|----------|--|---|
| ~p | Parent directory. | %~pI = C:\Users\john\Documents\doc src %~pi = file:/C:/Users/john/Documents/doc%20src/ (note the trailing '/') |
| ~n | Basename of the file, including the extension. | %~nI = manual.xml %~ni = manual.xml |
| ~r | Basename of the file, without the extension. | %~rI = manual %~ri = manual |
| ~e | File extension, if any. | %~eI = xml %~ei = xml |

B. XSL-FO processor parameters

1. XMLmind XSL-FO Converter (XFC)

Any of the options documented in XMLmind XSL-FO Converter - User's Guide may be specified as a parameter, notably:

outputFormat

Format of the output file: `rtf`, `wml`, `docx` or `odt`. Default: `rtf`. Note that command-line utility **fo2wml** automatically sets `outputFormat` to `wml`, **fo2docx** automatically sets `outputFormat` to `docx` and **fo2odt** automatically sets `outputFormat` to `odt`.

outputEncoding

Specifies the output encoding. Supported values depend on the target output format:

- For RTF output, supported values are `ASCII`, `Cp1250` (Windows Eastern European), `Cp1251` (Windows Cyrillic) and `Cp1252` (Windows Latin-1). The default value is `Cp1252` (Windows Latin-1).
- For WML output, all encodings available in the current JVM are supported. The option value may be either the encoding name (e.g. `ISO8859_1`) or the charset name (e.g. `ISO-8859-1`). The default value is `Cp1252` (Windows Latin-1).
- For Open XML output (`.docx`), this option specifies the encoding of XML content in the output document. Supported values are `UTF-8` and `UTF-16`. The default value is `UTF-8`.
- For OpenDocument output (`.odt`), this option specifies the encoding of XML content (files `styles.xml` and `content.xml`) in the output document. All encodings available in the current JVM are supported. The option value may be either the encoding name (e.g. `ISO8859_1`) or the charset name (e.g. `ISO-8859-1`). The default value is `UTF8`.

imageResolution

Default image resolution in DPI. A positive integer. Used to compute the intrinsic size of an image, but only when an image file does not contain resolution or absolute size information.

Default value: 96.

prescaleImages

Image scaling policy. `true` or `false`. Default: `false`.

Specify `prescaleImages=true` to minimize output document size. By default (`prescaleImages=false`), the original size of images is preserved and scaling directives are inserted in the output document.

Note that:

- Property `prescaleImages=true` will never create an image which has larger dimensions than the original image. It can only create an image which has smaller dimensions than the original image.
- Property `prescaleImages=true` is honored only for true raster graphics. Vector graphics (WMF, EMF) are never prescaled. Pre-rasterized vector graphics (SVG, MathML) are always prescaled (by the competent renderer, e.g. Batik or JEuclid, not by XMLmind XSL-FO Converter itself).

genericFontFamilies

May be used to map the generic font families `serif`, `sans-serif`, `monospace`, `fantasy` and `cursive` to actual font families.

Syntax:

```
map -> entry [',' entry]*  
  
entry -> generic_family '=' actual_family
```



```
generic_family -> 'serif' | 'sans-serif' | 'monospace'
                | 'cursive' | 'fantasy'
```

Example: `"-genericFontFamilies=fantasy=Impact,cursive=Comic Sans MS"`.

The default mapping depends on the output format: the generic font families `serif`, `sans-serif`, `monospace` are mapped to "Times New Roman", Arial, "Courier New" for RTF, WML and Open XML (.docx) and to "DejaVu Serif", "DejaVu Sans", "DejaVu Sans Mono" for OpenDocument (.odt).

Note that by default, generic font families `fantasy` and `cursive` are not mapped.

`set.graphic_factory_name.parameter_name`

Sets parameter `parameter_name` on graphic factory called `graphic_factory_name` (case-insensitive). A graphic factory is a software component in charge of processing one or more graphic formats. Examples of such graphic factories: ImageIO, WMF, EMF, SVG, MathML. Only few graphic factories may be parameterized this way.

Table B.1. Graphic factory parameters

| <code>graphic_factory_name</code> | <code>parameter_name</code> | Value | Default | Description |
|-----------------------------------|-----------------------------|-----------------------|---------|--|
| SVG | <code>resolution</code> | DPI, positive integer | 192 | Resolution used to convert SVG vector graphics to PNG raster images. |
| MathML | <code>resolution</code> | DPI, positive integer | 288 | Resolution used to convert MathML equations (may be seen as vector graphics) to PNG raster images. |
| | <code>mathsize</code> | pt, positive integer | 12 | The base font size of MathML equations. |

Examples:

```
-set.svg.resolution=300
-set.MathML.mathsize=11
-set.mathml.resolution=300
```

`singleSidedLayout`

Specifies single-sided page layout. By default RTF, WML and Open XML (.docx) output documents are given a double-sided page layout regardless of the input document properties. This option may be set to `true` to force a single-sided page layout.

`styles`

Specifies the location of an XML file containing the set of *user styles* to be used during the conversion.

This location is an URL in its string form (e.g. "file:///C:/My%20Folder/styles.xfc") or a filename (e.g. "C:\My Folder\styles.xfc"). A relative filename is relative to the current working directory.

The XML file must conform to the `styles.xsd` schema.

By default, XMLmind XSL-FO Converter generates only direct formatting (RTF, WordprocessingML, .docx) or automatic styles (.odt).

2. Apache FOP

Any of the options documented in Apache FOP: Configuration may be specified as a parameter. Example: `source-resolution`.

In addition, the following *pseudo-parameters* are also supported:

| Parameter | Value | Description |
|----------------------------|---|---|
| <code>renderer</code> | <code>pdf ps pcl svg xml mif txt</code> | Specifies which renderer to use. If this pseudo-parameter is absent, which renderer to use is guessed from the extension of the output file name. |
| <code>configuration</code> | Absolute URL or filename | Specifies the absolute URL or filename of a FOP user configuration file. Such configuration files are useful to specify font metrics, hyphenation files, etc. See also Section 6.4, "FOP preferences" [35]. |

3. RenderX XEP

Any of the options documented in the XEP User Guide may be specified as a parameter. Example: `PS.LANGUAGE_LEVEL`.

In addition, the following pseudo-parameters are also supported:

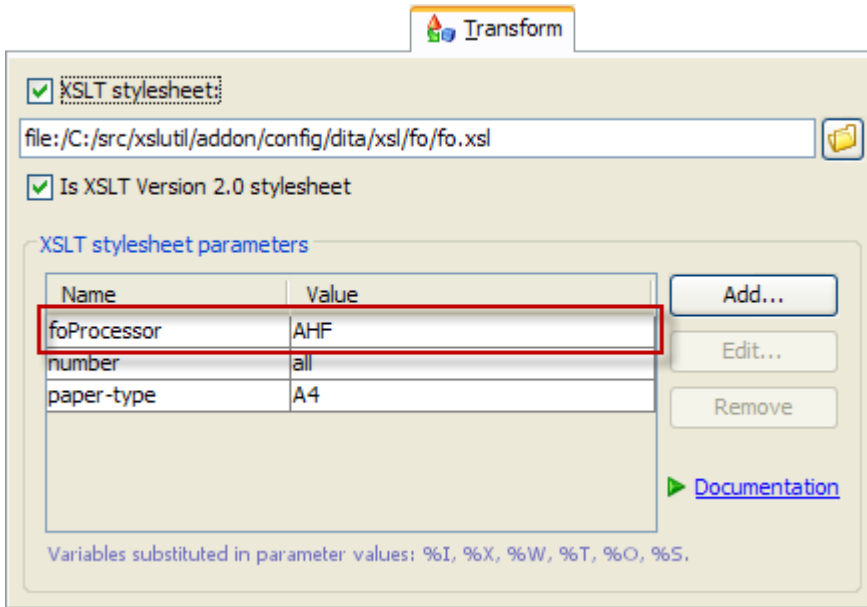
| Parameter | Value | Description |
|----------------------------|---|--|
| <code>OUTPUT_FORMAT</code> | <code>pdf ps xps afp svg html ppml xep</code> (Support of the AFP, Microsoft XPS, PPML, HTML and SVG formats requires getting the corresponding add-ons from RenderX.) | Specifies the target format of XEP. If this pseudo-parameter is absent, which target format to use is guessed from the extension of the output file name. |

C. Using Antenna House Formatter as your XSL-FO processor

1. In the **Transform** tab, add an XSLT stylesheet parameter specifying that you are using Antenna House Formatter.

| Conversion Name | Parameter Name | Parameter Value |
|-----------------|----------------|-----------------|
| ditaToxxx | foProcessor | AHF |
| dbToxxx | axf.extensions | 1 |
| db5toxxx | axf.extensions | 1 |
| xhtmlToxxx | foProcessor | AHF |

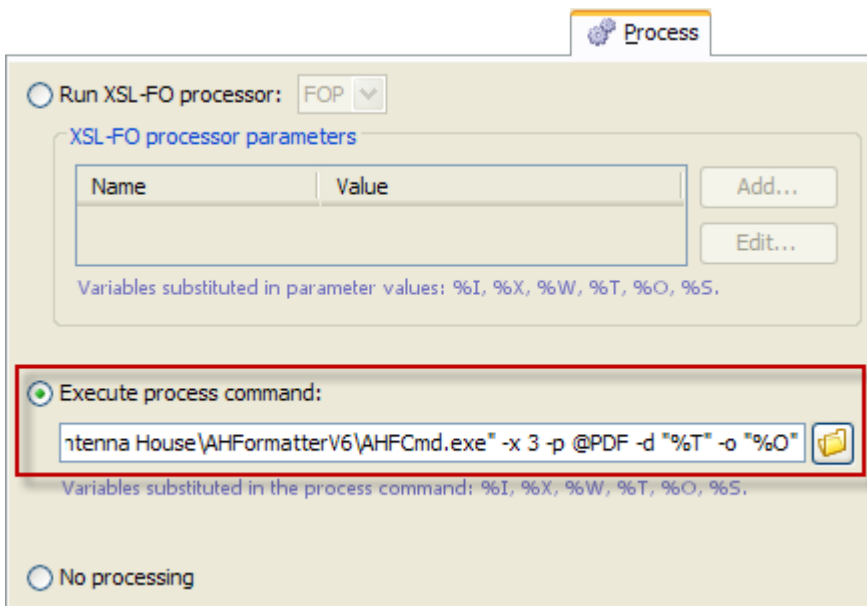
Figure C.1. DitaToxxx example: add the foProcessor XSLT stylesheet parameter



- In the **Process** tab, check the **"Execute process command"** checkbox and specify a command-line resembling to:

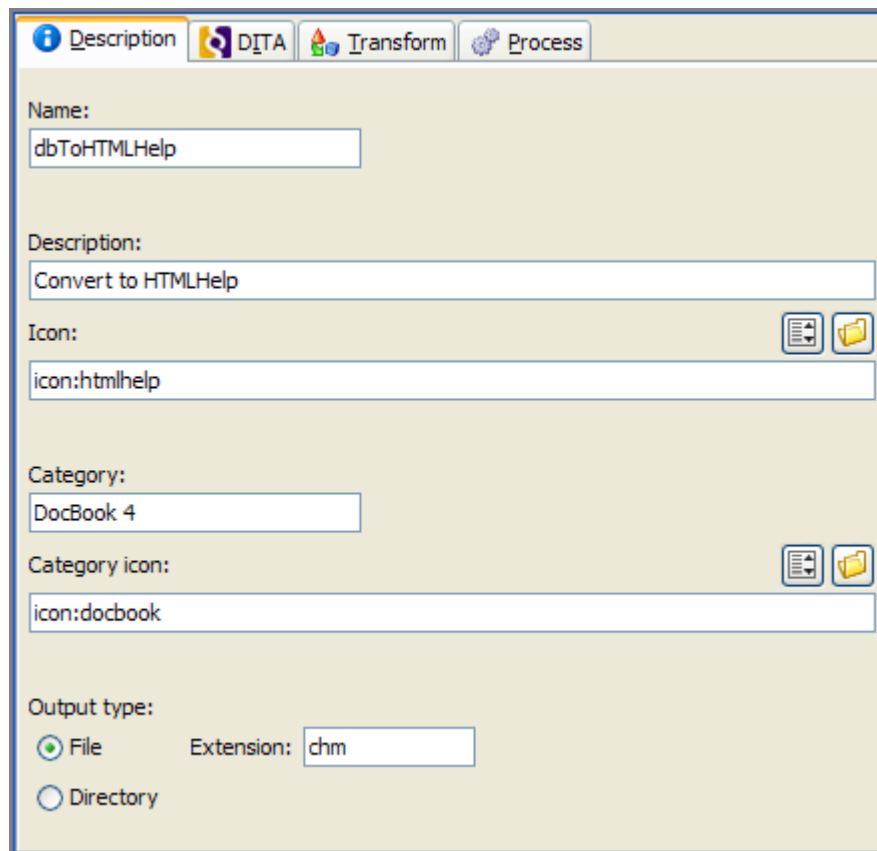
```
"C:\Program Files\Antenna House\AHFormatterV6\AHFCmd.exe" -x 3 -p @PDF -d "%T" -o "%O"
```

Figure C.2. DitaToxxx example: specify the command-line of Antenna House Formatter



D. Using XMLmind XSL Utility to convert a DocBook 4 document to HTML Help (.chm file)

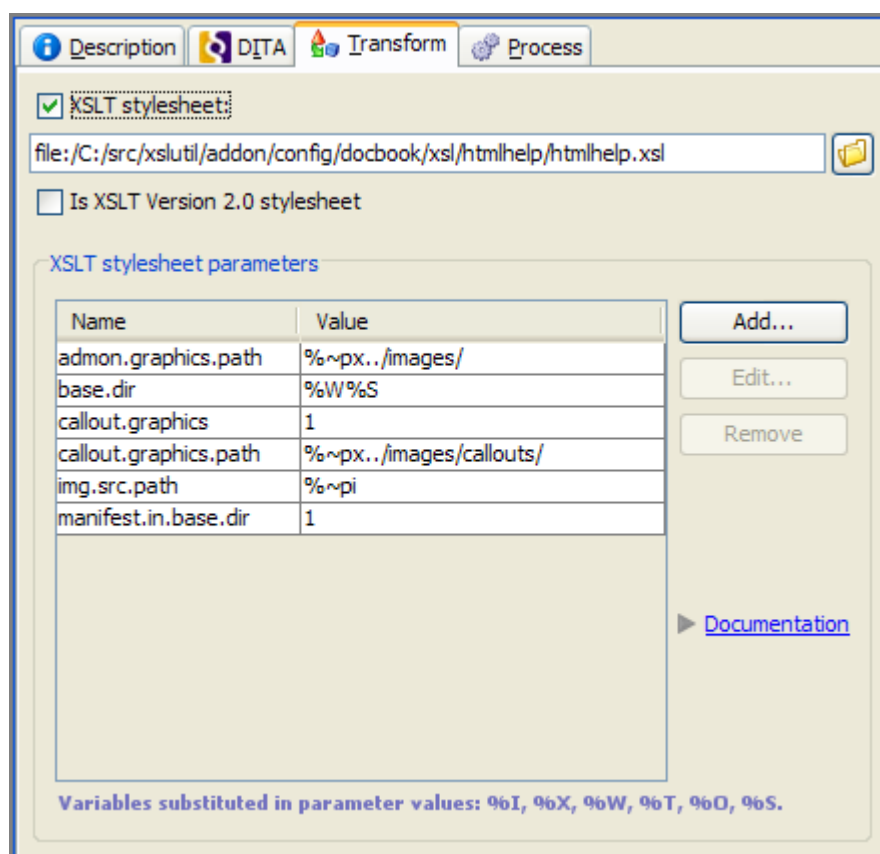
1. Conversion specification `dbToHTMLHelp` directly generates a `.chm` file.



The screenshot shows the configuration window for the 'dbToHTMLHelp' conversion specification. The window has a title bar with tabs for 'Description', 'DITA', 'Transform', and 'Process'. The 'Description' tab is active. The configuration fields are as follows:

- Name:** `dbToHTMLHelp`
- Description:** `Convert to HTMLHelp`
- Icon:** `icon:htmlhelp` (with browse and folder icons)
- Category:** `DocBook 4`
- Category icon:** `icon:docbook` (with browse and folder icons)
- Output type:** File Extension: `chm`
 Directory

2. The XSLT stylesheet used to convert DocBook 4 documents is found in `xslUtility_install_dir/addon/config/docbook/xsl/htmlhelp/htmlhelp.xsl`.



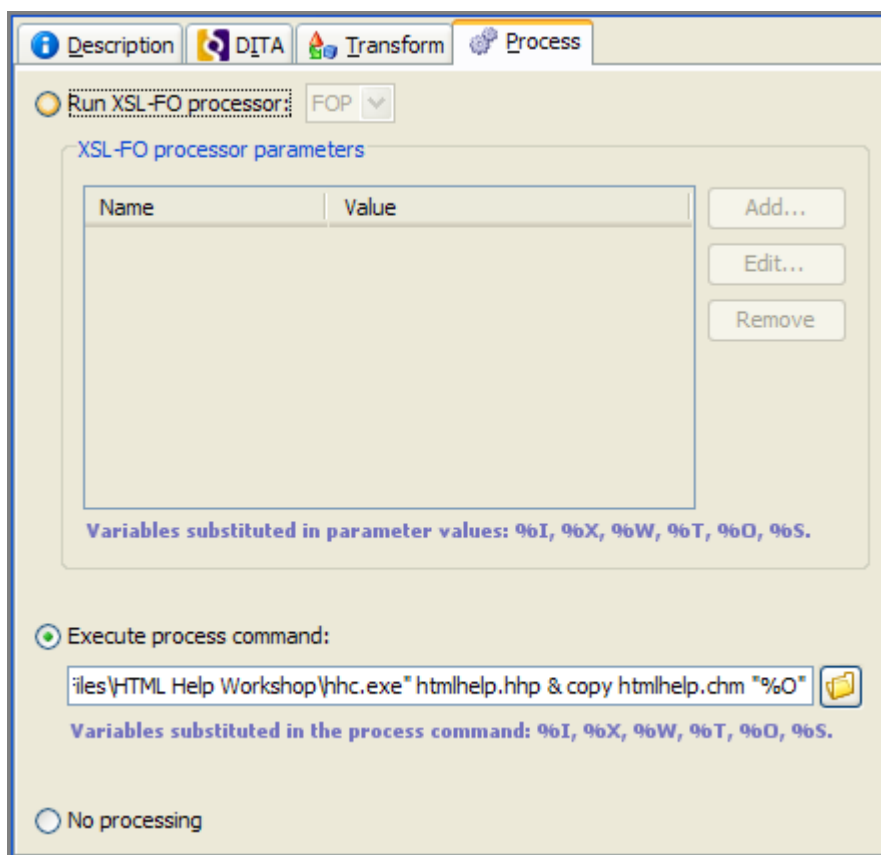
Conversion specification `dbToHTMLhelp` generates intermediate HTML files in the temporary working directory (variable `%w`) and not in the directory containing your input file. That's why you have to specify *all* the following XSLT stylesheet parameters:

| Name | Value |
|------------------------------------|--------------------------------------|
| <code>admon.graphics.path</code> | <code>%~px../images/</code> |
| <code>base.dir</code> | <code>%W%S</code> |
| <code>callout.graphics.path</code> | <code>%~px../images/callouts/</code> |
| <code>img.src.path</code> | <code>%~pi</code> |
| <code>manifest.in.base.dir</code> | <code>1</code> |

Other XSLT stylesheet parameters such as `callout.graphics=1` are optional.

The reference documentation of these parameters is found in *DocBook XSL Stylesheet Reference Documentation*.

- Conversion specification `dbToHTMLhelp` needs to execute a sequence of commands in order to copy some graphics files to the output directory and to run the HTML help compiler, `hhc.exe`, on the output directory.



This sequence of commands depends on:

- the paths of the directories containing the image files referenced by your input document;
- where **hhc.exe** (part of HTML Help Workshop) is installed on your computer.

Let's suppose:

- the image files referenced by your input document are found in the `screenshots/` subdirectory (relative to your input document);
- **hhc.exe** is installed in `C:\Program Files\HTML Help Workshop\`.

```
mkdir screenshots❶ &❷
copy "%~pI\screenshots\*.png" screenshots &❸
mkdir images &❹
mkdir images\callouts &
copy "%~pX\..\images\*.png" images &❺
copy "%~pX\..\images\callouts\*.png" images\callouts &
"C:\Program Files\HTML Help Workshop\hhc.exe" htmlhelp.hhp &❻
copy htmlhelp.chm "%O"❼
```

Note that the above sequence of commands works fine because *a process command is always executed in the temporary working directory* (variable `%W`).

- ❶ Create a `%W\screenshots\` subdirectory. All the images files referenced by your XML input file will be copied to this subdirectory.
- ❷ '&' is the command separator on Windows.

- ③ Copy all the image files referenced by your XML input file to the `%W\screenshots\` subdirectory.
- ④ Create an `%W\images\` and `%W\images\callouts\` subdirectories. All the images files referenced by the DocBook XSL stylesheets will be copied to these subdirectories.

In the case of the DocBook XSL stylesheets, these images resources are actually found in an `images\` directory, which is a sibling of the directories containing the stylesheets generating HTML Help.

- ⑤ Copy all the image resources referenced by the DocBook XSL stylesheets to the `%W\images\` and `%W\images\callouts\` subdirectories.
- ⑥ Run the HTML help compiler, `hhc.exe`, on `htmlhelp.hhp`, which is the file generated by the DocBook XSL stylesheets.
- ⑦ Doing so creates `htmlhelp.chm`. Copy this file to the desired output file.

E. Support of XInclude in XMLmind XSL Utility

Unlike XMLmind XML Editor, XMLmind XSL Utility delegates to Xerces, its XML parser, the task of transcluding XInclude elements. This means that, compared to XMLmind XML Editor, XMLmind XSL Utility has a number of limitations related to the support of XInclude. The two main limitations are:

- A XML document cannot include a part of itself. That is, in an `xi:include` element, the `href` attribute cannot be empty or missing.
- The `xpointer` attribute of an `xi:include` element cannot refer to the `ID` of the included element, *unless*
 - the document containing the included element conforms to a DTD (specified in this document using `<!DOCTYPE>`);
 - *OR* the `ID` attribute is `xml:id` (like in DocBook 5¹).

When this is not the case, you must limit yourself to including the root element (`xpointer="element(/1)`) of the document modules.

¹DocBook 4 is specified by a DTD. DocBook 5 is specified by a RELAX NG schema.