

---

# XMLmind XSL Utility - Online Help

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

March 8, 2010

1. Overview .....	1
2. Running XMLmind XSL Utility .....	2
2.1. System requirements .....	2
2.2. Installation .....	2
2.3. Contents of the installation directory .....	3
2.4. Starting XMLmind XSL Utility .....	3
2.5. XMLmind XSL Utility as a command-line tool .....	3
3. Converting an XML document to another format .....	4
3.1. Canceling the current conversion process .....	5
4. Specifying a conversion .....	5
4.1. Specifying the conversion of a DITA topic or map .....	10
4.1.1. Reference of the fields found in the DITA tab .....	10
5. User preferences .....	12
5.1. General preferences .....	12
5.2. Helper applications preferences .....	13
5.2.1. The "Helper Application Editor" dialog box .....	14
5.3. FOP preferences .....	16
5.4. XEP preferences .....	17
A. Variables .....	17
B. XSL-FO processor parameters .....	19
1. XMLmind XSL-FO Converter (XFC) .....	19
2. Apache FOP .....	19
3. RenderX XEP .....	19
C. Using XMLmind XSL Utility to convert a DocBook 4 document to HTML Help (.chm file) .....	19
D. Support of XInclude in XMLmind XSL Utility .....	21

## 1. Overview

XMLmind XSL Utility is a graphical application (see screenshot [4]) written in Java™ integrating two XSLT engines:

- Saxon 6, an XSLT 1 transformation engine,
- Saxon 9, an XSLT 2 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, XMLmind XSL Utility allows to convert DITA 1.1, DocBook 4, DocBook 5, and XHTML documents to PDF, RTF, WordprocessingML, Office Open XML and OpenOffice formats.

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 process<sup>1</sup>:

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 C, *Using XMLmind XSL Utility to convert a DocBook 4 document to HTML Help (.chm file)* [19] 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.

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

- Sun or Apple Java™ runtime 1.5 or above.
- 100Mb of free disk space.

XMLmind XSL Utility is officially supported on Windows XP/Vista/7, on Linux 2.6 and on Mac OS X 10.4 (Tiger), 10.5 (Leopard), 10.6 (Snow Leopard). It is possible to use it on other Java™ 1.5+ platforms (e.g. Solaris), but without support from XMLmind.

### 2.2. Installation

Simply unzip the distribution somewhere. Linux/Mac example:

```
~$ cd /opt
/opt$ unzip /tmp/xslutil-4_3_2.zip
/opt$ ls xslutil-4_3_2
addon/
bin/
doc/
legal.txt
legal/
```

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

---

<sup>1</sup>Conversion of DITA documents works differently.

## 2.3. Contents of the installation directory

addon/

Contains XMLmind XML Editor configurations (DITA 1.1, DocBook 4, DocBook 5, XHTML) and plug-ins (FOP, Batik, JEuclid, Jimi, XEP, XFC). (As of v4.3, XMLmind XSL Utility is based on the add-on architecture of XMLmind XML Editor.)

bin/xslutil.exe, xslutilc.bat

Executable file and .bat file used to run XMLmind XSL Utility on Windows. More information about xslutilc.bat in Section 2.5, “XMLmind XSL Utility as a command-line tool” [3].

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 Open-Office formats.

legal.txt, legal/

Contains XMLmind XSL Utility licences as well as the licences 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-4_3_2\bin\xslutil
```

After testing that it works, you may want to add a shortcut to C:\xslutil-4\_3\_1\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-4_3_2/bin/xslutil &
```

## 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 arguments, XMLmind XSL Utility is a 100% graphical application.
- 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 [4]:

```
xslutilc dbToDocx E:\src\4xe\docsrc\xhtml\help.xml E:\tmp\help.docx
```

### Important

On Windows, make sure to use xslutilc.bat and not xslutil.exe.

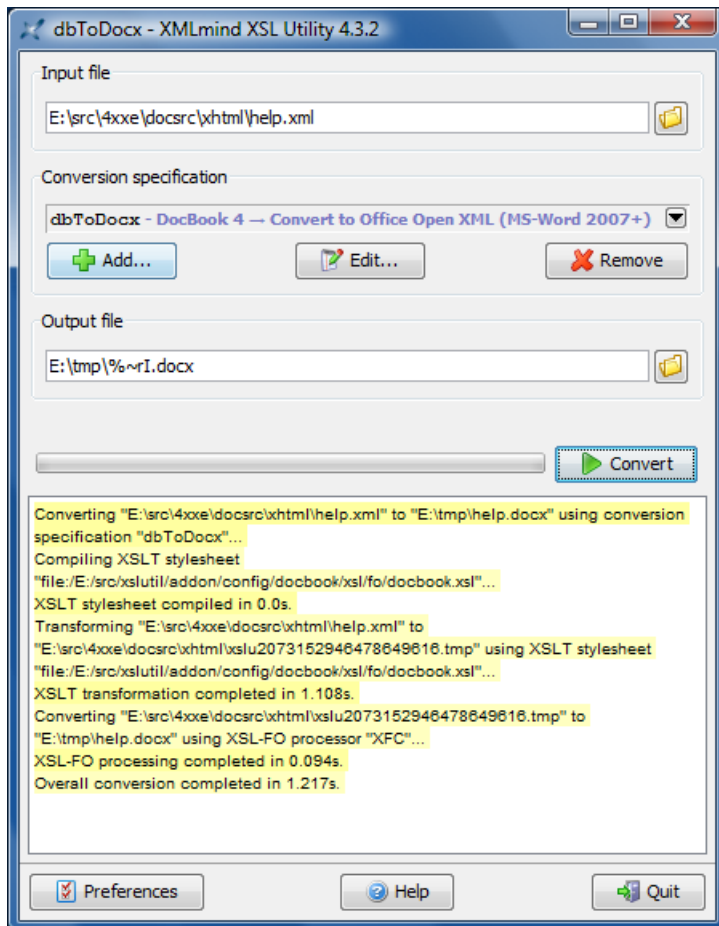
Linux/Mac example:

```
xslutil dbToDocx /home/john/src/4xxe/docsrc/xhtml/help.xml /home/john/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.

### 3. Converting an XML document to another format

**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.

#### Caution

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 document to RTF. Do not choose `db5ToRTF` because this will give unexpected results.

## Tip

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.

## Tip

The filename specified in Output file text field may reference the %I variable. (Appendix A, *Variables* [17] 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. Click the Convert button.

## 3.1. Canceling the current conversion process

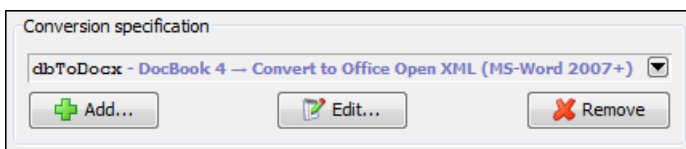
After the conversion process is started, the Convert button becomes the Cancel button and allows you to stop the conversion.

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. Specifying a conversion

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





### Procedure 2. 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.

- 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.

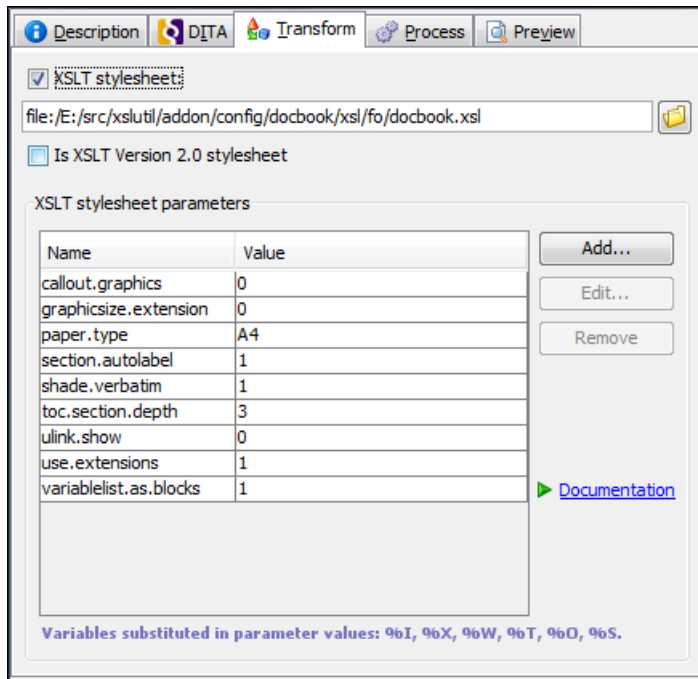
This icon may be selected from a predefined list (button ) or found in an external file (button .

- 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 an output file and in such case, specify the standard extension used for such file in the Extension text field.

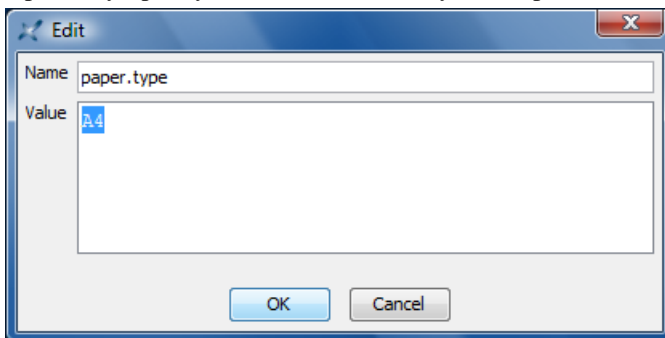
If your conversion requires/creates an output directory, check the Directory radio button. In this case, note that:

- 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 [6]) and/or to the process command (see the process step [8]) by referencing the `%o` variable in an XSLT style parameter value and/or in the process command.

3. 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 4.1, "Specifying the conversion of a DITA topic or map" [10].
4. Specify the transform step, if any.



- 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.
- Check the "Is XSLT Version 2.0 stylesheet" checkbox if the stylesheet you have chosen conforms to the XSLT Version 2 standard.
- 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* [17] describes all the variables supported by XMLmind XSL Utility.)

## Tip

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 most XSLT stylesheets are not documented and/or do not support any parameter.

5. Specify the process step, if any.

The screenshot shows the 'Process' dialog box with the following details:

- Buttons: Description, DITA, Transform, Process, Preview
- Selected radio button: Run XSL-FO processor
- Dropdown menu: XFC
- Section: XSL-FO processor parameters
- Table:
 

Name	Value
imageResolution	120
outputEncoding	UTF-8
outputFormat	docx
prescaleImages	false
- Buttons: Add..., Edit..., Remove
- Note: Variables substituted in parameter values: %I, %X, %W, %T, %O, %S.
- Radio button: Execute process command (unselected)
- Text field: (empty)
- Note: Variables substituted in the process command: %I, %X, %W, %T, %O, %S.
- Radio button: No processing (unselected)

- 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 (examples: PassiveTeX, Antenna House XSL Formatter) or when your conversion requires a specific post-transformation processing (see Appendix C, *Using XMLmind XSL Utility to convert a DocBook 4 document to HTML Help (.chm file)* [19] 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* [17] describes all the variables supported by XMLmind XSL Utility.)

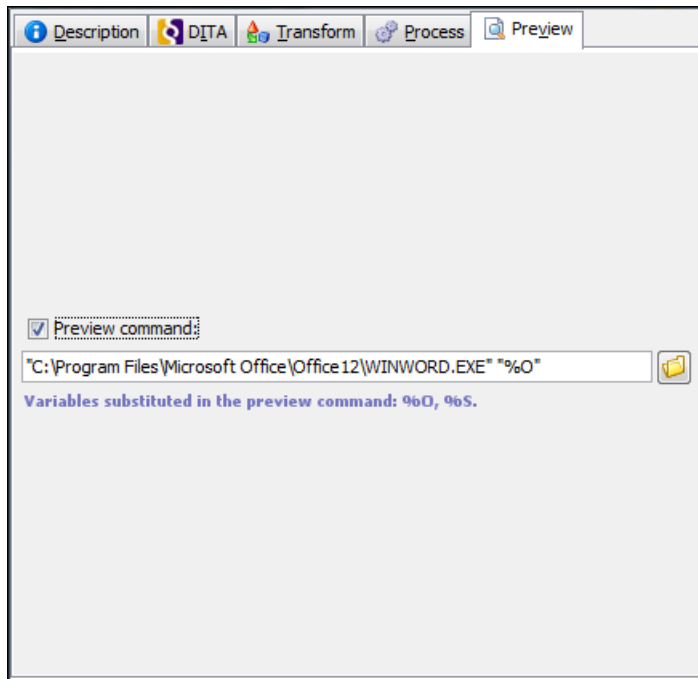
Finally, there are many cases where the transform step is all what you need (examples: conversion to HTML, Eclipse Help). In such case, simply check the No processing radio button.

- 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* [19].

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* [17] describes all the variables supported by XMLmind XSL Utility.)

6. Optionally specify a preview command.



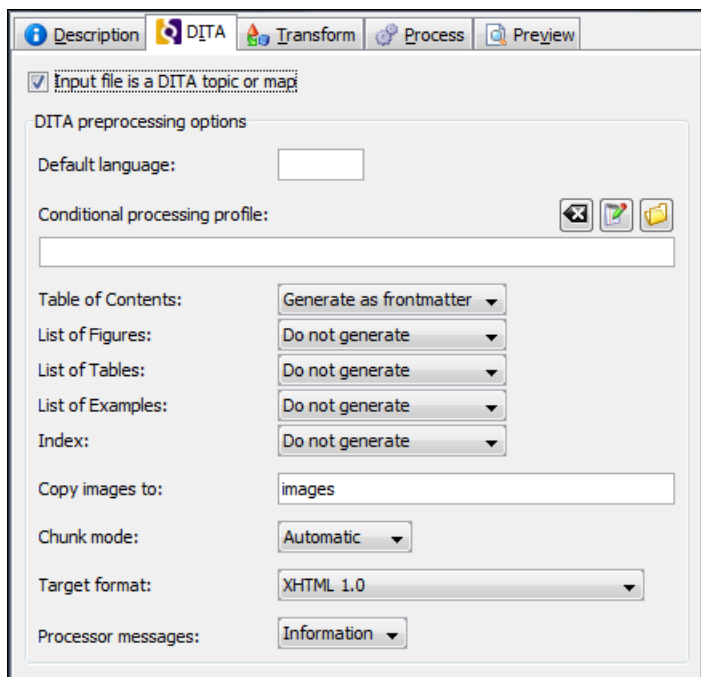
- a. Check the Preview command checkbox.
- b. Directly type the command in the corresponding text field (Windows example: `start "" "%O"`). Alternatively, you may use the button next to the text field to pick a helper application (example: `acroread`).

This 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 preview command may reference one or more of the following variables: %O, %S. (Appendix A, *Variables* [17] describes all the variables supported by XMLmind XSL Utility.)

## 4.1. Specifying the conversion of a DITA topic or map

Figure 3. The DITA tab



### Procedure 3. 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 [11]. For example, specify "images".
3. Select the format of the generated document from the "Target format" combobox [11].

#### 4.1.1. 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 4.1	<code>xslutil_install_dir/addon/config/dita/xsl/xhtml/html.xsl</code>
Java Help (.jar)	<code>xslutil_install_dir/addon/config/dita/xsl/javahelp/java-help.xsl</code>
HTML Help (.chm)	<code>xslutil_install_dir/addon/config/dita/xsl/htmlhelp/html-help.xsl</code>
Eclipse Help	<code>xslutil_install_dir/addon/config/dita/xsl/eclipsehelp/eclipsehelp.xsl</code>
EPUB	<code>xslutil_install_dir/addon/config/dita/xsl/epub/epub.xsl</code>
PostScript	<code>xslutil_install_dir/addon/config/dita/xsl/fo/fo.xsl</code>

Format	XSLT Stylesheet
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 preprocessor. Allowed values are (from not verbose to very verbose): None, Information, Verbose, Debug.

## 5. User preferences

The user preferences are stored in the following directory:

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

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

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.

### 5.1. General preferences

#### 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.

## 5.2. Helper applications preferences

This preferences sheet allows to specify the locations of respectively, `jhindex` and `hhc.exe`, two external tools, which are needed to convert DITA documents to respectively, Java™ Help and 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.

Default: the "text/plain" file type:

- On Windows: `text/plain:txt:::notepad "%F"`
- On the Mac:
  - for a version older than Mac OS X 10.5 (e.g. Tiger): `open -t "%F"`,
  - starting from Mac OS X 10.5 (Leopard): `open -W -n -t "%F"`.

See also Helper applications on the Mac [15].

- On Unix: `text/plain:txt:::xterm -e vi "%F"`

Buttons acting on this list:

#### Add

Displays the "Helper Application Editor" dialog box [14] in order to add a new file type to the list.

#### Edit

Displays the "Helper Application Editor" dialog box [14] 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. By default: `(mozilla -remote "openURL(%U)" 1> /dev/null 2>&1) | (mozilla "%U" &)`

Buttons acting on 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 [15].

### 5.2.1. The "Helper Application Editor" dialog box

This dialog box allows to view or modify a file type listed in the Helper applications preferences sheet [13].



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

**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 [13].

### MIME type

One or more MIME types (see definition [14] above) separated by spaces.

### Filename extension

One or more extensions (see definition [14] above) separated by spaces.

### Magic string

One or more magic strings (see definition [15] above) separated by spaces.

### XML name pattern

One or more XML name patterns (see definition [15] above) separated by spaces.

### 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:

- for a version older than Mac OS X 10.5 (e.g. Tiger): "open -a",
- starting from Mac OS X 10.5 (Leopard): "open -W -n -a".

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.

In practice, this means that, on Mac OS X versions older than 10.5, a helper application can only be used for viewing purposes and this, even if the helper is an editor.

### 5.3. FOP preferences

By default, only the 14 built-in fonts: Times, Helvetica, Courier, Symbol and ZapfDingbats are used in the generated PDF. This form 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 4. 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 5. How to choose specific fonts (for example, you want to replace Times fonts by Georgia fonts)**

1. Click Add.
  - a. Choose the .ttf file containing font Georgia.
  - 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.
6. Restart XMLmind XSL Utility.

### Note

It is recommended to also specify fonts replacing Helvetica, that is, fonts having a sans-serif alias and fonts replacing Courier, that is fonts having a monospace alias.

## 5.4. XEP preferences

XEP preferences are identical to FOP preferences [16].

### Note

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.

## A. Variables

Variable	Substituted in	Value
<code>%I</code>	Output file field, XSLT stylesheet parameter value, XSL-FO processor parameter value, process command.	The input file.
<code>%X</code>	XSLT stylesheet parameter value, XSL-FO processor parameter value, process command.	The XSLT stylesheet file. 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.  Note that the value of variable <code>%X</code> may be the empty string and that, at the same time, the value of variable <code>%X</code> may be a valid URI. See the URI counterparts below.
<code>%W</code>	XSLT stylesheet parameter value, XSL-FO processor	A temporary working directory always created by a conversion, just in case it could be useful.

Variable	Substituted in	Value
	parameter value, process command.	Not available when the "Input file is a DITA topic or map" checkbox is checked.
%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"> <li>When there is no transform step, the value of this variable is the empty string.</li> <li>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.</li> </ul> <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 command, preview command.	The output file or directory.
%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/1.74.0/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

Modifier	Description	Example
		%~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. Example: `outputEncoding`.

### 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 5.3, "FOP preferences" [16].

### 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</code>	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 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 'Description' dialog box for the conversion specification 'dbToHTMLHelp'. The dialog has a title bar with icons for Description, DITA, Transform, Process, and Preview. The main area contains the following fields:

- Name:** dbToHTMLHelp
- Description:** Convert to HTML Help
- Icon:** icon:htmlhelp
- Category:** DocBook 4
- Category icon:** icon:docbook
- Output type:**
  - File      Extension: chm
  - Directory

2. Conversion specification `dbToHTMLHelp` generates intermediate HTML files in the temporary working directory (variable `%W`).

The screenshot shows the 'XSLT stylesheet' dialog box for the conversion specification 'dbToHTMLHelp'. The dialog has a title bar with icons for Description, DITA, Transform, Process, and Preview. The main area contains the following fields and controls:

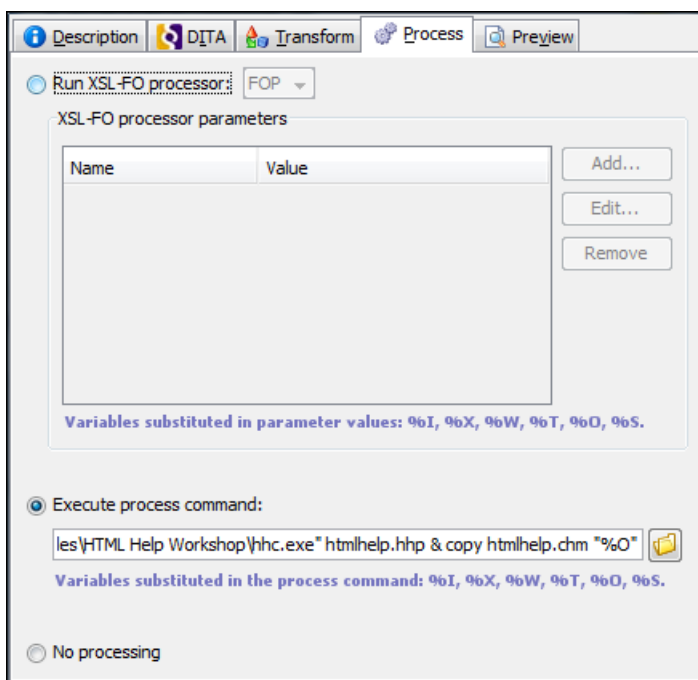
- XSLT stylesheet:
- file:/E:/src/xslutil/addon/config/docbook/xsl/htmlhelp/htmlhelp.xsl
- Is XSLT Version 2.0 stylesheet
- XSLT stylesheet parameters**

Name	Value
base.dir	%W%S
manifest.in.base.dir	1

Buttons: Add..., Edit..., Remove

[Documentation](#)
- Variables substituted in parameter values: %I, %X, %W, %T, %O, %S.

3. 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 is:

```
mkdir icons1 &2
copy "%~pI\icons\*.png" icons3 &
copy "%~pX\..\icons\*.png" icons4 &
"C:\Program Files\HTML Help Workshop\hhc.exe" htmlhelp.hhp5 &
copy htmlhelp.chm "%O"6
```

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

- 1** Create an %w\icons\ subdirectory. All the graphics files will be copied to this subdirectory.
- 2** '&' is the command separator on Windows.
- 3** Copy all the graphics files referenced by the XML input file to the %w\icons\ subdirectory.

This command assumes that all the graphics files referenced by the XML input file are found in an icons\ subdirectory of the input directory.

- 4** Copy all the graphics resources referenced by the XSLT stylesheet to the %w\icons\ subdirectory.

In the case of the DocBook XSL stylesheets, these graphics resources are actually found in an icons\ directory, which is a sibling of the directories containing the stylesheets.

- 5** Run the HTML help compiler, hhc.exe, on htmlhelp.hhp, which is the file generated by the DocBook XSL stylesheet.
- 6** Doing so creates htmlhelp.chm. Copy this file to the desired output file.

## D. 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 5<sup>1</sup>).

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

---

<sup>1</sup>DocBook 4 is specified by a DTD. DocBook 5 is specified by a RELAX NG schema.