# w2x-app - Online Help

Hussein Shafie, XMLmind Software <w2x-support@xmlmind.com>

June 20, 2023

**Abstract**

This document contains the online help of w2x-app, a graphical application which is easier to use than the **w2x** command-line utility.

# Table of Contents



## 1. Starting w2x-app

### 1.1. After installing the native distribution (`setup.exe` on Windows, `.dmg` on the Mac)

On Windows, double-click  , the **XMLmind Word To XML** icon.

On the Mac, double-click  , the **WordToXML** icon.

## 1.2. After installing the `.zip` distribution

Application w2x-app is intended to be used directly from the directory created by unzipping its distribution.

On Windows, you can start w2x-app by double-clicking *`w2x_install_dir`*`\bin\w2x-app.exe`[1].

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

```
/opt$ w2x-1_10_0/bin/w2x-app &
```

### Running w2x-app on a computer having a very high resolution (HiDPI) screen

w2x-app 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:

```
w2x-app -putpref displayScaling 200
```

Preference key `displayScaling` may be used to globally change the size of all the items comprising the user interface of w2x-app. More information [22].

### 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 This is needed because on Linux, the “system” Look & Feel (called "`Metal`") looks rather outdated.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 w2x-app by running

```
w2x-app -putpref lookAndFeelClassName fallback
```

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

```
w2x-app -delpref lookAndFeelClassName
```

[1] Alternatively, type the following command in a command prompt and then press Enter:

```
C:\> w2x-1_10_0\bin\w2x-app-c.bat
```

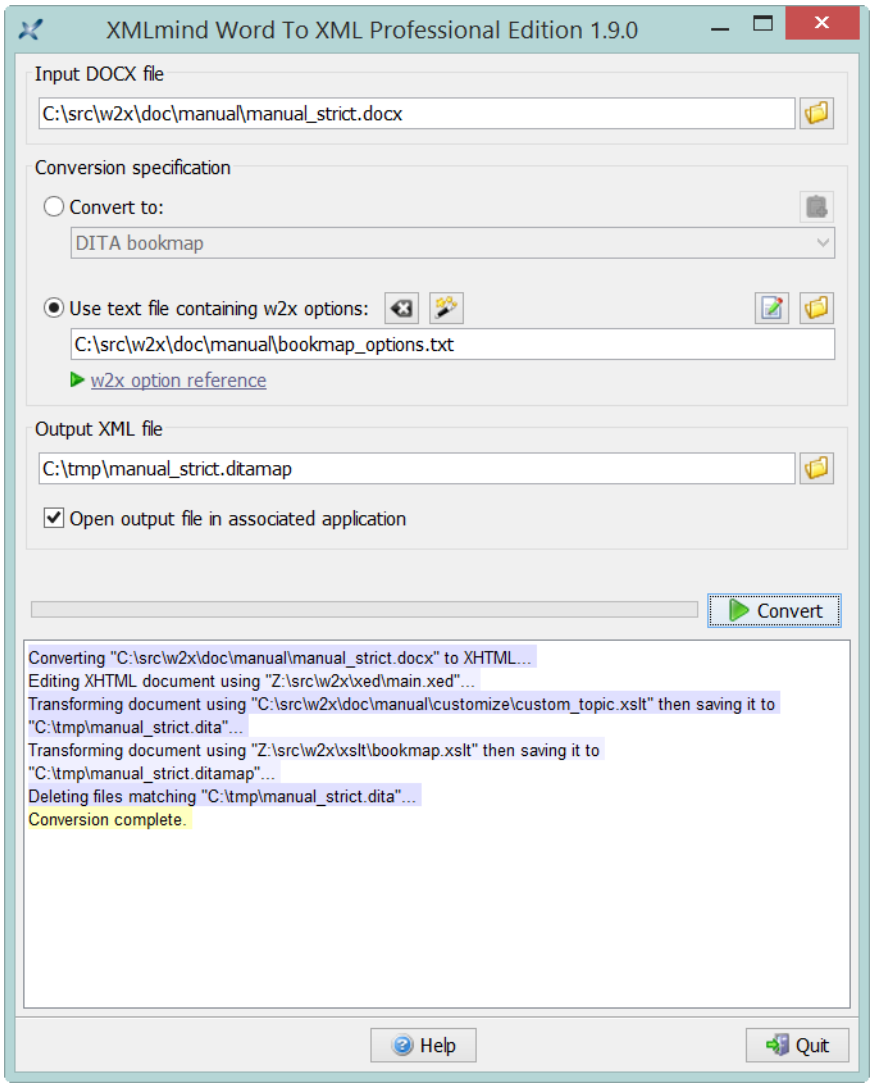
Use `w2x-app-c.bat` on Windows , but only when you need to start w2x-app with a console. On Windows, a console is needed to be able to see low-level error messages.

# 2. Converting a DOCX file

### About Evaluation Edition

Do not be surprised because XMLmind Word To XML Evaluation Edition generates output containing random words replaced by string "`[XMLmind]`". Of course, this does not happen with Professional Edition!

*Figure 1. w2x-app main window*




Procedure:

1. In the "**Input DOCX file**" frame, click to select the "`.docx`" file which is to be converted to XML.

   This "`.docx`" file must have been created using MS-Word 2007+.

2. In the "**Conversion specification**" frame,

- Click "**Convert to**" to select predefined conversions then choose a predefined conversion from the combobox.

| Name | Description |
|---|---|
| DITA bookmap | A DITA `bookmap` containing possibly nested `chapter` and `topicref` elements, themselves referencing files containing `topic` elements. |
| DITA concept | A DITA `concept` possibly containing nested `concept` elements. |
| DITA map | A DITA `map` containing possibly nested `topicref` elements, themselves referencing files containing `topic` elements. |
| DITA topic | A DITA `topic` possibly containing nested `topic` elements. |
| DocBook V4.5 article | DocBook v4.5 `article` possibly containing CALS (not HTML) tables. |
| DocBook V4.5 book | DocBook v4.5 `book` possibly containing CALS (not HTML) tables. |
| DocBook V5.0 article | DocBook v5.0 `article` possibly containing CALS (not HTML) tables. |
| DocBook V5.0 book | DocBook v5.0 `book` possibly containing CALS (not HTML) tables. |
| DocBook V5.1 assembly | DocBook v5.1 `assembly` in which `topics` possibly contain CALS (not HTML) tables. |
| EPUB 2.0 containing styled XHTML | Similar to "Multi-page styled (X)HTML [4]" except that the generated XHTML pages are packaged as an EPUB 2.0 book.<br><br>Output file extension must be "`.epub`". |
| EPUB 2.0 containing “semantic” XHTML 1.1 | Similar to "Multi-page “semantic” (X)HTML 1.1 [5]" except that the generated XHTML pages are packaged as an EPUB 2.0 book.<br><br>Output file extension must be "`.epub`". |
| Multi-page styled (X)HTML | Similar to "Single-page styled (X)HTML [6]" except that the source DOCX document is automatically split into parts.<br><br>A new part is created each time a paragraph having an outline level less than or equal to specified *`split-before-level`* parameter is found in the source. An outline level is an integer between 0 (e.g. style **Heading 1**) and 8 (e.g. style **Heading 9**). The default value of parameter *`split-before-level`* is 0, which means: for each **Heading 1**, create a new page starting with this **Heading 1**.<br><br>The file specified in the "**Output XML file**" field will contain the generated frameset. Output file extension must be "`.html`".<br><br>While an obsolete HTML feature, a frameset makes it easy browsing these HTML pages. Moreover the table of contents used as the left frame is a convenient way to programmatically list all the generated HTML pages. |

| Name | Description |
|---|---|
| Multi-page “semantic” (X)HTML 1.0 Strict | Similar to "Single-page “semantic” (X)HTML 1.0 Strict [6]" except that the intermediate, automatically generated, single (X)HTML page is automatically split into multiple pages.<br><br>A new page is created each time a heading (`h1`, `h2`, `h3`, `h4`, `h5`, `h6`) having a “level” less than or equal to specified *`split-before-level`* parameter is found in the source. A “level” is an integer between 0 (corresponds to `h1`) and 5 (corresponds to `h6`). The default value of parameter *`split-before-level`* is 0, which means: for each `h1`, create a new page starting with this `h1`.<br><br>The file specified in the "**Output XML file**" field will contain the generated frameset. Output file extension must be "`.html`".<br><br>While an obsolete HTML feature, a frameset makes it easy browsing these HTML pages. Moreover the table of contents used as the left frame is a convenient way to programmatically list all the generated HTML pages. |
| Multi-page “semantic” (X)HTML 1.0 Transitional | Similar to "Single-page “semantic” (X)HTML 1.0 Transitional [6]" except that the intermediate, automatically generated, single (X)HTML page is automatically split into multiple pages.<br><br>A new page is created each time a heading (`h1`, `h2`, `h3`, `h4`, `h5`, `h6`) having a “level” less than or equal to specified *`split-before-level`* parameter is found in the source. A “level” is an integer between 0 (corresponds to `h1`) and 5 (corresponds to `h6`). The default value of parameter *`split-before-level`* is 0, which means: for each `h1`, create a new page starting with this `h1`.<br><br>The file specified in the "**Output XML file**" field will contain the generated frameset. Output file extension must be "`.html`".<br><br>While an obsolete HTML feature, a frameset makes it easy browsing these HTML pages. Moreover the table of contents used as the left frame is a convenient way to programmatically list all the generated HTML pages. |
| Multi-page “semantic” (X)HTML 1.1 | Similar to "Single-page “semantic” (X)HTML 1.1 [6]" except that the intermediate, automatically generated, single (X)HTML page is automatically split into multiple pages.<br><br>A new page is created each time a heading (`h1`, `h2`, `h3`, `h4`, `h5`, `h6`) having a “level” less than or equal to specified *`split-before-level`* parameter is found in the source. A “level” is an integer between 0 (corresponds to `h1`) and 5 (corresponds to `h6`). The default value of parameter *`split-before-level`* is 0, which means: for each `h1`, create a new page starting with this `h1`. |

| Name | Description |
|---|---|
| | The file specified in the "**Output XML file**" field will contain the generated frameset. Output file extension must be "`.html`".<br><br>While an obsolete HTML feature, a frameset makes it easy browsing these HTML pages. Moreover the table of contents used as the left frame is a convenient way to programmatically list all the generated HTML pages. |
| Multi-page “semantic” (X)HTML 5.0 | Similar to "Single-page “semantic” (X)HTML 5.0 [7]" except that the intermediate, automatically generated, single (X)HTML page is automatically split into multiple pages.<br><br>A new page is created each time a heading (`h1`, `h2`, `h3`, `h4`, `h5`, `h6`) having a “level” less than or equal to specified *`split-before-level`* parameter is found in the source. A “level” is an integer between 0 (corresponds to `h1`) and 5 (corresponds to `h6`). The default value of parameter *`split-before-level`* is 0, which means: for each `h1`, create a new page starting with this `h1`.<br><br>The file specified in the "**Output XML file**" field will contain the generated frameset. Output file extension must be "`.html`".<br><br>While an obsolete HTML feature, a frameset makes it easy browsing these HTML pages. Moreover the table of contents used as the left frame is a convenient way to programmatically list all the generated HTML pages. |
| Styled (X)HTML | Single-page styled XHTML 1.0 Transitional document, embedding a CSS stylesheet, *looking very much like the input DOCX file*.<br><br>This document has a "`text/html;charset=UTF-8`" `Content-Type` `meta` in order to be treated as HTML (not XHTML) by Web browsers.<br><br>Output file extension must be "`.html`". |
| Single page “semantic” (X)HTML 5.0 | Semantic (non-styled) XHTML 5.0 document.<br><br>This XHTML 5.0 document may contain nested `section` elements.<br><br>This document has a "`text/html;charset=UTF-8`" `Content-Type` `meta` in order to be treated as HTML (not XHTML) by Web browsers.<br><br>Output file extension must be "`.html`". |
| Single page “semantic” XHTML 1.0 Strict | Semantic (non-styled) XHTML 1.0 Strict document.<br><br>Output file extension must be "`.xhtml`". |
| Single page “semantic” XHTML 1.0 Transitional | Semantic (non-styled) XHTML 1.0 Transitional document. |

| Name | Description |
|---|---|
| | Output file extension must be ".xhtml". |
| Single page “semantic” XHTML 1.1 | Semantic (non-styled) XHTML 1.1 document.<br><br>Output file extension must be ".xhtml". |
| Web Help containing styled (X)HTML | Similar to "Single-page styled (X)HTML [6]" except that the generated (X)HTML pages are compiled into a Web Help. The Web Help compiler used to do this is free, open source, XMLmind Web Help Compiler.<br><br>The file specified in the "**Output XML file**" field will contain the entry point of the Web Help. Output file extension must be ".html". |
| Web Help containing “semantic” (X)HTML 1.0 Strict | Similar to "Multi-page “semantic” (X)HTML 1.0 Strict [5]" except that the generated (X)HTML pages are compiled into a Web Help. The Web Help compiler used to do this is free, open source, XMLmind Web Help Compiler.<br><br>The file specified in the "**Output XML file**" field will contain the entry point of the Web Help. Output file extension must be ".html". |
| Web Help containing “semantic” (X)HTML 1.0 Transitional | Similar to "Multi-page “semantic” (X)HTML 1.0 Transitional [5]" except that the generated (X)HTML pages are compiled into a Web Help. The Web Help compiler used to do this is free, open source, XMLmind Web Help Compiler.<br><br>The file specified in the "**Output XML file**" field will contain the entry point of the Web Help. Output file extension must be ".html". |
| Web Help containing “semantic” (X)HTML 1.1 | Similar to "Multi-page “semantic” (X)HTML 1.1 [5]" except that the generated (X)HTML pages are compiled into a Web Help. The Web Help compiler used to do this is free, open source, XMLmind Web Help Compiler.<br><br>The file specified in the "**Output XML file**" field will contain the entry point of the Web Help. Output file extension must be ".html". |
| Web Help containing “semantic” (X)HTML 5.0 | Similar to "Multi-page “semantic” (X)HTML 5.0 [6]" except that the generated (X)HTML pages are compiled into a Web Help. The Web Help compiler used to do this is free, open source, XMLmind Web Help Compiler.<br><br>The file specified in the "**Output XML file**" field will contain the entry point of the Web Help. Output file extension must be ".html". |

The combobox may contain additional entries corresponding to custom conversions specified by the means of plugins (if any).

- **OR** Click "**Use text file containing w2x options**" to select custom conversions then click to choose the "`.txt`" file containing the custom conversion specification.

  A custom conversion specification is simply a text file containing **w2x** command-line options. More information in Section 3, “Custom conversion specifications” [8].

3. In the "**Output XML file**" frame, click to select the XML file which is the result of the conversion.

   The "Single-page styled (X)HTML [6]" and "Web Help [7]" conversions generate several files in the directory containing the file specified in the "**Output XML file**" field. Therefore it is recommended to specify a *new output directory* each time you use these conversions. Note that this output directory is created on the fly if needed too. If on the contrary, such output directory already exists, it is not automatically made empty. However, in such case, a dialog box is displayed to ask you if you want to make the output directory empty before proceeding with the conversion.

4. Optionally check "**Open output file in associated application**"[2] if you want to preview the result of the conversion in an external application.

   The file extension of the output file must have been associated to an external application. For example, if the name of the output file ends with "`.html`", this external application will be the system's Web browser.

5. Click **Convert**.

   During the file conversion, the **Convert** button becomes a **Cancel** button. Click **Cancel** to cancel the current file conversion. Click **Cancel** while pressing the Shift key if you want to forcibly cancel the current file conversion. Use Shift+click only when the file conversion seems to be blocked for a long time.

## 3. Custom conversion specifications

A custom conversion specification is simply a text file, UTF-8 encoded, containing a number of **w2x** command-line options separated by space or new line characters.

*Example 1. Simple example used to generate a DocBook v4.5 `book` possibly containing CALS (not HTML) tables*

```
-o docbook
-p convert.set-column-number yes
-p transform.cals-tables yes
```

*Example 2. Advanced example used to convert XMLmind Word To XML Manual to a DITA bookmap*

Custom conversion specification file *`w2x_install_dir`*`/doc/manual/bookmap_options.txt`:

```
-p edit.prune.preserve p-ProgramListing
-p edit.inlines.convert " c-Code code ! c-Abbrev abbr "
-p edit.blocks.convert "p-Term dt g:id='dl' g:container='dl' !¬
 p-Definition dd g:id='dl' g:container='dl' !¬
```

[2] This checkbox may be disabled on some platforms (e.g. Linux).

```
 p-ProgramListing span g:id='pre' g:container='pre'"
-pu edit.after.blocks customize/notes.xed
-p transform.root-topic-id manual
-p transform2.topic-path manual_bookmap_topics
-p transform2.section-depth 6
-o bookmap
-t customize/custom_topic.xslt
```

where `customize/notes.xed` is:

```
namespace "http://www.w3.org/1999/xhtml";
namespace html = "http://www.w3.org/1999/xhtml";
namespace g = "urn:x-mlmind:namespace:group";

for-each /html/body//p[get-class("p-Note")] {
    delete-text("note:\s*", "i");
    if content-type() <= 1 and not(@id) {
        delete();
    } else {
        remove-class("p-Note");
        set-attribute("g:id", "note_group_member");
        set-attribute("g:container", "div class='role-note'");
    }
}

group();
```

and `customize/custom_topic.xslt` is:

```
<xsl:stylesheet version="1.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  xmlns:h="http://www.w3.org/1999/xhtml"
  exclude-result-prefixes="h">

<xsl:import href="w2x:xslt/topic.xslt"/>

<xsl:template match="h:div[@class = 'role-note']">
  <note>
    <xsl:call-template name="processCommonAttributes"/>
    <xsl:apply-templates/>
  </note>
</xsl:template>

<xsl:template match="h:code">
  <tt>
    <xsl:call-template name="processCommonAttributes"/>
    <xsl:apply-templates/>
  </tt>
</xsl:template>

</xsl:stylesheet>
```

Note that relative URIs referenced by `-pu` options are relative to the text file containing the custom conversion specification. For example, `customize/notes.xed` referenced in *`w2x_install_dir`*`/doc/manual/bookmap_options.txt` is in fact *`w2x_install_dir`*`/doc/manual/customize/notes.xed`.

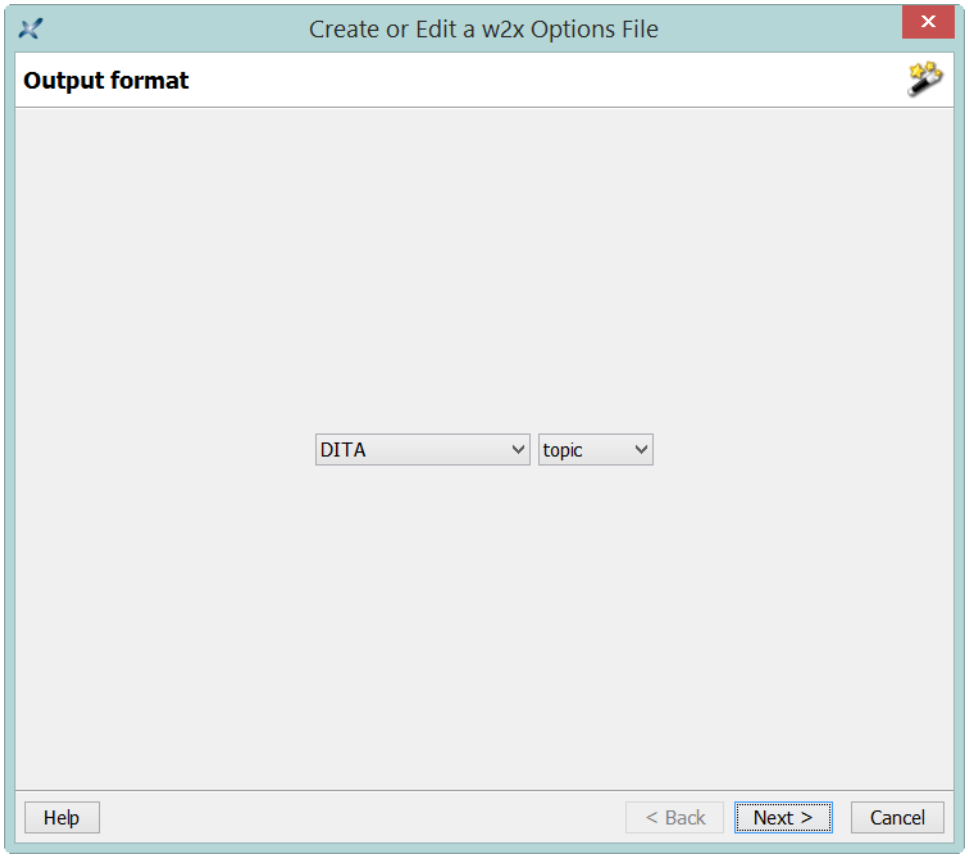
## 3.1. Using the setup assistant to create custom conversion specifications

This setup assistant (**AKA** “wizard” style dialog box) is displayed when you click  . It makes it quick and easy creating **w2x** option files.

An important feature of this setup assistant is that it has a screen which may be used to map MS-Word character and paragraph styles (e.g. `p-KeyboardInput`) to XML elements possibly having attributes (e.g. DITA `pre outputclass="keyboard-input"`).

This setup assistant may also be used to *modify* the **w2x** options files it has created. It cannot modify **w2x** options file created by hand. Other limitation: if you modify by hand a **w2x** options file created using the setup assistant, all your changes will be lost the next time you'll use the setup assistant to modify this **w2x** options file.

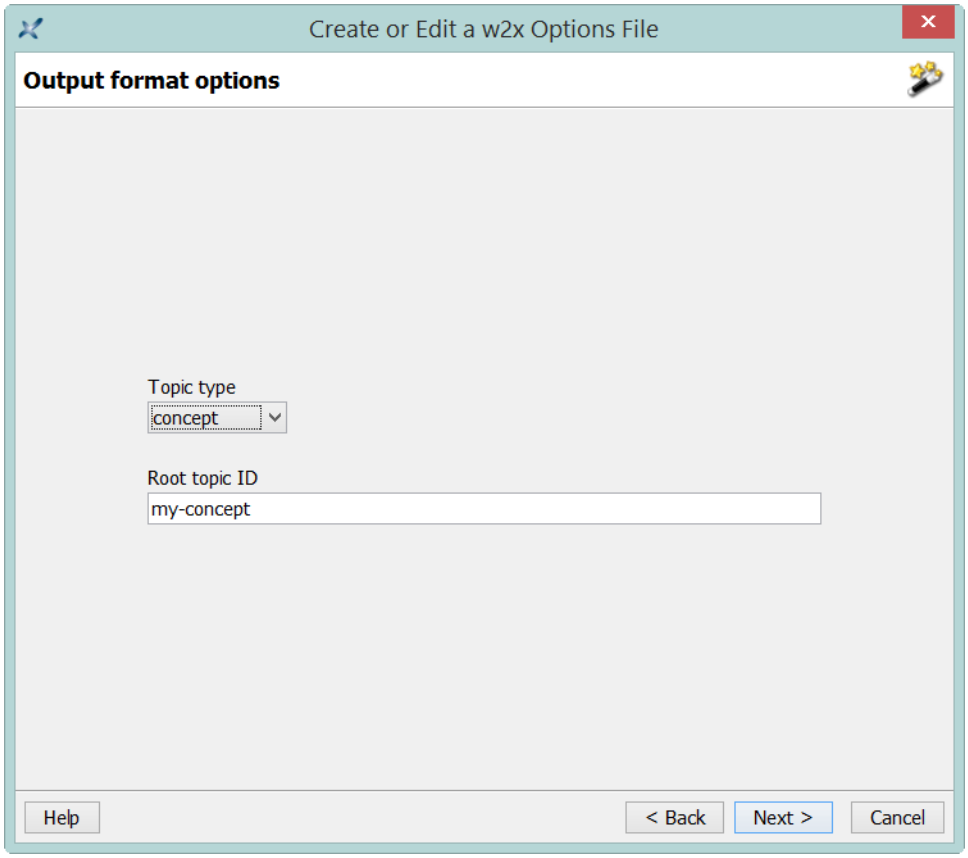
### 3.1.1. "Output format" screen




This screen is used to specify the output format.

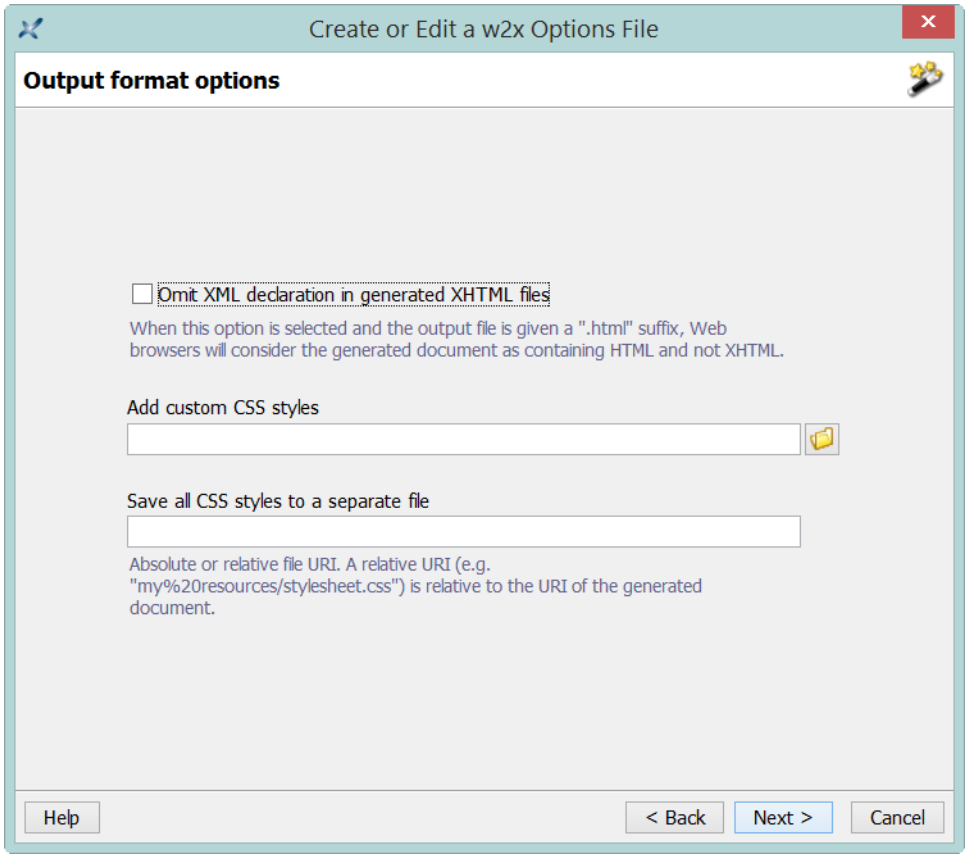
The **Other** category may be used to select custom conversions specified by the means of plugins (if any).

### 3.1.2. "Output format options" screen



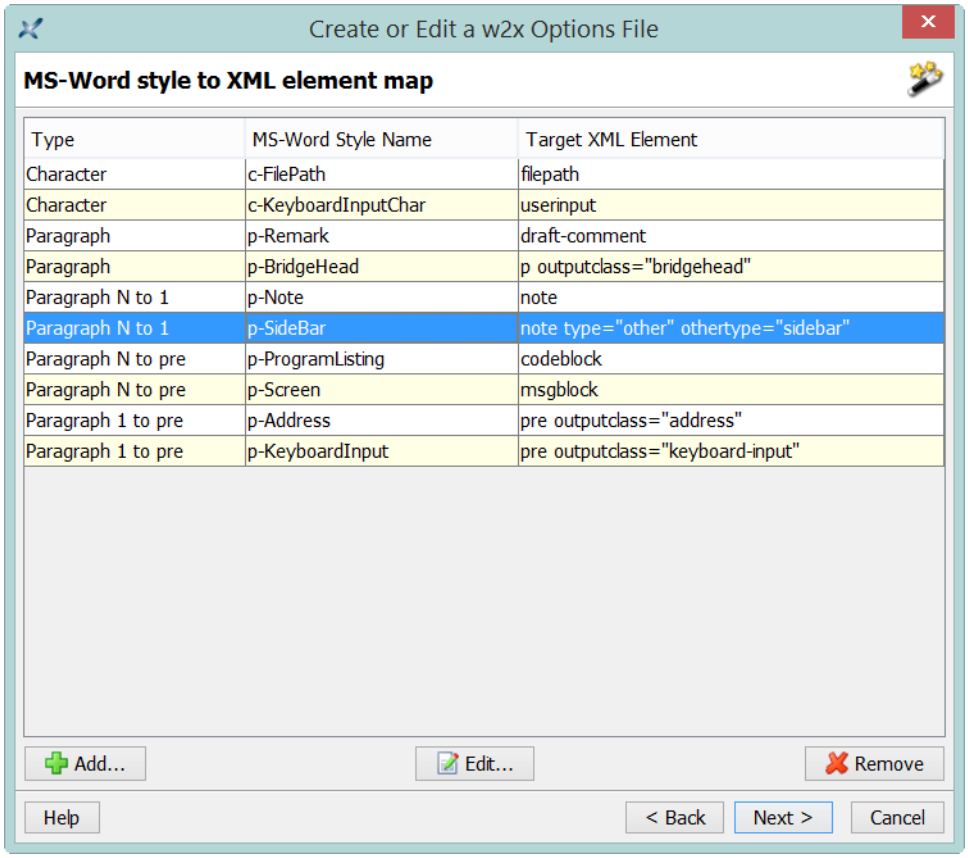

This is screen is used to specify output format options.

The content of this screen depends on the output format selected in the previous screen. For example, the options screen below is displayed after selecting **Single-page (X)HTML** → **Styled XHTML**.




This screen is *not* displayed when the chosen output format does not have any commonly used option. This is the case for custom conversions specified by the means of plugins (if any).

### 3.1.3. "MS-Word style to XML element map" screen

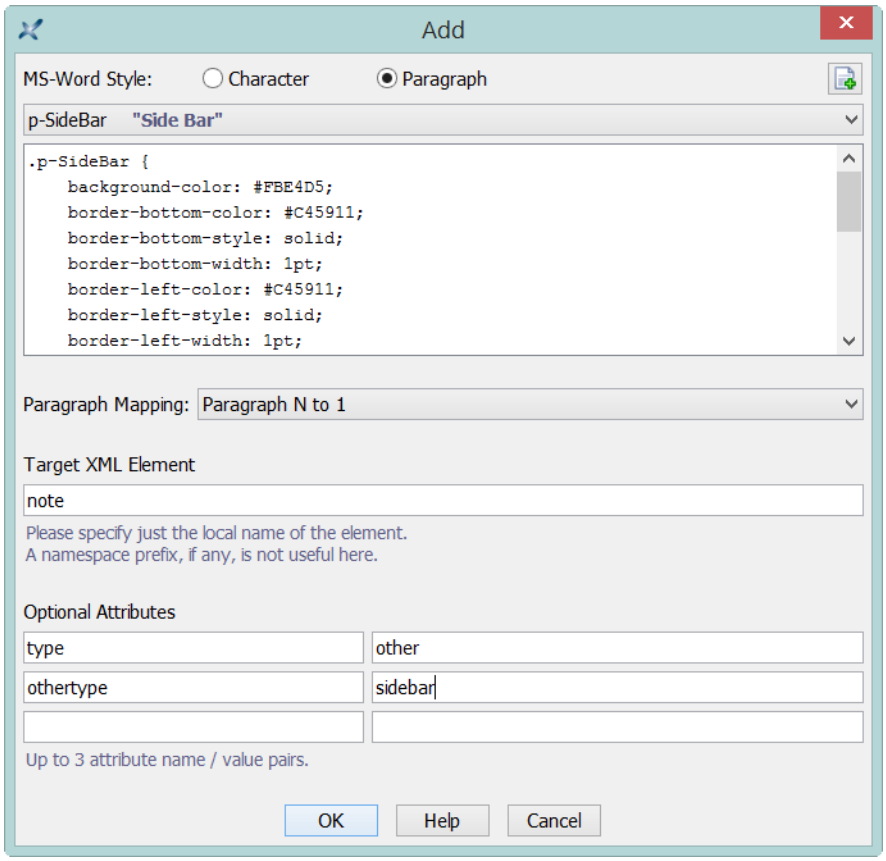



This screen is used to map MS-Word character and paragraph styles (e.g. `p-SideBar`) to XML elements possibly having attributes (e.g. DITA `note type="other" othertype="sidebar"`).

This screen is *not* displayed when the chosen output format is *styled (X)HTML of any kind* (single-page, multi-page, EPUB, Web Help).

Click **Add** to add an entry to the map. Click **Edit** to modify selected entry. Click **Remove** to remove selected entry.

### 3.1.3.1. Dialog box allowing to add or modify an entry of the MS-Word style to XML element map [13]




If you want paragraphs or text runs found in the input DOCX file and having a given *custom* MS-Word style to be converted to a specific XML element, please proceed as follows:

1. Select an MS-Word style by:

   a. Clicking **Character** to list custom character (text run) styles and **Paragraph** to list custom paragraph styles.

   b. Selecting a style from the custom styles combobox.

   It's possible to add to the combobox custom styles coming from other DOCX files by clicking  .

#### About the "p-" and "c-" style name prefixes automatically added by w2x

The name of a paragraph style is automatically given a "`p-`" or "`tp-`" prefix and the name of a character style is automatically given a "`c-`" prefix.

For example, custom combined paragraph and character MS-Word style called "`CodeSnippet`" is represented in w2x by paragraph style "`p-CodeSnippet`" and by character style "`c-CodeSnippet`".

2. There is just one way to map a character style to a specific XML element. For example, a text run found in the input DOCX file having a custom style `c-FilePath` is converted to an DITA element `filepath`.

   However, there are *4 different ways* to map a paragraph style to a specific XML element. Therefore if you have selected a paragraph style, you must now choose a mapping type using the "**Paragraph Mapping**" combobox.

   **Paragraph**

   A paragraph found in the input DOCX file and having a custom style, e.g. `p-Remark`, is converted to an XML element, e.g. DITA `draft-comment`.

   **Paragraph N to1**

   Consecutive paragraphs all having the same custom style, e.g. `p-Sidebar`, are *grouped* into the same parent XML element, e.g. DITA `note type="other" othertype="sidebar"`.

   **Paragraph N to pre**

   Consecutive paragraphs all having the same custom style and containing significant whitespace but no forced line breaks (that is, each paragraph typically containing a single line of code), e.g. `p-programListing`, are converted to text runs and then *grouped* into the same parent XML element where whitespace and line breaks are preserved, e.g. DITA `codeblock`.

   **Paragraph 1 to pre**

   A paragraph having a custom style and containing significant whitespace and forced line breaks (that is, a paragraph typically containing multiple lines of code), e.g. `p-KeyboardInput`, is converted to an XML element where whitespace and line breaks are preserved, e.g. DITA `pre outputclass="keyboard-input"`.

   The mapping type of a paragraph style — "**Paragraph**", "**Paragraph N to1**", "**Paragraph N to pre**" or "**Paragraph 1 to pre**" — is automatically suggested by this dialog box after you select this paragraph style. However the heuristics used to implement the suggestions being rather conservative, the suggestion will be generally "**Paragraph**".

3. Type the name of the target element in the "**Target XML Element**" field.

   This element name must have no namespace prefix. This is possible because the setup assistant assumes that if needed to (e.g. output format is XHTML or DocBook 5.0), a default namespace has been specified in the XSLT stylesheets used to implement the conversion.

   Carefully choose the target XML element otherwise you may end up with an invalid output file.

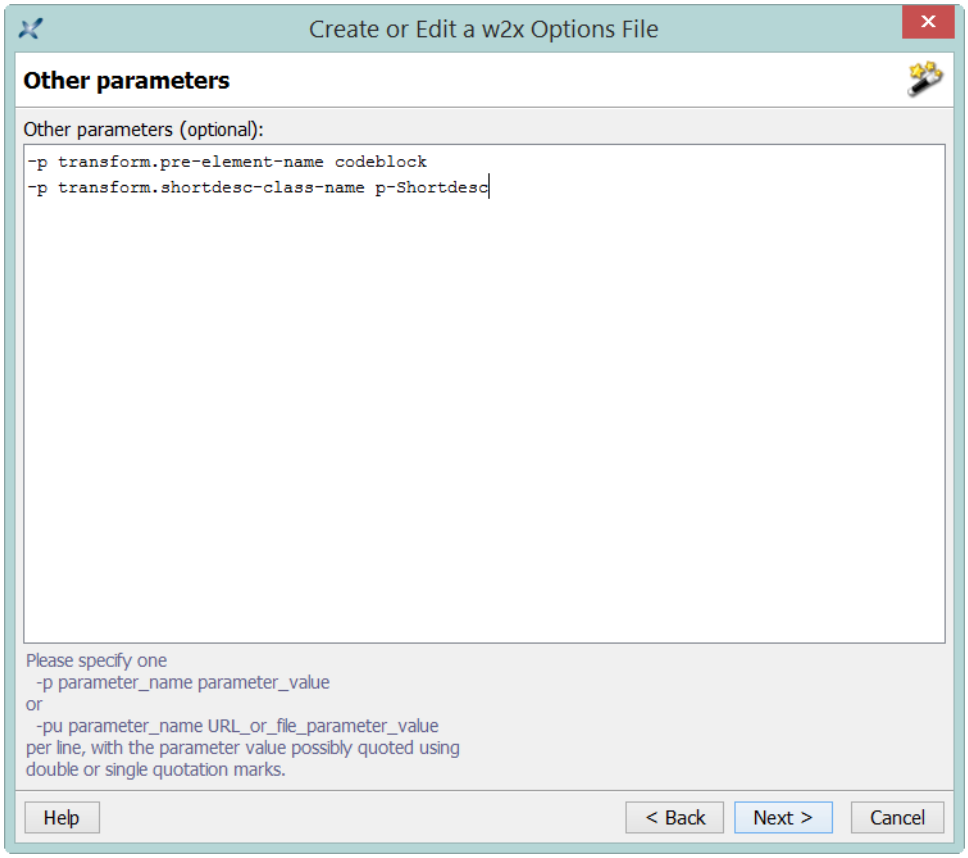
   For example, when the type of the mapping is "**Paragraph N to1**", do not choose DITA `cite` as a `cite` element may not have `p` child elements (DOCX paragraphs are converted to DITA `p`). Instead choose DITA elements like `note` or `section`.

4. Optionally type up to 3 attribute name/value in the "**Optional Attributes**" fields.

The name of an attribute must have no namespace prefix. More precisely, only the standard, predefined, "`xml:`" prefix is supported here.

5. Click **OK**.

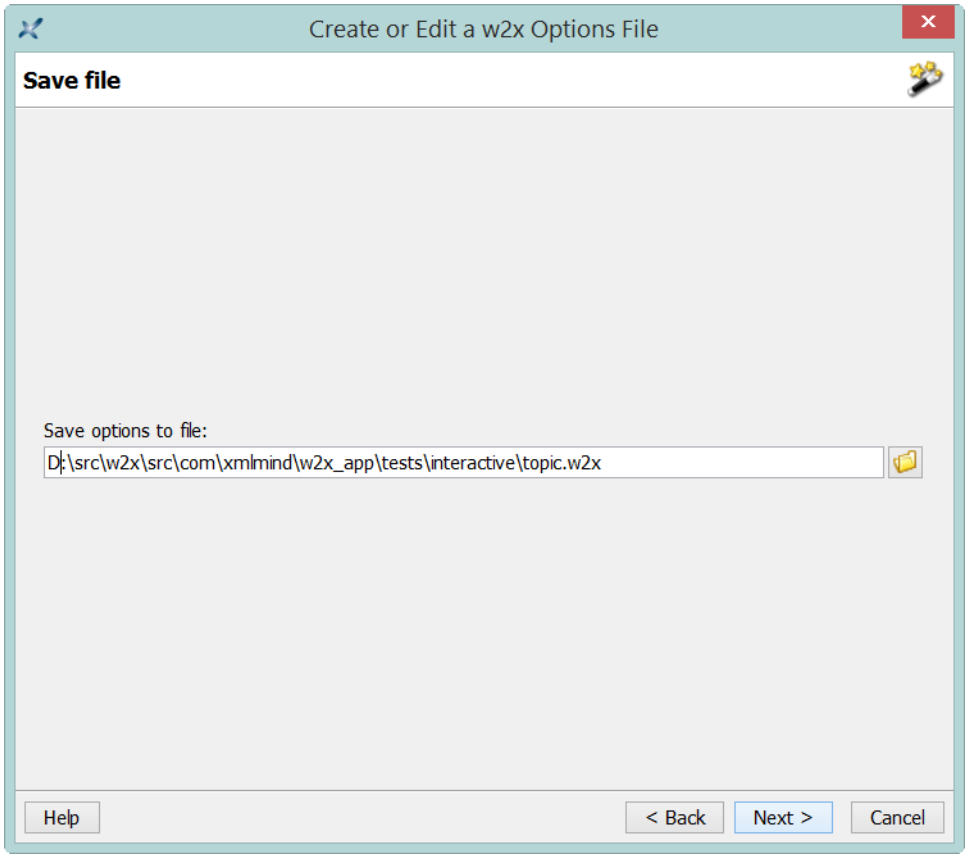
### 3.1.4. "Other parameters" screen




This screen lets the user specify parameters which are not supported by the "**Output format options**" and "**MS-Word style to XML element map**" screens. For example, when generating a DITA document, the other screens do not let the user specify `-p transform.pre-element-name codeblock` (default value being `pre`).

For clarity, one of `-p` *`parameter_name`* `parameter_value` or `-pu` *`parameter_name URL_or_file_parameter_value`* must be specified per line, with the parameter value possibly quoted using double or single quotation marks.

### 3.1.5. "Save file" screen




This screen is used to specify a save file for the **w2x** options file created or edited using the setup assistant. The recommended extensions for **w2x** options files are "`.txt`"[3] or "`.w2x`".

#### About the files created by the setup assistant

Note that when needed to, the setup assistant also creates a custom XSLT stylesheet which is referenced by the **w2x** options file.

For example, in the case depicted in the above screenshot, the setup assistant creates both `topic.w2x` and `topic_scripts/custom_transform.xslt`. Excerpts from `topic.w2x`:

```
# AUTOMATICALLY CREATED BY w2x-app SETUP ASSISTANT. PLEASE DO NOT EDIT BY HAND!
###outputFormat topic
###option transform.topic-type concept
...
-o topic
-p transform.topic-type concept
...
-t topic_scripts/custom_transform.xslt
```

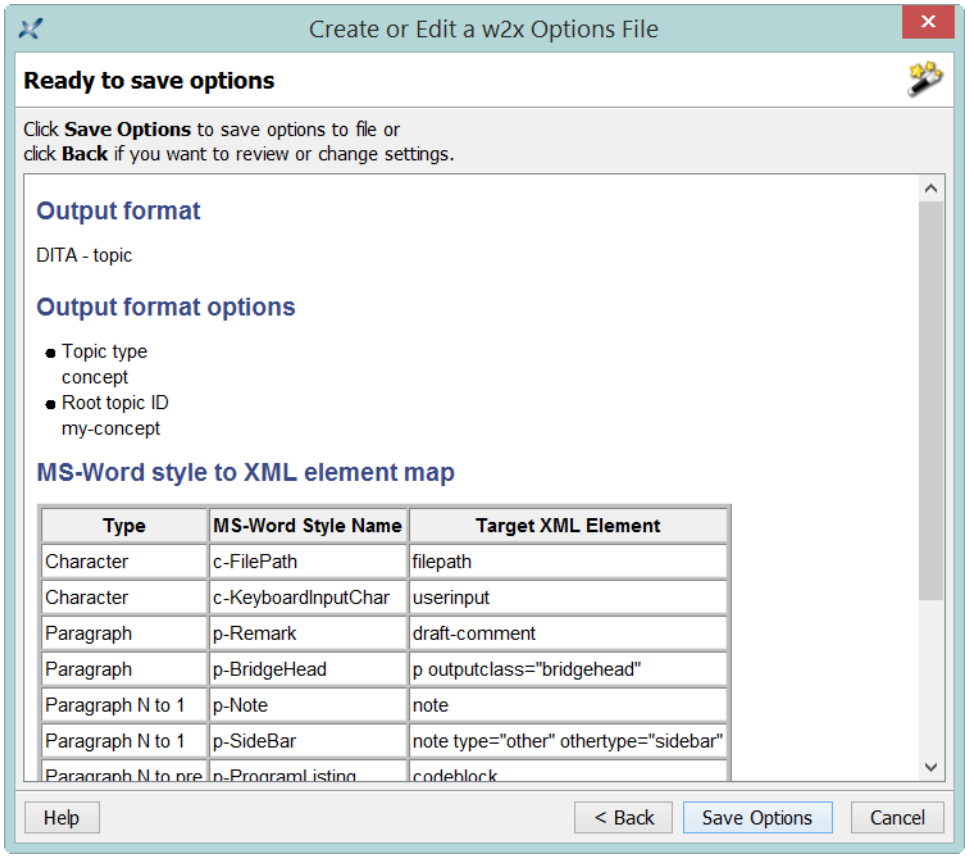
Excerpts from `topic_scripts/custom_transform.xslt`:

```
<xsl:stylesheet version="1.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
```

[3] Remember that **w2x** options filesare plain text, UTF-8 encoded, files.

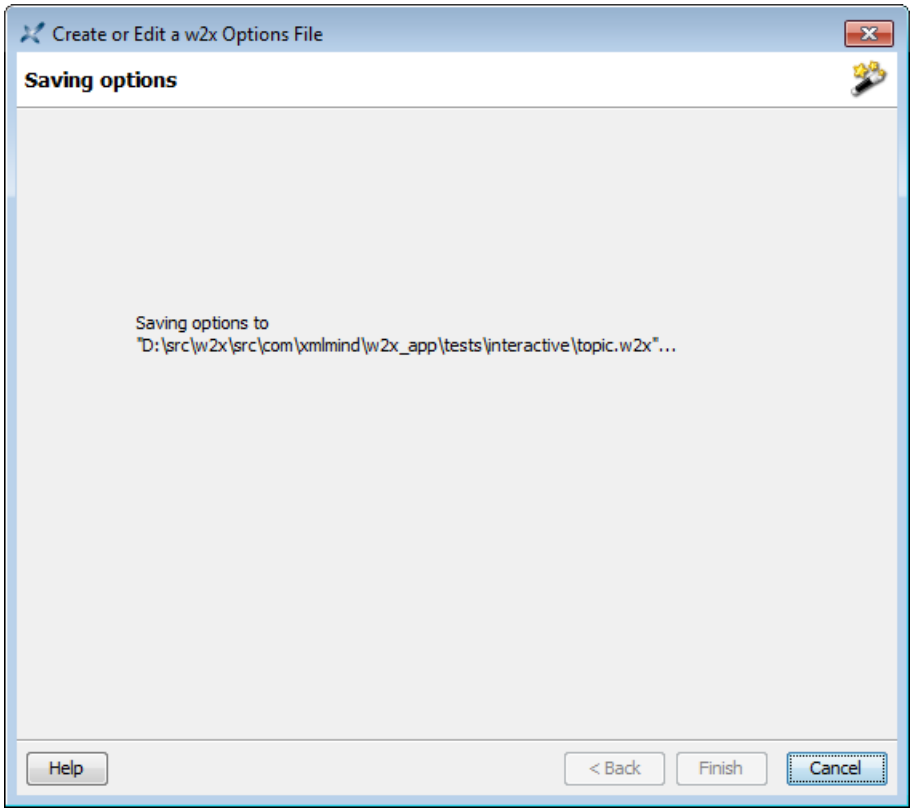
```
  xmlns:h="http://www.w3.org/1999/xhtml"
  exclude-result-prefixes="h">

<xsl:import href="w2x:xslt/topic.xslt"/>

<xsl:template match="h:span[@class='c-FilePath']">
  <filepath>
    <xsl:call-template name="processCommonAttributes"/>
    <xsl:apply-templates/>
  </filepath>
</xsl:template>
...
```

### 3.1.6. "Ready to save options" screen



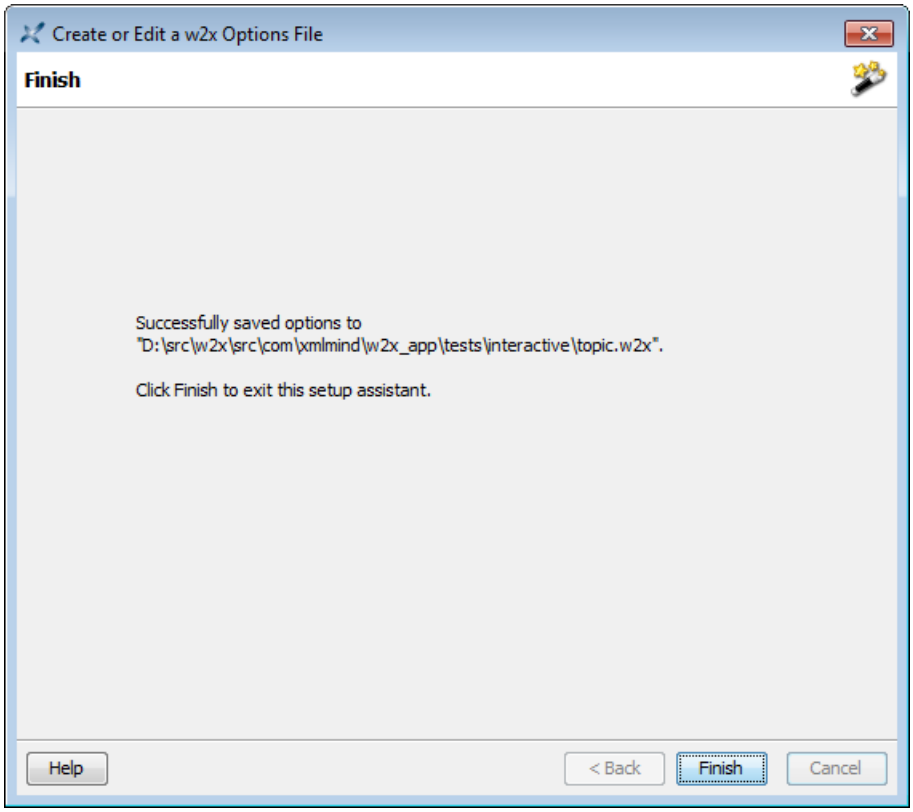

This screen is displayed to let the user review all the options chosen during the previous steps.

### 3.1.7. "Saving options" screen




The screen is very briefly displayed by the setup assistant while it is saving the **w2x** options file.

### 3.1.8. "Finish" screen




This screen is displayed after the **w2x** options file has been saved by the setup assistant.

## 3.2. Creating custom conversion specification without the help of the setup assistant

### 3.2.1. Creating a custom conversion specification

If you need to create a custom conversion specification, use a predefined conversion as a starting point.

Click  to copy to the clipboard the **w2x** command-line options which are equivalent to the currently selected predefined conversion. Then paste the contents of the clipboard into the new text file which will be your custom conversion specification.

For example, clicking  while selected predefined conversion is "**DocBook V5.0 article**" combobox copies:

```
-o docbook5
-p convert.set-column-number yes
-p transform.cals-tables yes
-p transform.hierarchy-name article
```

to the clipboard.

### 3.2.2. Modifying an existing custom conversion specification

If you need to modify an existing custom conversion specification, click  [4] to open in the system's text editor the file specified in the text field below "**Use text file containing w2x options**".

## A. Command line options

```
w2x-app [ advanced_option ]* [ input_docx_file ]?
```

Advanced options:

`-vv`
: Show debug messages.

`-putpref` *key value*
: Adds or replace preference specified by *key*/*value* to the set of the user's preferences. See Appendix B, *User preferences* [21].

`-delpref` key
: Removes preference specified by *key* from the set of the user's preferences.

## B. User preferences

### 1. How to change a user preference

A user preference is set or changed once for all by using the `-putpref` command-line option. Its syntax is: `-putpref` *preference_key preference_value*. Example:

```
w2x-app -putpref displayScaling 200
```

A user preference is removed, and thus the default behavior is restored, by using the `-delpref` command-line option. Its syntax is: `-delpref` *preference_key*. Example:

```
w2x-app -delpref displayScaling
```

### 2. Where user preferences are stored

The preferences specified in this dialog box are stored in *w2x_user_preferences_dir*`/w2x-app.properties`, a Java™ property file (an ISO-8859-1 text file using a very simply *key* :*value* format).

W2x user preferences directory is:

- *$HOME*`/.w2x/` on Linux.
- *$HOME*`/Library/Application Support/XMLmind/WordToXML/` on the Mac.
- *%APPDATA%*`\XMLmind\WordToXML\` on Windows. Example: `C:\Users\john\AppData\Roaming\XMLmind\WordToXML\`.

[4] This button may be disabled on some platforms (e.g. Linux).

If you cannot see the "`AppData`" directory using Microsoft Windows File Manager, turn on **Tools**>**Folder Options**>**View**>**File** and **Folders**>**Show** hidden files and folders.

## 3. Supported user preferences

displayScaling

May be used to globally change the size of all the items comprising the user interface of w2x-app. More information [2]. Value: percentage between `100` and `400` or `-1` which means: use system settings. Default: `-1`.

lookAndFeelClassName

May be used to change the Look&Feel of w2x-app. More information [2].Value: Java™ class name of a Look And Feel or "`-`" (an alias for "`default`") or "`default`" or "`fallback`" (means: system **LAF**). Default: "`default`".

useNativeFileChooser

Use the native file chooser in preference to the multi-platform file chooser. Value: `true` or `false`. Default: `false` (because file extension filters are not supported when the native file chooser is invoked by Java™).

This user preference is ignored on platforms other than Windows and the Mac.