XMLmind XML Editor - Support of DocBook 5.1+ Assemblies

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

July 8, 2025

Abstract

This document describes the commands which are specific to DocBook 5.1+ assemblies.

Table of Contents

1. Why read this document?

This document describes the support of DocBook 5.1+ *assemblies* in XMLmind XML Editor. It does not deal with DocBook 5.1+ elements which are not related to the `assembly` element, that is, `book`, `chapter`, `section`, `topic`, etc. How to create and edit these elements in XMLmind XML Editor is documented in another document: *XMLmind XML Editor - DocBook Support*. The content of this other document applies to DocBook version 4+ and 5+.

About XMLmind Assembly Processor

When you'll use **Assembly** → **Check Assembly** [3] or **Assembly** → **Convert Document**, you'll implicitly invoke XMLmind Assembly Processor.

XMLmind Assembly Processor is a free, open source, Java™ software component which processes a DocBook `assembly` and all the referenced topics in order to create the equivalent “flat”, monolithic, document (e.g. a DocBook book).

The limitations and implementation specificities of this assembly processor are documented in "*XMLmind Assembly Processor Manual, Limitations and implementation specificities*".

2. The Assembly menu

About DocBook 5.1+ assemblies and the "Easy Profiling" add-on

The "**Easy Profiling**" add-on, included by default in all XMLmind XML Editor distributions, automatically adds a "**Conditional Processing**" sub-menu to the **Assembly** menu.

Menu item "**Select Profile**" allows to associate a profile set (`.profiles` file) to an `assembly`.

However, conditional processing is never applied to the `assembly` itself. Conditional processing is applied to the realized document if the DocBook *profiling* XSLT stylesheets have been selected using "**Options** → **Customize Configuration** → **Customize Document Conversion Stylesheets** in *XMLmind XML Editor - Online Help*".

In consequence, in the case of `assembly`, unlike what happens for `book`, `chapter`, etc, profiling attributes and/or the selected/unselected profiles are never rendered on screen.

So why bother associating a profile set to an assembly?

1. The main reason to do this is that once a profile set has been associated to an `assembly` which has been declared as being *a master document* in *XMLmind XML Editor - Online Help*, all the resources (`topic`, `chapter`, `section`, etc) referenced by this `assembly` will automatically "inherit" the chosen conditional processing profile. More information in Section 3.1, "Associating the "`.profiles`" file to the map rather than to individual topics" in *XMLmind XML Editor - Easy Profiling*.

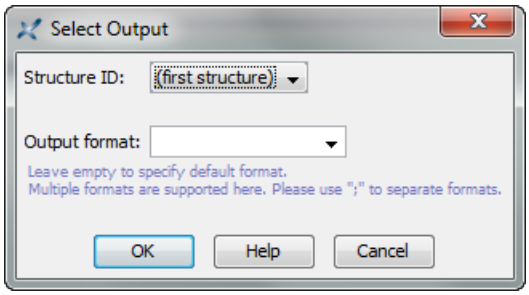
2. It may be convenient to use menu item "**Set Profiling Attributes**" to add profiling attributes to the `filterin` and `filterout` elements.

Select Output

An assembly may contain several `structure` elements. A `structure` element may specify several output formats (`web`, `print`, etc)[1] .

When this is the case, you'll want to specify which structure you want to convert by the means of the "**Convert Document**" menu items and also which output format you target. Menu item "**Select Output**" displays a dialog box allowing this specification.

*Figure 1. The dialog box displayed by menu item "**Select Output**"*



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

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**" combobox. 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 `output-format` attribute.

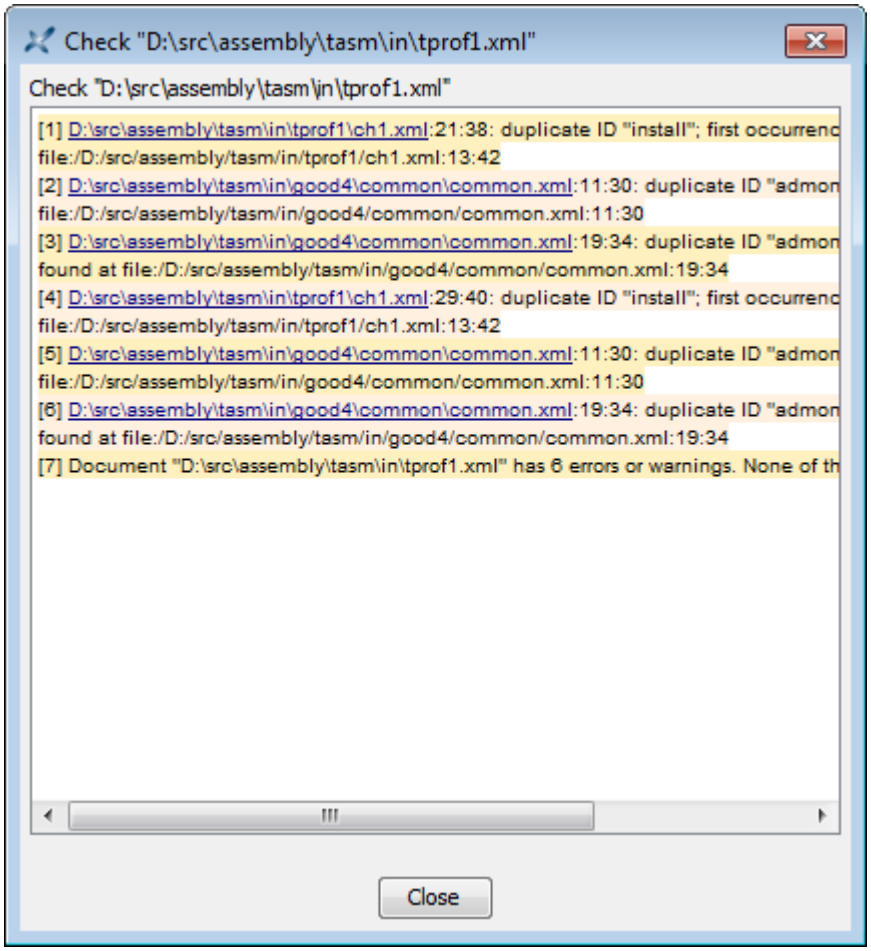
Multiple output formats separated by "`;`" may be specified. For example, "`web;print`" means output format is "`web`" *OR* "`print`".

Check Assembly

Check the assembly being edited and all its referenced topics for all kinds of errors, including cross-reference errors and missing image resources.

This check task which can be lengthy is run in background. While this task is running, a non-modal dialog box displays all the errors and warnings found in the assembly being edited and all the topics referenced by this assembly. If no errors or warnings are found, the dialog box is automatically closed. Otherwise it stays opened allowing you to review each error or warning. After you are done, you'll have to close the dialog box by clicking **Close** if you want to be able to re-run **Check Assembly**.

*Figure 2. The dialog box displayed by command **Check Assembly***



An assembly being an advanced specification, what part of this specification is checked by command Check Assembly?

- The **Structure ID** and **Output format** specified using menu item **Select Output** [2] are automatically taken into account by command **Check Assembly**.

- If your assembly requires conditional processing (that is, *profiling*), then command **Check Assembly** may report false errors. These false errors are caused by the fact that the conditional processing step has not been applied to the realized document prior to the check step.

  Example: two of the chapters referenced by assembly `book.xml` have `xml:id="install"`. First chapter has also `os="windows"`. Second chapter has also `os="mac"`.

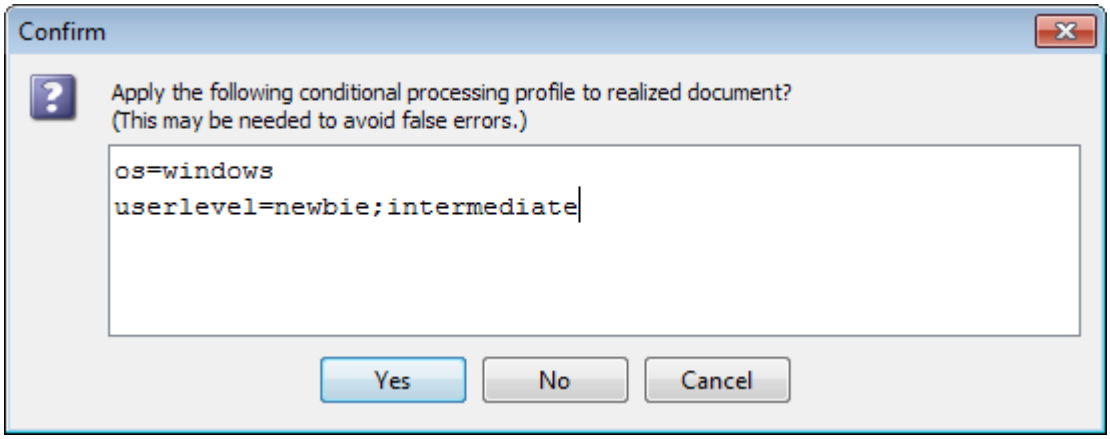
  If you don't apply any conditional processing to the realized `book.xml`, you'll get a duplicate ID warning caused by `xml:id="install"`.

  On the other hand, if you apply, say, profile `os="window"` to the realized `book.xml`, you'll not have this duplicate ID error. Why that? Because by applying profile `os="windows"`, second chapter (having `os="mac"`) is excluded from the realized document prior to checking it.

That's why the **Check Assembly** command automatically takes into account last used profile whether this profile has been specified using **Conditional Processing** → **Select Profile**[2] or this profile has been specified as one or more XSLT stylesheet parameters (e.g. `profile.os=windows`) using the "**Document conversion parameters**" panel in *XMLmind XML Editor - Online Help*.

When such last used profile is detected by **Check Assembly**, this command displays a dialog box letting the user review this profile before performing the check.

*Figure 3. The "apply last used profile" dialog box displayed by command **Check Assembly***



2.1. The "Convert Document" sub-menu

How does it work?

All the "**Convert Document**" menu items first invoke XMLmind Assembly Processor to generate the *realized document* out of the assembly being edited and then, this realized document is converted to other formats (HTML, PDF, DOCX, etc) normally, using the DocBook XSL stylesheets.

What we call the realized document is the “flat”, monolithic, document (e.g. a DocBook 5.1+ `book`) specified by the means of the assembly.

This first processing step explains why the assembly being edited must have been saved to disk prior to be able to use any of the "**Convert Document**" menu items.

Menu items

The "**Convert to RTF**", **WML**, **DOCX**, **ODT**, entries documented below are absent in XMLmind DocBook Editor. They are found only in XMLmind XML Editor.

Using the profiling stylesheets

Conditional processing, also called *profiling* or conditional text, means that you can create a single XML document with some elements marked as conditional. When you process such a document, you can specify which conditions apply for that version of the output, and the

[2]See also About DocBook 5.1+ assemblies and the "**Easy Profiling**" add-on [2].

XSLT stylesheet will include or exclude the marked text to satisfy the conditions. More information in DocBook XSL: The Complete Guide.

If you need to use the profiling XSLT stylesheets rather than the regular ones, use **Options → Customize Configuration → Customize Document Conversion Stylesheets** in *XMLmind XML Editor - Online Help* and select the corresponding stylesheet.

Convert to HTML, Convert to HTML [one page]

Converts the document being edited to multi page or single page HTML.

Generating XHTML rather than HTML

If you prefer to generate XHTML 1.0 or 5 rather than plain HTML, use **Options → Customize Configuration → Customize Document Conversion Stylesheets** and select the corresponding stylesheet.

Convert to Web Help

Converts the document being edited to *Web Help* containing XHTML 5 pages.

The Web Help is generated by XMLmind Web Help Compiler and not by the XSLT stylesheets generating Web.Help which are part of the stock DocBook XSL Stylesheets.

Therefore, you *cannot* use any of XSLT stylesheet parameters having a name starting with "`webhelp.`" and which are documented in "*DocBook XSL Stylesheets: Reference Documentation, HTML Parameter Reference, WebHelp*". For example, using *`webhelp.autolabel`* would have *no effect* on the Web Help generated by this menu entry.

Instead please use any of the parameter documented in "*XMLmind Web Help Compiler Manual, Parameters*" after prefixing their names with "`wh-`". Example: `wh-favicon=https://docbook.org/graphics/banner.png`

Convert to HTML Help

Converts the document being edited to a `.chm` file. This command is disabled on platforms other than Windows.

For this command to work, the HTML Help compiler, `hhc.exe`, must have been declared as the helper application associated to files having a "`hhp`" extension. This can be specified by using the **Preferences** dialog box, **Helper Applications** section.

Convert to Eclipse Help

Converts the document being edited to Eclipse Help.

If you want Eclipse to display your Eclipse Help document in its help viewer, you must

1. specify the following XSLT stylesheet parameters: *`eclipse.plugin.name`*, *`eclipse.plugin.id`*, *`eclipse.plugin.provider`*, prior to selecting **DocBook → Convert Document → Convert to Eclipse Help**;
2. give to the output folder the name specified in *`eclipse.plugin.id`*;
3. copy the output folder containing the generated Eclipse Help document to *`eclipse_install_dir`*`/dropins/` and not *`eclipse_install_dir`*`/plugins/`.

Convert to EPUB

Converts the document being edited to EPUB.

Convert to RTF (Word 2000+)

Converts the document being edited to RTF (Rich Text Format) using XMLmind FO Converter (see http://www.xmlmind.com/foconverter/). The document generated by this command can be edited and printed using Microsoft® Word 2000 and above.

Convert to WordprocessingML (Word 2003+).

Converts the document being edited to WordprocessingML using XMLmind FO Converter. The document generated by this command can be edited and printed using Microsoft® Word 2003 and above.

Convert to Office Open XML (Word 2007+)

Converts the document being edited to Office Open XML (`.docx` file) using XMLmind FO Converter. The document generated by this command can be edited and printed using Microsoft® Word 2007 and above.

Convert to OpenDocument (OpenOffice.org 2+)

Converts the document being edited to OpenDocument (`.odt` file) using XMLmind FO Converter. The document generated by this command can be edited and printed using OpenOffice.org 2.

Convert to PDF

Converts the document being edited to PDF (Adobe® Portable Document Format, also known as Acrobat®) using RenderX XEP (see http://www.renderx.com/), if its plug-in has been installed, and Apache FOP otherwise (see http://xmlgraphics.apache.org/fop/).

All the above **Convert** commands display the URL chooser dialog box rather than the standard file chooser dialog box.

For all **Convert** commands except for the "**Convert to HTML**" command, you must specify the URL (Uniform Resource Locator) of a save file. The "**Convert to HTML**" command creates multiple HTML pages with a first page called `index.html`, therefore you need to specify the URL of a save directory.

Note that these commands can create directories on the fly, if needed to. For example, if you specify `http://www.acme.com/docs/report43/mydoc.html` as the URL of the save file and if directory `report43/` does not exist, this directory will be created during command execution.

Syntax highlighting

You can automatically colorize the source code contained in `programlisting` elements. This feature, commonly called *syntax highlighting*, has been implemented using an open source software component called "XSLT syntax highlighting".

If you want to turn on syntax highlighting in a DocBook document:

1. Add attribute `language` to element `programlisting`. The value of attribute `language` must be any of: `bourne`, `c`, `cmake`, `cpp`, `csharp`, `css21`, `delphi`, `ini`, `java`, `javascript`, `lua`, `m2` (Modula 2), `perl`, `php`, `python`, `ruby`, `sql1999`, `sql2003`, `sql92`, `tcl`, `upc` (Unified Parallel C), `html`, `xml`.
2. Specify XSLT stylesheet parameter highlight.source=1 using **Options** → **Customize Configuration** → **Change Document Conversion Parameters**. Do this for each output format you want to generate.

If you want to customize syntax highlighting for an HTML-based output format (XHTML, EPUB, etc), redefine any of the following CSS styles: `.hl-keyword`, `.hl-string`, `.hl-number`, `.hl-comment`, `.hl-doccomment`, `.hl-directive`, `.hl-annotation`, `.hl-tag`, `.hl-attribute`, `.hl-value`, `.hl-doctype`. Example:

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

This can be done from within **XXE** using **Options** → **Customize Configuration** → **Customize Document Conversion Stylesheets**.

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

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

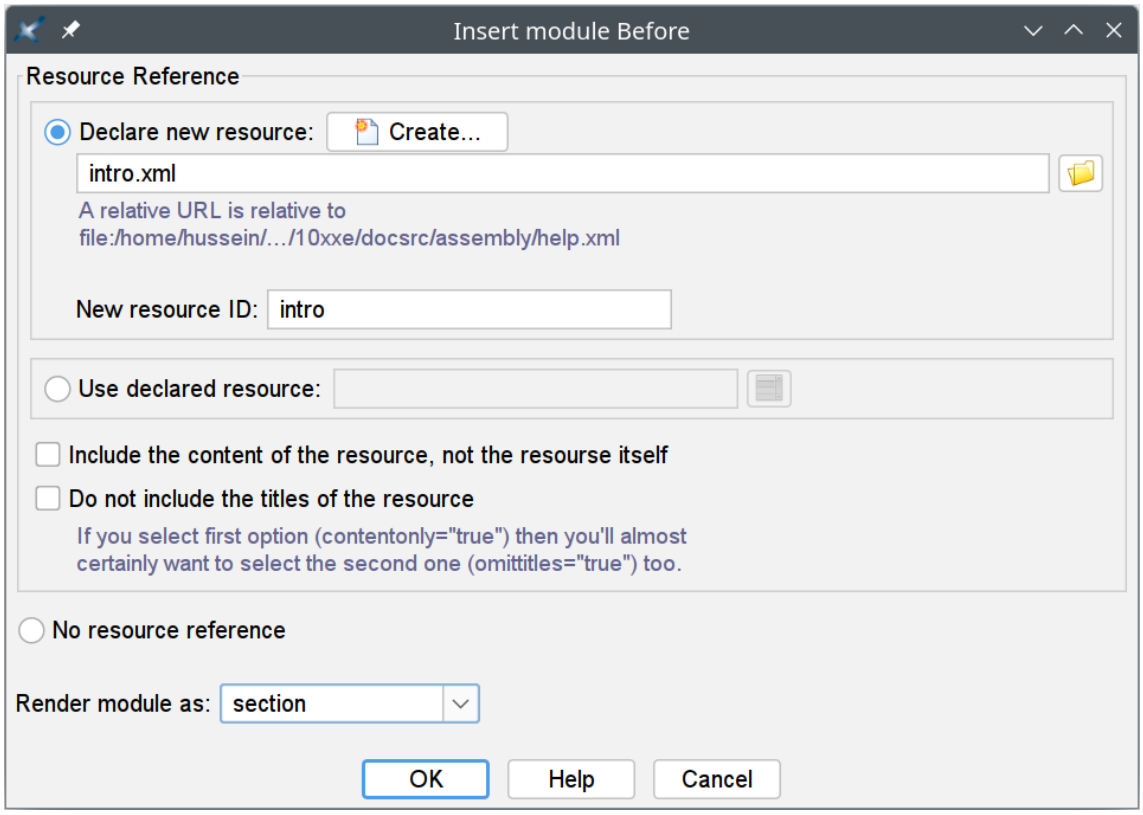
This can be done from within **XXE** using **Options** → **Customize Configuration** → **Customize Document Conversion Stylesheets**.

3. The Assembly toolbar

No need to first declare a `resource` and *then* create the `module` referencing it!

The dialog box [11] displayed by toolbar buttons allowing to insert or edit a module, is an *all-in-one dialog box*. It allows you to create or edit a module and, if needed to, at the same time, declare the resource referenced by this module. In fact, this dialog box even allows you to create a new document (`topic`, `section`, `chapter`, etc) corresponding to the resource.

Figure 4. The dialog box allowing to insert a `module` element and to edit a `module` or a `structure` element



Insert Module Before

Insert a `module` before selected `module`. Displays the "**Edit Module**" dialog box [11].

Insert Module Into

Insert a `module` as the last child of selected `module`. Displays the "**Edit Module**" dialog box [11].

Insert Module After

Insert a `module` after selected `module`. Displays the "**Edit Module**" dialog box [11].

Edit Module or Structure

Displays the "**Edit Module**" dialog box [11] in order to modify selected `module` or `structure`.

Configure Module or Structure

Add any of the following elements to selected `module` or `structure`: `merge`, `output`, `filterin`, `filterout`, `info`.

Move Up

Move selected element up, that is, swap it with its preceding sibling node. Requires the element to be explicitly selected.

Move Down

Move selected element down, that is, swap it with its following sibling node. Requires the element to be explicitly selected.

Promote Module

Decrease the nesting level of selected `module`. Clears its `renderas` attribute, if any.

Demote Module

Increase the nesting level of selected `module`. Clears its `renderas` attribute, if any.

Edit or Add Relationship

If an element is implicitly or explicitly selected anywhere in a `relationship` element, displays a dialog box [14] allowing to modify this `relationship`.

Otherwise, if an element is implicitly or explicitly selected anywhere in a `relationships` element, displays a dialog box [14] allowing to add a new `relationship` element to this `relationships` parent.

Otherwise, displays a dialog box [14] allowing to add a new `relationship` element to the *last* `relationships` element of the `assembly`. This `relationships` parent is created on the fly and added to the `assembly` if needed to.

Clean up Resources

Delete all `resource` elements not referenced in any `module`, `structure`, `resource` or `instance` element. However, this command never makes a `resources` element completely empty (which would make the `assembly` invalid) and never deletes `resources` elements.

Show Level

Displays a menu containing "**Show Level 1**", "**Show Level 2**", ..., "**Show Level 9**" items. "Show Level *N*" means: expand all the collapsible elements of the assembly up to nesting level *N* and recursively collapse all the collapsible elements having a nesting level greater than *N*.

Open All Resources R/O

Opens in read-only mode all the resources (`topic`, `chapter`, `section`, etc) referenced in the selected elements and their descendants.

Open All Resources

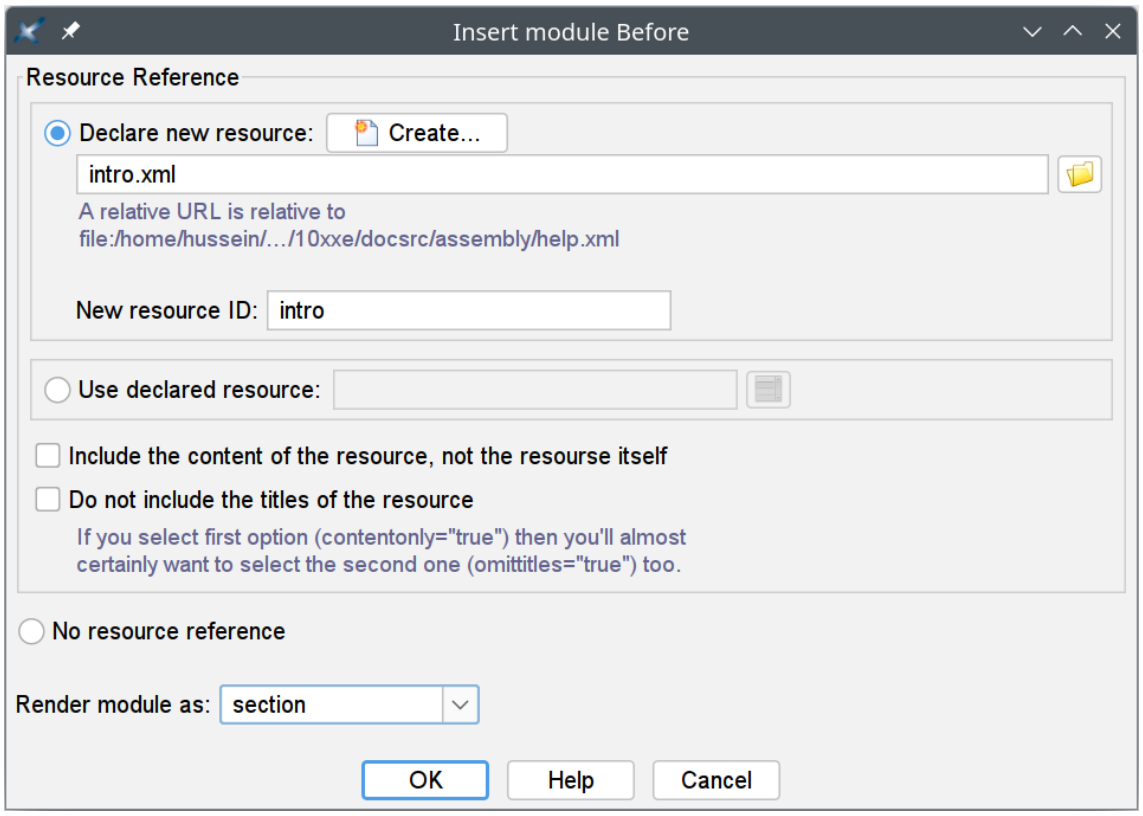
Opens in normal read-write mode all the resources (`topic`, `chapter`, `section`, etc) referenced in the selected elements and their descendants.

See also

- Section 3.1, “The "**Edit Module**" dialog box” [11]
- Section 3.2, “The "**Edit or Add Relationship**" dialog box” [14]
- Section 4, “Custom bindings” [16]

3.1. The "Edit Module" dialog box

Figure 5. The dialog box allowing to insert a `module` element and to edit a `module` or a `structure` element



- Check "**Declare new resource**" if you want edited `module` or `structure` to reference a `resource` which has not already been declared in the `assembly`.

  The referenced resource is found in the file specified in the text field below the checkbox. *This resource is automatically declared in a `resources` element of the `assembly`.*

  If the resource you want to reference in the edited `module` or `structure` does not exist yet, you can create it using the **Create** button. This button displays the same dialog box as menu item **File** → **New**.

  Field "**New resource ID**" allows to specify the value of the `xml:id` of the newly declared `resource`. This value is automatically computed for you if the use the **Create** or the "**Choose resource file**" button.

- Check "**Use declared resource**" if you want edited `module` or `structure` to reference a `resource` which has already been declared in the `assembly`. Button "**Selected declared resource**" displays a dialog box allowing to choose a declared resource.

- Check "**No resource reference**" if you don't want edited `module` or `structure` to reference a `resource`. In such case do not forget to fill the "**Render module as**" field. You may also want to use toolbar button "**Add to Module or Structure**" to add an `info` element to the edited `module` or `structure`.

When edited `module` or `structure` references a `resource`:

- Checking "**Include the content of the resource, not the resource itself**" lets you include in the realized document all the child elements of the root element of the resource. In practice, checking this button adds attribute `contentonly=true` to edited `module`.

- Checking "**Do not include the titles of the resource**" lets you include the resource without its `title`, `titleabbrev` and `subtitle` elements in the realized document. In practice, checking this button adds attribute `omittitles=true` to edited `module`.

- Checking both "**Include the content of the resource, not the resource itself**" and "**Do not include the titles of the resource**" is a useful combination which lets you include just the “body” of the resource in the realized document. See example below.

Example 1. Effect of attributes `contentonly=true` and `omittitles=true` on the content included in the realized document.

- When `contentonly=false` and `omittitles=false`, included content is:

```
<section xml:id="intro" xml:lang="en-US"
  xml:base="sections/introl.xml">
  <info>
    <title>Introduction</title>
    <titleabbrev>Intro.</titleabbrev>
    <author>
      <personname>John Doe</personname>
    </author>
  </info>
  <para>This is the introduction.</para>
</section>
```

- When `omittitles=true`, included content is:

```
<section xml:id="intro" xml:lang="en-US"
  xml:base="sections/introl.xml">
  <info>
    <author>
      <personname>John Doe</personname>
    </author>
  </info>
  <para>This is the introduction.</para>
</section>
```

- When `contentonly=true`, included content is:

```
<info xml:lang="en-US" xml:base="sections/introl.xml">
  <title>Introduction</title>
  <titleabbrev>Intro.</titleabbrev>
  <author>
    <personname>John Doe</personname>
  </author>
</info>
```

```
<para xml:lang="en-US"
  xml:base="sections/introl.xml">This is the introduction.</para>
```

- When `contentonly=true` and `omittitles=true`, included content is:

```
<para xml:lang="en-US"
  xml:base="sections/introl.xml">This is the introduction.</para>
```

Note that included content is **not**:

```
<info xml:lang="en-US" xml:base="sections/introl.xml">
  <author>
    <personname>John Doe</personname>
  </author>
</info>
<para xml:lang="en-US"
  xml:base="sections/introl.xml">This is the introduction.</para>
```

More information in *XMLmind Assembly Processor Manual - Implementation specificities*.

Render module as

This field allows to specify the value of the `renderas` attribute of the edited `module` or `structure`.

Default output format

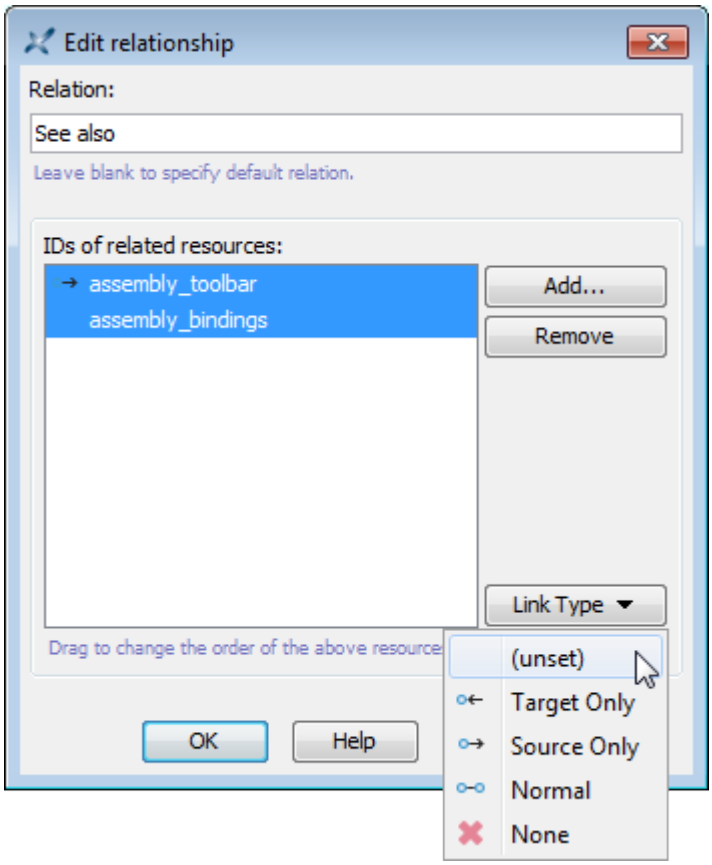
This field allows to specify the value of the `defaultformat` attribute of the edited `structure`.

See also

- Section 3, “The **Assembly** toolbar” [8]

3.2. The "Edit or Add Relationship" dialog box

Figure 6. The dialog box allowing to edit or add a `relationship` element



Relation

The content of this field specifies the name of the relation which exists between the resources specified in the ID list. It corresponds to the content of element `relationship/association`.

IDs of related resources

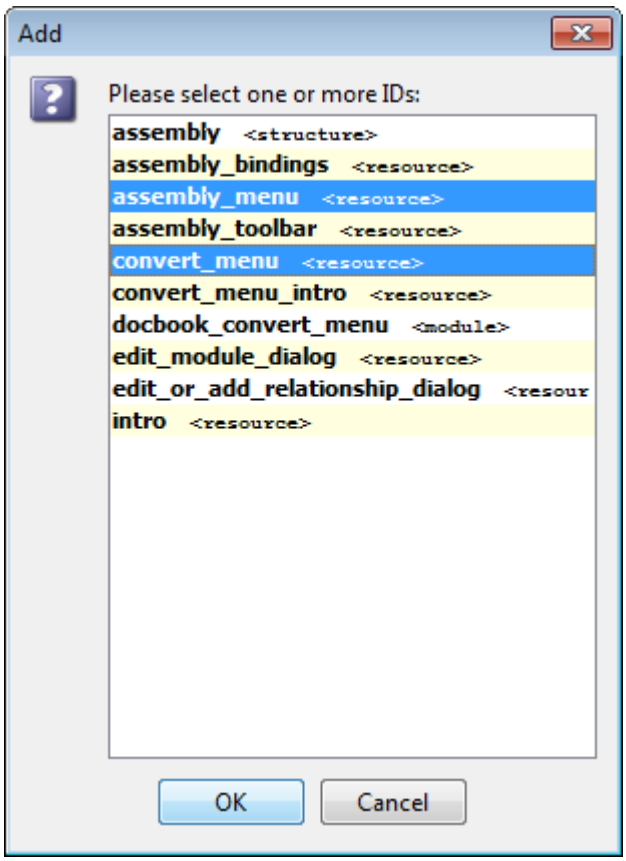
This list contains the IDs of the resources which are to be related. This list corresponds to a `relationship` element and its items to `instance` elements.

The `relationship` element corresponding to above figure [14] is thus:

```
<relationship>
  <association>See also</association>
  <instance linkend="assembly_toolbar" linking="sourceonly"/>
  <instance linkend="assembly_bindings"/>
</relationship>
```

Add

Clicking this button displays a dialog box allowing to select one or more resource IDs. Selected IDs are added to the list.



Note that these IDs come not only from the `xml:id` attributes of `resource` elements found in the `assembly`, but also from the `xml:id` attributes of `structure` and `module` (not having attribute `contentonly="true"`) elements. More information in *XMLmind Assembly Processor Manual - Implementation specificities*.

Remove

Selecting one or more IDs from the list and clicking **Remove** allows to remove these IDs from the list.

Link Type

Selecting one or more IDs from the list and clicking **Link Type** displays a popup menu allowing to change the "link type" of all selected items. The menu items correspond to the following values of attribute `instance/linking`:

Menu Item	Attribute Value
(unset)	Remove attribute `linking`.
Target Only	`targetonly`
Source Only	`sourceonly`
Normal	`normal`
None	`none`

More information in *XMLmind Assembly Processor Manual - Implementation specificities*.

See also

- Section 3, “The **Assembly** toolbar” [8]

4. Custom bindings

When a DocBook `assembly` is opened in XMLmind XML Editor, additional keyboard shortcuts which are specific to this kind of document are automatically made available to the user. This chapter contains a description of such keyboard shortcuts.

Action	Description
Up	If a `module` is selected, select preceding `module`; elsewhere, default behavior.
Down	If a `module` is selected, select following `module`; elsewhere, default behavior.
Enter	Insert Module After [9]
Shift+Enter	Insert Module Before [9]
Ctrl+Enter	Insert Module Into [9]
Esc e	Edit Module or Structure [9]
Alt+Shift+Up	Move Up [9]
Alt+Shift+Down	Move Down [10]
Alt+Shift+Left	Promote Module [10]
Alt+Shift+Right	Demote Module [10]
Ctrl+**Alt**+**C**	Like **Ctrl**+**Shift**+**C**, copy to the clipboard a reference to the selected nodes. However this variant is useful when the reference is intended to be pasted to *several different locations* in the same document. It leverages XInclude 1.1 features[a] which allow to avoid duplicate IDs caused by transclusions.
Double-click	On a `module`, `resource` or `instance`, open referenced resource [10]; elsewhere default behavior.
Esc o	Open Referenced Resource [10]
Esc O	Open Referenced Resource in Read-only Mode [10]
Esc r	Edit or Add Relationship [10]
Esc w	Clean up Resources [10]
Drag	Dragging selected `resource` or `transform` drags the value of its `href` attribute. Elsewhere, default drag behavior.
Drop	Dropping a file or URL onto a `module` displays a popup menu containing Insert Module Before [9], Insert Module Into [9], Insert Module After [9], Edit Module or Structure [9] and **Cancel**. Elsewhere, default drop behavior.

[a]The `xi:include` element implicitly created by pasting the reference has a `trans:idfixup="auto"` attribute.