XMLmind Ebook Compiler Manual

Hussein Shafie

XMLmind Software

35 rue Louis Leblanc,
78120 Rambouillet,
France,
Phone: +33 (0)9 52 80 80 37,
Web: www.xmlmind.com/ebookc
Email: ebookc-support@xmlmind.com

September 18, 2023

Table of Contents

List of Figures iii
List of Tables iv
List of Examples v
List of Equations vi

I. User guide 1
1. What is XMLmind Ebook Compiler? 2
2. Primer 3
3. Getting started 13
4. Handy features 16
4.1. Markdown support 16
4.1.1. Supported Markdown extensions 17
4.2. Automatic resource management 29
4.3. Conditional processing 32
4.4. Transclusion 34

II. Reference 36
5. Installation 37
6. Content of a source HTML page 39
6.1. Valid XHTML5 39
6.2. Headings 40
6.3. Examples 40
6.4. Equations 43
6.5. Admonitions 44
6.6. Footnotes 45
6.7. Cross-references 46
6.8. Index terms 48
7. Reference of ebook elements 51
7.1. Element `appendices` 51
7.2. Element `appendix` 52
7.3. Element `backmatter` 53
7.4. Element `body` 53
7.5. Element `book` 54
7.6. Element `chapter` 62
7.7. Element `content` 63
7.8. Element `frontmatter` 63
7.9. Element `head` 64
7.10. Element `headcommon` 66
7.11. Element `index` 67
7.12. Element `loe` 67
7.13. Element `lof` 68
7.14. Element `lot` 69
7.15. Element `lox` 70
7.16. Element `part` 71
7.17. Element `related` 72
7.18. Element `section` 73
7.19. Element `title` 74
7.20. Element `toc` 75

7.21. Common attributes 76
8. How it works 79
9. The `ebookc` command-line utility 81
10. XSLT stylesheets parameters 90
10.1. Parameters of the XSLT stylesheets used to convert an ebook specification to EPUB 90
10.2. Parameters of the XSLT stylesheets used to convert an ebook specification to Web Help 90
10.3. Parameters of the XSLT stylesheets used to convert an ebook specification to XSL-FO 96
10.3.1. Specifying a header or a footer 105

Appendices 110
A. Embedding `com.xmlmind.ebook.convert.Converter` 111

Index i

List of Figures

3-1. This manual, `manual.ebook`, opened in XMLmind XML Editor 15
6-1. The "**Edit index term**" dialog box of XMLmind XML Editor 48
8-1. XMLmind Ebook Compiler components 79
10-1. Page areas 104
10-2. Layout of a header 106

List of Tables

4-1. Table caption here 27
6-1. Admonition classes 45
9-1. Low-level processor options 82
9-2. Output formats 85

List of Examples

4-1. Example of conditional processing 33
4-2. Transclusion works fine within the same input HTML page 35
6-1. "Hello World" program in the C language 41

List of Equations

6-1. Special relativity 44

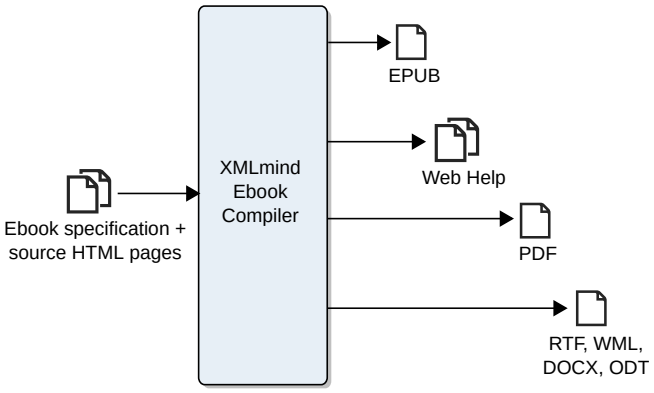
Part I. User guide

Table of Contents

1. What is XMLmind Ebook Compiler? 2
2. Primer 3
3. Getting started 13
4. Handy features 16
    4.1. Markdown support 16
        4.1.1. Supported Markdown extensions 17
    4.2. Automatic resource management 29
    4.3. Conditional processing 32
    4.4. Transclusion 34

Chapter 1. What is XMLmind Ebook Compiler?

XMLmind Ebook Compiler (**ebookc** for short) is a free, open source tool which can turn a set of HTML (or Markdown) pages into a self-contained *ebook*[1]. Supported output formats are: EPUB, *Web Help*, PDF[2], RTF, WML, DOCX (MS-Word) and ODT (OpenOffice/LibreOffice)[3].



You can of course use **ebookc** to create books having a simple structure like novels, but this tool also has all the features needed to create large, complex, reference manuals:

- Builds on topic-oriented structuring like DITA or DocBook 5.1. (Each source HTML page is expected to deal with a single topic.)
- Automatic generation of global and local table of contents.
- Automatic generation of a “back-of-the-book index”.
- Automatic numbering of parts, chapters, appendices, sections, figures, tables, examples and equations.
- Automatic creation of links between some user-specified book divisions.
- Automatic generation of text in cross-references.
- Footnote support.
- Conditional processing (also called *profiling*).
- Built-in support of XInclude (allows reuse of content at different locations in the book).

Being based on HTML, **ebookc** relies on CSS to create nicely formatted books and this, even for output formats like PDF and DOCX which are not directly related to HTML and CSS.

All in all, **ebookc** is an authoring and publishing tool *nearly as powerful* as DITA or DocBook and their advanced conversion toolkits, but being based on HTML and on CSS, it is *much easier to learn*, use and customize. Moreover you can create with it ebooks which are more interactive (audio, video, slide shows, multiple-choice questions, etc) than those created using DITA or DocBook.

[1] Here “ebook” shall be understood in the widest possible sense.

[2] Requires an XSL-FO processor like Apache FOP, RenderX XEP, Antenna House Formatter to be installed and registered with XMLmind Ebook Compiler (for example, using option `-foconverter`). We'll assume in this manual that you have downloaded and installed the distribution of XMLmind Ebook Compiler which includes Apache FOP.

[3] Requires XMLmind XSL-FO Converter to be installed and registered with XMLmind Ebook Compiler (using option `-xfc`).

Chapter 2. Primer

A book is an assembly of HTML pages

The basic idea is simple. You author a set of HTML pages and then you create an ebook specification assigning a role —part, chapter, section, appendix, etc— to each page. Example: `primer/book1.ebook`:

```
<book xmlns="http://www.xmlmind.com/schema/ebook"
      href="titlepage.html">
  <frontmatter>
    <toc/>
  </frontmatter>

  <chapter href="ch1.html"/>

  <chapter href="ch2.html"/>

  <appendix href="a1.html"/>
</book>
```

The HTML pages comprising a book may contain anything you want including CSS styles and links between the pages (e.g. `<a href="ch2.html#fig1">`). However make sure that this content is *valid XHTML*[4].

Once the ebook specification has been created, you can compile it using XMLmind Ebook Compiler and generate EPUB, *Web Help*, PDF[5], RTF, ODT, DOCX[6], etc. Examples:

```
ebookc book1.ebook out/book1.epub

ebookc book1.ebook out/book1.pdf
```

“Rich”, numbered, chapter titles

If you look at `out/book1.pdf`, you'll see that chapter and appendix titles are numbered and that these titles are copied verbatim from the `html/head/title` of the corresponding input HTML page.

It's of course possible to specify how book components should be numbered (if at all). It's also possible to replace the plain text titles of chapters and appendices by “rich” titles[7] by adding `ebook:head` child elements to the book divisions. Example: `primer/book2.ebook`:

---

[4] Preferably *valid XHTML5*, because `ebookc` anyway generates XHTML5 markup. “Plain HTML” cannot be parsed by `ebookc`.

[5] Requires an XSL-FO processor like Apache FOP, RenderX XEP, Antenna House Formatter to be installed and registered with XMLmind Ebook Compiler (for example, using option `-foconverter`). We'll assume in this manual that you have downloaded and installed the distribution of XMLmind Ebook Compiler which includes Apache FOP.

[6] Requires XMLmind XSL-FO Converter to be installed and registered with XMLmind Ebook Compiler (using option `-xfc`).

[7] That is, possibly containing the same elements as an HTML `p` (`em`, `kbd`, `img`, etc.)

```
<book xmlns="http://www.xmlmind.com/schema/ebook"
      xmlns:html="http://www.w3.org/1999/xhtml"
      href="titlepage.html" appendixnumber="A%1.">
  <frontmatter>
    <toc/>
  </frontmatter>

  <chapter href="ch1.html"/>

  <chapter href="ch2.html">
    <head>
      <title>“<html:em>Rich</html:em>” title of
      second chapter</title>
    </head>
  </chapter>

  <appendix href="a1.html"/>
</book>
```

The content of a `ebook:head` element specified this way is added to the `html/head` of the corresponding output HTML page, except for the `ebook:title` element which replaces `html/head/title`.

Assembling a book division rather than referencing an external file

We have already seen that it's possible to add a `ebook:head` child to elements like `book`[8], `chapter`, `appendix`, etc. Likewise, it's also possible to add a `ebook:body` child to any book division. Example: `primer/book3.ebook`:

```
<book xmlns="http://www.xmlmind.com/schema/ebook"
      xmlns:html="http://www.w3.org/1999/xhtml"
      appendixnumber="A%1">
  <head>
    <title>Title of this sample book</title>
  </head>
  <body>
    <content href="titlepage.html"/>
  </body>

  <frontmatter>
    <toc/>
  </frontmatter>

  <chapter href="ch1.html"/>

  <chapter href="ch2.html">
```

[8] In that matter, the root `book` element is no different from `part`, `chapter`, `appendix`, `section`, etc.

```
    <head>
      <title>“<html:em>Rich</html:em>” title of
      second chapter</title>
    </head>
  </chapter>

  <appendix href="a1.html"/>
</book>
```

In the above example, the content of the `html/body` element of file `titlepage.html` is “pulled” and added to the book. Several `ebook:content` child elements are allowed in an `ebook:body` element.

Controlling generated page names

When you generate multi-page HTML (e.g. Web Help) out of an ebook specification, it may be important to specify the names of the generated pages. It may also be useful to group several consecutive book divisions into the same output page.

This is specified using the `pagename` and `samepage` attributes of any book division. Example: `primer/book4.ebook`:

```
<book xmlns="http://www.xmlmind.com/schema/ebook"
      xmlns:html="http://www.w3.org/1999/xhtml"
      appendixnumber="A%1">
  <head>
    <title>Title of this sample book</title>
  </head>
  <body>
    <content href="titlepage.html"/>
  </body>

  <frontmatter>
    <toc/>
    <section href="intro.html" pagename="the introduction"/>
  </frontmatter>

  <chapter href="ch1.html">
    <section href="s1.html">
      <section href="s2.html" samepage="true"/>
    </section>
  </chapter>

  <chapter href="ch2.html">
    <head>
      <title>“<html:em>Rich</html:em>” title of
      second chapter</title>
    </head>
```

```
  </chapter>

  <appendix href="a1.html"/>
</book>
```

By default, each book division is created in its own file and the name of this file comes the `href` attribute of the book division. Web Help example:

```
ebookc -f webhelp book4.ebook out/book4
```

- Without attribute `pagename="the introduction"`, the introduction would have been generated in file `out/book4/intro.html`. With this attribute, the introduction is generated in file "`out/book4/the introduction.html`".
- Without attribute `samepage="true"`, the second section would have been generated in its own file `out/book4/s2.html`. With this attribute, the second section is appended to file `out/book4/s1.html`, also containing first section.

But wait a minute… HTML has not enough elements to write books

That's right, some semantic elements like admonitions, footnotes, etc, found in larger XML vocabularies like DITA or DocBook are missing from XHTML5. However, it's easy to emulate these missing elements by defining semantic values for the `class` attribute of standard HTML elements (typically `span` and `div`).

XMLmind Ebook Compiler has special support for the following semantic class names:

Semantic class	Description
`<figure class="role-equation">`	A “displayed equation” having a title (`figcaption`).
`<figure class="role-example">`	An example —for example a code snippet— having a title (`figcaption`).
`<pre class="role-listing-c-1">`	A code listing, possibly featuring line numbering and syntax coloring (class name suffix "`-c-1`" means: C language, first line number is 1).
`<blockquote class="role-note">`	Admonitions. Supported class names are: `role-note`, `role-attention`, `role-caution`, `role-danger`, `role-fastpath`, `role-important`, `role-notice`, `role-remember`, `role-restriction`, `role-tip`, `role-trouble`, `role-warning`.
`<span class="role-footnote">`	A short footnote, inline with the rest of the text.
`<a class="role-footnote-ref" href="#fn1">`	A call to footnote "`fn1`".
`<div class="role-footnote" id="fn1">`	Footnote "`fn1`".
`<a class="role-index-term">Cat</a>`	An index term. May be much more elaborate than the very simple example shown here.

Excerpts from file `primer/semantic_classes.html` which has been added to `primer/book5.ebook` as its second appendix:

```
...
<figure class="role-equation">
  <figcaption>Figure containing
  an equation</figcaption>
  <div>
    <math display="block"
          xmlns="http://www.w3.org/1998/Math/MathML">
      <mrow>
        <mi>E</mi>
        <mo>=</mo>
        <mrow>
          <mi>m</mi>
          <mo> </mo>
          <msup>
            <mi>c</mi>
            <mn>2</mn>
          </msup>
        </mrow>
      </mrow>
    </math>
  </div>
</figure>
...
<p>Short footnote<span class="role-footnote">Content of
short footnote.</span>.
...
<p>Simplest index term<a class="role-index-term">Cat</a>.
Other index term<a class="role-index-term">Cat<span
class="role-term">Siamese</span></a>...</p>
...
```

Because `primer/semantic_classes.html` contains figures, tables and index terms, the following book divisions have also been added to `primer/book5.ebook`:

```
...
  <frontmatter>
    <toc/>
    <lof/>
    <lot/>
    <lox/>
    <loe/>
    <section href="intro.html" pagename="the introduction"/>
...
```

```
  <backmatter>
    <index/>
  </backmatter>
...
```

`<lof/>` specifies that a List of Figures is to be generated as a front matter. `<lot/>` means: List of Tables. `<lox/>` means: List of Examples. `<loe/>` means: List of Equations.

Nicely formatted books

If you compile `primer/book5.ebook`, you'll get a *very dull* result whatever the output format:

```
ebookc -f webhelp book5.ebook out/book5

ebookc book5.ebook out/book5.pdf
```

This is caused by the fact that all the source HTML pages referenced by `book5.ebook` do not specify any CSS style.

It's a good practice to keep it this way because this allows separation of presentation and content. However, you'll want to create nice books, so the simplest and cleanest is to add CSS styles to the ebook specification (and not to each input HTML page).

If you do it like this:

```
<book xmlns="http://www.xmlmind.com/schema/ebook"
      xmlns:html="http://www.w3.org/1999/xhtml"
      appendixnumber="A%1">
  <head>
    <title>Title of this sample book</title>
    <html:link href="css/styles.css" rel="stylesheet"
               type="text/css"/>
  </head>
  ...
```

The above specification would *not* work because only the title page would get styled.

You need to use a `headcommon` element for that. The child elements of `headcommon` are automatically copied the `html/head` of all output HTML pages. Excerpts from `primer/book6.ebook`:

```
<book xmlns="http://www.xmlmind.com/schema/ebook"
      xmlns:html="http://www.w3.org/1999/xhtml"
      appendixnumber="A%1">
  <headcommon>
    <html:link href="css/styles.css" rel="stylesheet"
               type="text/css"/>
  </headcommon>

  <head>
```

```
    <title>Title of this sample book</title>
    <html:style>
div.role-book-title-div {
    text-align: center;
}

h1.role-book-title {
    margin: 4em 0;
    padding-bottom: 0;
    border-bottom-style: none;
}
    </html:style>
  </head>
  ...
```

In the above example:

- Element `ebook:head` may contain, not only `ebook:title`, but also any of the HTML elements allowed in `html/head`, namely `style`, `script`, `meta`, `link`. This facility is used here to give a specific style to the title page.
- Unlike `<blockquote class="role-note">` for example, which is found in the source HTML page, `<div class="role-book-title-div">` and `<h1 class="role-book-title">` are elements *generated* by XMLmind Ebook Compiler.

  Knowing about these elements is required to be able to give nice looks to the generated book. These elements and their class names are all listed in `template/skeleton.css`, with suggested CSS styles for some of these elements.

Leveraging `base.css`, the stock CSS stylesheet

As of version 1.4, the easiest way to add CSS styles to an ebook specification is to set attribute `includebasestylesheet` of element `book` to "`true`". This very simple setting guarantees to effortlessly create a nicely formatted book.

More precisely , attribute `includebasestylesheet="true"` instructs `ebookc` to include the *`ebookc_install_dir`*`/xsl/common/resources/base.css` stock CSS stylesheet in all the output HTML pages.

In the following example, we not only use `base.css`, but we also customize most of its colors by including a custom stylesheet called `custom_colors.css`:

```
<book xmlns="http://www.xmlmind.com/schema/ebook"
      xmlns:html="http://www.w3.org/1999/xhtml"
      includebasestylesheet="true">
  <headcommon>
    <html:link href="custom_colors.css" rel="stylesheet"
               type="text/css"/>
  </headcommon>
  ...
```

A sample color customization stylesheet is found in `template/custom_colors.css`.

What about output formats like PDF, RTF, DOCX?

The CSS styles specified in the ebook specification and in the source HTML pages are also used when generating output formats like PDF, RTF, DOCX, even if these formats are not directly related to HTML and CSS.

However in this case, CSS 2.1 support is partial. While there are no restrictions related to the use of CSS selectors, only the most basic CSS properties are supported. For example, generated content (e.g. `:before`) and floats are not supported at all.

There are two ways to work around this limitation:

1. Use simpler CSS styles when targeting output formats like PDF, RTF, DOCX. This is done using `@media screen` and `@media print`[9] rules. This is done in `primer/css/styles.css`:

```
blockquote.role-warning {
    font-size: 12px;
    background-color: #e1f5fe;
    color: #0288d1;
    padding: 12px 24px 12px 60px;
    margin: 16px 0;
}

blockquote.role-warning:before {
    float: left;
    content: url(star.svg);
    width: 16px;
    height: 16px;
    margin-left: -36px;
}

@media print {
    /* Floating generated content not supported.
       No need to leave room for the admonition icon. */
    blockquote.role-warning {
        padding-left: 24px;
        border-left: solid 5px #0288d1;
    }
}
```

[9] It's also possible to use `@media` *`XSL_FO_PROCESSOR_NAME`* rules, where *`XSL_FO_PROCESSOR_NAME`* is `FOP` (Apache FOP), `XEP` (RenderX XEP), `AHF` (Antenna House Formatter) or `XFC` (XMLmind XSL-FO Converter).

2. Some features like watermark images or admonition icons are directly implemented the XSLT stylesheets which generate XSL-FO[10]. Example:

```
ebookc -p use-note-icon yes book6.ebook out/book6.pdf

ebookc -f webhelp book6.ebook out/book6
```

Without XSLT stylesheet parameter `use-note-icon=yes`, admonitions in `out/book6.pdf` would have no icons.

Such parameter is not needed when generating Web Help (like EPUB, an HTML+CSS-based output format) because admonition icons are specified in CSS stylesheet `primer/css/styles.css`.

Creating links between book divisions

An book is specified as an assembly of source HTML pages. If you want to reuse some of these HTML pages to author other books, it is recommended to avoid creating links (e.g. `<a href="ch2.html#fig1">`) between these pages.

Fortunately, there is a simple way to create links between book divisions, which is using the `ebook:related` element. Excerpts from `primer/book7.ebook`:

```
...
<chapter href="ch1.html" xml:id="ch1">
  <related ids="ch1 ch2 a1" relation="See also"/>

  <section href="s1.html">
    <section href="s2.html" samepage="true"/>
  </section>
</chapter>

<chapter href="ch2.html" xml:id="ch2">
  <head>
    <title>“<html:em>Rich</html:em>” title of
    second chapter</title>
  </head>

  <related ids="ch1 ch2 a1" relation="See also"/>
</chapter>

<appendix href="a1.html" xml:id="a1">
  <related ids="ch1 ch2 a1" relation="See also"/>
</appendix>
...
```

[10] A standard, intermediate page-layout format which is then used by XSL-FO processors like Apache FOP or XMLmind XSL-FO Converter to generate PDF, RTF, DOCX, etc.

See links automatically generated in first chapter, second chapter and first appendix by running for example:

```
ebookc -f webhelp book7.ebook out/book7
```

Conditionally excluding some content from the generated book

This feature called *conditional processing* or *profiling* has many uses, the most basic one being to include or exclude some content depending on the chosen output format. For example, some source HTML pages may contain interactive content (e.g. a feedback form) and this interactive content simply cannot be rendered in PDF or DOCX.

In order to conditionally exclude some content from the generated book, you must first "mark" the conditional sections using `data-*` attributes. Excerpts from `primer/book8.ebook`:

```
...
<backmatter data-output-format="docx odt pdf rtf wml">
  <index/>
</backmatter>
...
```

Excerpts from `primer/intro.html`:

```
...
<blockquote class="role-tip"
            data-output-format="epub html webhelp">
  <p>This document is also available in PDF ... format.</p>
</blockquote>
...
```

You may specify one or more conditional processing `data-*` attribute on any element. Choose the attribute names you want. Such conditional processing `data-*` attribute may contain one or more values separated by space characters. Choose the attribute values you want.

If you generate a single HTML page by running:

```
ebookc book8.ebook out/book8_no_profile.html
```

the marked sections will *not* be excluded because XMLmind Ebook Compiler does not associate any special meaning to attribute `data-output-format`. However if you run:

```
ebookc -p profile.output-format html book8.ebook out/book8.html
```

then file `out/book8.html` will not have an index. Option "`-p profile.output-format html`" reads as: unless an element has no `data-output-format` attribute or has a `data-output-format` attribute containing "`html`", exclude this element from the generated content.

If you run:

```
ebookc -p profile.output-format pdf book8.ebook out/book8.pdf
```

then the introduction will not contain the tip about the availability of the document in PDF format.

Chapter 3. Getting started

Installing XMLmind Ebook Compiler

How to install XMLmind Ebook Compiler is explained in Chapter 5. Installation.

Writing an ebook specification

You have learned in Chapter 2. Primer:

- What is an ebook specification. The corresponding reference is found in Chapter 7. Reference of ebook elements.
- What an ebook page may contain. The corresponding reference is found in Chapter 6. Content of a source HTML page.

You'll find a template for your ebook specification in *`ebookc_install_dir`*`/doc/manual/template/template.ebook`. The recommended extension for these files is "`.ebook`".

Writing a CSS stylesheet for your ebooks

If you want your ebook to look good, the simplest is to set attribute `includebasestylesheet` of element `book` to "`true`" as explained in *Leveraging `base.css`, the stock CSS stylesheet*.

Alternatively, you may want to use a custom CSS stylesheet developed from scratch, starting from the template found in *`ebookc_install_dir`*`/doc/manual/template/skeleton.css`. In this case, as explained in *Nicely formatted books*, make sure to add this kind of link to the `headcommon` element of your root `book` element:

```
<headcommon>
  <html:link href="my_custom_stylesheet.css" rel="stylesheet"
             type="text/css"/>
</headcommon>
```

Compiling an ebook specification

An ebook specification is compiled using a command-line tool called `ebookc`. Run *`ebookc_install_dir`*`/bin/ebookc.bat` on Windows and *`ebookc_install_dir`*`/bin/ebookc` on the Mac and on Linux.

Example, convert this manual to EPUB:

```
C:\ebookc_1_9_0\docsrc\manual> ..\..\bin\ebookc.bat manual.ebook out\manual.epub
```

Example, convert this manual to Web Help (output directory being "`out\manual_webhelp\`"):

```
C:\ebookc_1_9_0\docsrc\manual> ..\..\bin\ebookc.bat -f webhelp¬
  manual.ebook out\manual_webhelp
```

Example, convert this manual to DOCX using a copy of XMLmind XSL-FO Converter installed in "`C:\xfc\`":

```
C:\ebookc_1_9_0\docsrc\manual> ..\..\bin\ebookc.bat¬
  -xfc C:\xfc\bin\fo2rtf.bat¬
  manual.ebook out\manual.docx
```


WARNING

XMLmind XSL-FO Converter Evaluation Edition (download page) generates output containing *random duplicate letters*. This makes this edition useless for any purpose other than evaluating XMLmind XSL-FO Converter. Of course, this does not happen with XMLmind XSL-FO Converter Professional Edition!

Example, convert this manual to PDF using a copy of RenderX XEP installed in "`C:\xep\`":

```
C:\ebookc_1_9_0\docsrc\manual> ..\..\bin\ebookc.bat¬
  -xep C:\xep\xep.bat¬
  manual.ebook out\manual.pdf
```

Tip

To avoid specifying options `-xep` and `-xfc` each time you run `ebookc`, the simplest if to create once for all an `ebookc.options` file in the `ebookc` user preferences directory. This directory is:

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

Your `ebookc.options` file would contain:

```
-xep C:\xep\xep.bat
-xfc C:\xfc\bin\fo2rtf.bat
```

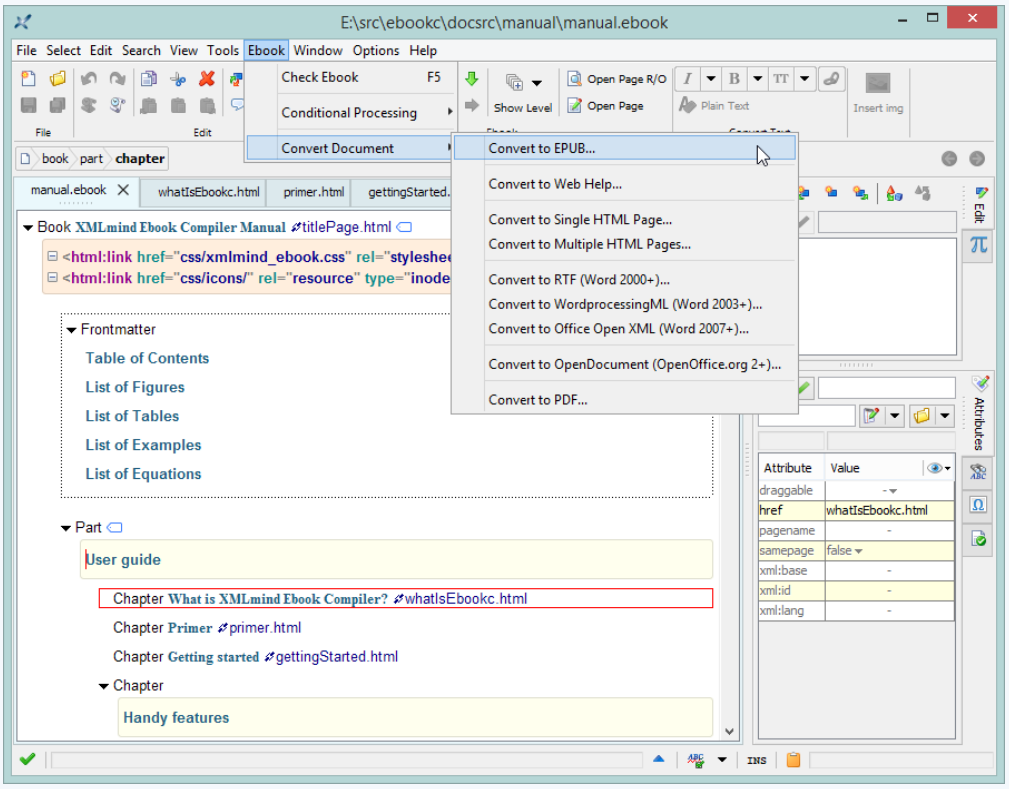
What if you just want to quickly experiment with XMLmind Ebook Compiler?

The simplest is to download and install XMLmind XML Editor Personal Edition from http://www.xmlmind.com/xmleditor/download.shtml.

You can then open this document —"*XMLmind Ebook Compiler Manual*", an ebook specification found in *ebookc_install_dir*`/docsrc/manual/manual.ebook`— in XMLmind XML Editor and use menu **Ebook** > **Convert Document** to convert it to any format you want.

In fact, XMLmind XML Editor fully supports the creation of ebook specifications and ebook pages. This support is as extensive as the DITA or DocBook support in XMLmind XML Editor.

Figure 3-1. *This manual, `manual.ebook`, opened in XMLmind XML Editor*



Chapter 4. Handy features

Table of Contents

4.1. Markdown support 16
4.1.1. Supported Markdown extensions 17
4.2. Automatic resource management 29
4.3. Conditional processing 32
4.4. Transclusion 34

4.1. Markdown support

In addition to HTML, an ebook page may be written in Markdown. However for this to work, the file extension of the page written in Markdown must be `md`, `markdown`, `mdown`, `mkdn`, `mdwn`, `mkd`, `rmd`, `text` or `txt`.

Note

The encoding of a Markdown file is, by default, the system encoding (e.g. `window-1252` on a Western PC).

If you want to explicitly specify the encoding of a Markdown file, please save your file with a `UTF-8` or `UTF-16` BOM (Byte Order Mark) or add an *encoding directive* inside a comment anywhere at the beginning of your file. Example:

```
<!-- -*- coding: iso-8859-1 -*- -->

Heading
=======

## Sub-heading

Paragraphs are separated
by a blank line.
```

The above example should work fine because ebookc understands the GNU Emacs file variable called `coding`.

Out of the box, the Markdown parser is configured to support the commonmark 0.28 “Markdown dialect” plus all the following extensions:

- Abbreviations
- Admonitions
- Attributes
- Definition lists
- Footnotes
- Ins
- Strikethrough and subscript
- Media tags
- Superscript

- Tables
- Typographic characters
- YAML front matter

However, thanks to the flexmark-java software component used by `ebookc` to implement Markdown support, all this can be configured by passing some `load.markdown.XXX` options to `ebookc`.

For example, pass

- `-p load.markdown.profile GITHUB`
- `-p load.markdown.less-extensions true`
- `-p load.markdown.gfm-tasklist true`

to `ebookc` in order to parse the Github-flavored Markdown dialect and to enable a minimal set of extensions plus the task lists syntax extension.

Supported "Markdown dialects" are `COMMONMARK`, `COMMONMARK_0_26`, `COMMONMARK_0_27`, `COMMONMARK_0_28`, `FIXED_INDENT`, `KRAMDOWN`, `MARKDOWN`, `GITHUB_DOC`, `GITHUB`, `MULTI_MARKDOWN`, `PEGDOWN`, `PEGDOWN_STRICT`. See Markdown Processor Emulation.

Parameter `-p load.markdown.less-extensions true` is a shorthand parameter instructing `ebookc` to reset its extensions to the following minimal set of extensions:

- Strikethrough and subscript
- Superscript
- Tables
- YAML front matter

The above minimal set of extensions corresponds to what's described in the Markdown Cheatsheet.

All supported Markdown syntax extensions are documented in Section 4.1.1. Supported Markdown extensions.

4.1.1. Supported Markdown extensions

Abbreviations

Converts plain text abbreviations (e.g. IBM) to `<abbr>` elements.

This Markdown syntax extension is enabled by default. In order to disable it, pass parameter `-p load.markdown.abbreviation false` to `ebookc`.

Example:

```
The HTML specification is maintained by the W3C.

*[HTML]: Hyper Text Markup Language
*[W3C]: World Wide Web Consortium
```

is converted to:

```
<p>The <abbr title="Hyper Text Markup Language">HTML</abbr> specification
is maintained by the <abbr title="World Wide Web Consortium">W3C</abbr>.</p>
```

which is rendered as:

The **HTML** specification is maintained by the **W3C**.

Admonitions

Syntax for creating admonitions such as notes, tips, warnings, etc.

This Markdown syntax extension is enabled by default. In order to disable it, pass parameter `-p load.markdown.admonition false` to `ebookc`.

After the "`!!!`" tag, the admonition type must be one of "`note`", "`attention`","`caution`", "`danger`", "`fastpath`", "`important`", "`notice`", "`remember`", "`restriction`", "`tip`","`trouble`", "`warning`".

A note example not having a title:

```
!!! note ""
    Support is limited to bug reports.
```

is converted to:

```
<blockquote class="role-note">
  <p>Support is limited to bug reports.</p>
</blockquote>
```

which is rendered as:

Note

Support is limited to bug reports.

A tip example having a title:

```
!!! tip "How do you do a hard reboot on an iPad?"
    Press and hold both the **Home** and **Power** buttons
    until your iPad&reg; reboots.

    You can release both buttons when you see Apple&reg; logo.
```

is converted to:

```
<blockquote class="role-tip">
  <h4 class="role-admonition-title">How do you do a hard reboot on an iPad?</h4>
  <p>Press and hold both the <strong>Home</strong> and <strong>Power</strong>
  buttons until your iPad&reg; reboots.</p>
  <p>You can release both buttons when you see Apple&reg; logo.</p>
</blockquote>
```

which is rendered as:

How do you do a hard reboot on an iPad?

Press and hold both the **Home** and **Power** buttons until your iPad® reboots.

You can release both buttons when you see Apple® logo.

Attributes

Syntax for adding attributes to the generated **HTML** elements:

```
attributes -> '{' attribute_spec ( S attribute_spec)* '}'

attribute_spec ->   name=value
                  | name='value'
                  | name="value"
                  | #id
                  |.class
```

An easy rule to remember

If an `{...}` specification is separated by space characters from some plain text (e.g. `some plain text {...}`) then the attributes are added to the parent element of the text.

This Markdown syntax extension is enabled by default. In order to disable it, pass parameter `-p load.markdown.attributes false` to `ebookc`.

Example:

```
The *circumference { .first-term }* is the length of one circuit along the
circle, or the distance around the circle. {#circumference title="See
https://en.wikipedia.org/wiki/Circle"}
```

is converted to:

```
<p id="circumference" title="See https://en.wikipedia.org/wiki/Circle">The <em
class="first-term">circumference</em> the length of one circuit along the
circle, or the distance around the circle.</p>
```

which is rendered as:

The *circumference* is the length of one circuit along the circle, or the distance around the circle.

Pitfall

By default, heading IDs are not “rendered” in **HTML** (which is somewhat counterintuitive). You must pass
`-p load.markdown.renderer.RENDER_HEADER_ID true`
to `ebookc` get them “rendered”.

Automatic links

Turns plain text URLs and email addresses into `<a href="...">` elements.

This Markdown syntax extension is disabled by default. In order to enable it, pass parameter `-p load.markdown.autolink true` to `ebookc`.

Example:

```
Please send your bug reports to ebookc-support@xmlmind.com, a public,
moderated, mailing list. More information in
http://www.xmlmind.com/ebookc/support.html.
```

is converted to:

```
<p>Please send your bug reports to <a
href="mailto:ebookc-support@xmlmind.com">ebookc-support@xmlmind.com</a>, a
public, moderated, mailing list. More information in <a
href="http://www.xmlmind.com/ebookc/support.html"
>http://www.xmlmind.com/ebookc/support.html</a>.</p>
```

which is rendered as:

Please send your bug reports to ebookc-support@xmlmind.com, a public, moderated, mailing list. More information in http://www.xmlmind.com/ebookc/support.html.

Definition lists

Syntax for creating definition lists, that is `<dl>`, `<dt>` and `<dd>` elements.

This Markdown syntax extension is enabled by default. In order to disable it, pass parameter `-p load.markdown.definition false` to `ebookc`.

Example:

```
Glossary:

LED
: Light emitting diode.

ABS
: Antilock braking system.
```

```
ESC
ESP
: Electronic stability control, also known as Electronic Stability Program.

: On motorcycles, ESC/ESP is called *Traction Control*.

  > Ducati was one of the first to introduce a true competition-level
  > traction control system (**DTC**) on a production motorcycle.

EBA
: Emergency brake assist.
```

is converted to:

```
<p>Glossary:</p>
<dl>
  <dt>LED</dt>
  <dd><p>Light emitting diode.</p></dd>

  <dt>ABS</dt>
  <dd><p>Antilock braking system.</p></dd>

  <dt>ESC</dt>
  <dt>ESP</dt>
  <dd><p>Electronic stability control, also known as Electronic Stability
  Program.</p></dd>
  <dd><p>On motorcycles, ESC/ESP is called <em>Traction
  Control</em>.</p>
  <blockquote><p>Ducati was one of the first to introduce a true
  competition-level traction control system (<strong>DTC</strong>)
  on a production motorcycle.</p></blockquote></dd>

  <dt>EBA</dt>
  <dd><p>Emergency brake assist.</p></dd>
</dl>
```

which is rendered as:

Glossary:

LED
: Light emitting diode.

ABS
: Antilock braking system.

ESC
ESP
: Electronic stability control, also known as Electronic Stability Program.

  On motorcycles, ESC/ESP is called *Traction Control*.

  > Ducati was one of the first to introduce a true competition-level traction control system (**DTC**) on a production motorcycle.

EBA
: Emergency brake assist.

Remember

Remember that:

- The leading ":" character of a definition must be followed by one or more space characters.
- Terms must be separated from the previous definition by a blank line.
- A blank line is not allowed between two consecutive terms.
- A blank line is allowed before a definition.

Footnotes

Syntax for creating footnotes and footnote references.

This Markdown syntax extension is enabled by default. In order to disable it, pass parameter `-p load.markdown.footnotes false` to `ebookc`.

Example:

```
The differences between the programming languages C++[^1] and Java can be
traced to their heritage.

[^1]: The C++ Programming Language by Bjarne Stroustrup.

C++[^1] was designed for systems and applications programming, extending the
procedural programming language C[^2].

[^2]: The C Programming Language by Brian Kernighan and Dennis Ritchie.

      Originally published in 1978.
```

is converted to:

```
<p>The differences between the programming languages
C++<a class="role-footnote-ref" href="#__FN1"></a> and Java can be traced to
```

```
their heritage.</p>

<div class="role-footnote" id="__FN1">
  <p>The C++ Programming Language by Bjarne Stroustrup.</p>
</div>

<p>C++<aclass="role-footnote-ref" href="#__FN1"></a> was designed for systems
and applications programming, extending the procedural programming language
C<a class="role-footnote-ref" href="#__FN2"></a>.</p>

<div class="role-footnote" id="__FN2">
  <p>The C Programming Language by Brian Kernighan and Dennis
  Ritchie.</p>
  <p>Originally published in 1978.</p>
</div>
```

which is rendered as:

The differences between the programming languages C++[11] and Java can be traced to their heritage.

C++[11] was designed for systems and applications programming, extending the procedural programming language C[12].

Ins

Converts tagged text "`++something new++`" to `<ins>something new</ins>`, which is rendered as: something new

This Markdown syntax extension is enabled by default. In order to disable it, pass parameter `-p load.markdown.ins false` to `ebookc`.

Strikethrough and subscript

Converts

- tagged text "`~~something deleted~~`" to `<del>something deleted</del>`, which is rendered as: ~~something deleted~~
- tagged text "`~a subscript~`" to `<sub>a subscript<sub/>`, which is rendered as: $_{\text{a subscript}}$

This Markdown syntax extension is enabled by default. In order to disable it, pass parameter `-p load.markdown.gfm-strikethrough false` to `ebookc`.

Superscript

Converts tagged text "`^a superscript^`" to `<sup>a superscript</sup>`, which is rendered as: $^{\text{a superscript}}$

[11] The C++ Programming Language by Bjarne Stroustrup.

[12] The C Programming Language by Brian Kernighan and Dennis Ritchie.
Originally published in 1978.

This Markdown syntax extension is enabled by default. In order to disable it, pass parameter `-p load.markdown.superscript false` to `ebookc`.

Media tags

Converts prefixed links to audio, embed, picture and video HTML5 elements.

- `!A[Text](links)` - audio. *Links* is one or more links separated by character "`|`".
- `!E[Text](links)` - embed.
- `!P[Text](links)` - picture.
- `!V[Text](links)` - video.

Audio example:

```
Audio example: !A[Falcon calling](media/falcon.mp3|media/falcon.wav).
```

is converted to:

```
<p>Audio example: <audio controls="" title="Falcon calling">
  <source src="media/falcon.mp3" type="audio/mpeg">
  <source src="media/falcon.wav" type="audio/wav">
  Your browser does not support the audio element.
</audio>.</p>
```

which is rendered as:

Audio example: audio.

Video example:

```
Video example: !V[Funny big bunny](media/bbb.mp4).
```

is converted to:

```
<p>Video example: <video controls="" title="Funny big bunny">
  <source src="media/bbb.mp4" type="video/mp4">
  Your browser does not support the video element.
</video>.</p>
```

which is rendered as:

Video example: video.

This Markdown syntax extension is enabled by default. In order to disable it, pass parameter `-p load.markdown.media-tags false` to `ebookc`.

Tables

Converts pipe "`|`" delimited text to `<table>` elements.

This Markdown syntax extension is enabled by default. In order to disable it, pass parameter `-p load.markdown.tables false` to `ebookc`.

Simple table example:

```
| Header 1 | Header 2 | Header 3 |
| -------- | -------- | -------- |
| Cell 1,1 | Cell 1,2 | Cell 1,3 |
| Cell 2,1 | Cell 2,2 | Cell 2,3 |
```

is converted to:

```
<table border="1">
  <thead>
    <tr>
      <th>Header 1</th>
      <th>Header 2</th>
      <th>Header 3</th>
    </tr>
  </thead>
  <tbody>
    <tr>
      <td>Cell 1,1</td>
      <td>Cell 1,2</td>
      <td>Cell 1,3</td>
    </tr>
    <tr>
      <td>Cell 2,1</td>
      <td>Cell 2,2</td>
      <td>Cell 2,3</td>
    </tr>
  </tbody>
</table>
```

which is rendered as:

Header 1	Header 2	Header 3
Cell 1,1	Cell 1,2	Cell 1,3
Cell 2,1	Cell 2,2	Cell 2,3

Table example having centered and right-aligned columns:

```
| Header 1 | Header 2        | Table Header 3 |
| -------- | :-------------: | -------------: |
| Cell 1,1 | Table cell 1,2  | Cell 1,3       |
| Cell 2,1 | Cell 2,2        | Cell 2,3       |
```

is converted to:

```
<table border="1">
  <thead>
    <tr>
      <th>Header 1</th>
      <th style="text-align: center;">Header 2</th>
      <th style="text-align: right;">Table Header 3</th>
    </tr>
  </thead>
  <tbody>
    <tr>
      <td>Cell 1,1</td>
      <td style="text-align: center;">Table cell 1,2</td>
      <td style="text-align: right;">Cell 1,3</td>
    </tr>
    <tr>
      <td>Cell 2,1</td>
      <td style="text-align: center;">Cell 2,2</td>
      <td style="text-align: right;">Cell 2,3</td>
    </tr>
  </tbody>
</table>
```

which is rendered as:

Header 1	Header 2	Table Header 3
Cell 1,1	Table cell 1,2	Cell 1,3
Cell 2,1	Cell 2,2	Cell 2,3

Table example having cells spanning several columns and a caption:

```
| Header 1 | Header 2 | Header 3 |
| -------- | -------- | -------- |
| Cell 1,1 + 1,2     || Cell 1,3 |
| Cell 2,1 + 2,2 + 2,3         |||
| Cell 3,1 | Cell 3,2 | Cell 3,3 |
[Table caption here]
```

is converted to:

```
<table border="1">
  <caption>Table caption here</caption>
  <thead>
    <tr>
```

```
      <th>Header 1</th>
      <th>Header 2</th>
      <th>Header 3</th>
    </tr>
  </thead>
  <tbody>
    <tr>
      <td colspan="2">Cell 1,1 + 1,2</td>
      <td>Cell 1,3</td>
    </tr>
    <tr>
      <td colspan="3">Cell 2,1 + 2,2 + 2,3</td>
    </tr>
    <tr>
      <td>Cell 3,1</td>
      <td>Cell 3,2</td>
      <td>Cell 3,3</td>
    </tr>
  </tbody>
</table>
```

which is rendered as:

Table 4-1. *Table caption here*

Header 1	Header 2	Header 3
Cell 1,1 + 1,2		Cell 1,3
Cell 2,1 + 2,2 + 2,3		
Cell 3,1	Cell 3,2	Cell 3,3

Typographic characters

Converts

- "`'`" to apostrophe `’`, which is rendered as in word: "don’t"
- "`...`" and "`. . .`" to ellipsis `…`, which are both rendered as: …
- "`--`" to en dash `–`, which is rendered as: –
- "`---`" to em dash `—`, which is rendered as: —
- single quoted `'some text'` to `‘some text’`, which is rendered as: ‘some text’
- double quoted `"some text"` to `“some text”`, which is rendered as: “some text”
- double angle quoted `<<some text>>` to `«some text»`, which is rendered as: «some text»

This Markdown syntax extension is enabled by default. In order to disable it, pass parameter `-p load.markdown.typographic false` to `ebookc`.

If you don’t want some of the above plain text sequences to be processed, specify:

`-p load.markdown.typographic.ENABLE_QUOTES false`
: Do not process single quotes, double quotes, double angle quotes.

`-p load.markdown.typographic.ENABLE_SMARTS false`
: Do not process "`'`", "`...`", "`. . .`", "`--`", "`---`".

YAML front matter

Syntax for adding metadata to the generated **HTML** document, that is, for adding `<head>/<title>` and/or `<head>/<meta>` elements.

These metadata are specified by key/value pairs written using a subset of the YAML (see also http://yaml.org/) syntax.

This Markdown syntax extension is enabled by default. In order to disable it, pass parameter `-p load.markdown.yaml-front-matter false` to `ebookc`.

Example:

```
---
title: The C Programming Language
author:
  - Brian W. Kernighan
  - Dennis Ritchie
description: |
    One of the best-selling programming books published
    in the last fifty years, "K&R" has been called everything
    from the "bible" to "a landmark in computer science" and
    it has influenced generations of programmers.
date: 1988-03-22
---
```

is converted to:

```
<head>
  <title>The C Programming Language</title>
  <meta content="Brian W. Kernighan" name="author" />
  <meta content="Dennis Ritchie" name="author" />
  <meta content="One of the best-selling programming books published
                 in the last fifty years, &amp;quot;K&amp;amp;R&amp;quot; has been called
                 everything from the &amp;quot;bible&amp;quot; to
                 &amp;quot;a landmark in computer science&amp;quot; and it has
                 influenced generations of programmers." name="description" />
  <meta content="1988-03-22" name="date" />
</head>
```

Other extensions

The following Markdown syntax extensions are also supported:

- anchorlink
- aside
- emoji
- enumerated-reference
- gfm-issues
- gfm-tasklist
- gfm-users
- macros
- toc
- wikilink
- youtube-embedded

All the above extensions are disabled by default. In order to enable an extension, pass parameter `-p load.markdown.EXTENSION_NAME true` to `ebookc`. For example: `-p load.markdown.emoji true`

Any extension listed in this section may be parameterized by passing parameter `-p load.markdown.EXTENSION_NAME.PARAMETER_NAME PARAMETER_VALUE`[13] to `ebookc`. Examples:

- `-p load.markdown.emoji.ATTR_IMAGE_SIZE 16`
- `-p load.markdown.emoji.ATTR_ALIGN ""`
- `-p load.markdown.emoji.USE_IMAGE_TYPE IMAGE_ONLY`
- `-p load.markdown.emoji.USE_SHORTCUT_TYPE ANY_GITHUB_PREFERRED`

With the above emoji parameters, "`:smile:`" is rendered as: 😄

More generally, the Markdown parser (pseudo *EXTENSION_NAME* is "`parser`") and the **HTML** renderer (pseudo *EXTENSION_NAME* is "`renderer`") may also be parameterized this way. For example, automatically generate an ID for all headings not already having an ID **and** "render" all heading IDs in **HTML**[14]:
`-p load.markdown.renderer.GENERATE_HEADER_ID true`
`-p load.markdown.renderer.RENDER_HEADER_ID true`.

More information about extensions and their parameters in Extensions (flexmark-java is the software component used by ebookc to parse Markdown and convert it to **HTML**).

4.2. Automatic resource management

XMLmind Ebook Compiler automatically copies all the resources referenced by the ebook specification and the input HTML pages to the output directory in order to create a self-contained deliverable. Creating self-contained deliverables is generally desirable and for some output formats, like EPUB, this is really required.

For example, if you run (single-page HTML output format):

```
ebookc doc.ebook out/doc.html
```

all the resources of `doc.ebook` are copied to `out/doc_files/`.

Other example, if you run:

[13] The only types supported for *PARAMETER_VALUE* are: string, boolean (`true` or `false`), integer and any enumerated type.

[14] By default, heading IDs are not "rendered" in **HTML**, which is somewhat counterintuitive.

```
ebookc -f webhelp doc.ebook out/webhelp/
```

all the resources of `doc.ebook` are copied to `out/webhelp/_res/`.

What is a resource?

By default, XMLmind Ebook Compiler considers that any file [15] referenced by the ebook specification or an input HTML page using a *relative URI is a resource*. This is generally the case of images, audio and video files, CSS stylesheets, scripts files referenced by the ebook specification and the input HTML pages.

In this example, image "`cc-by-sa.png`" is obviously a resource and file "`creativecommons.html`" not being an input HTML page, is also considered to be a resource:

```
<p>All the above tutorials are licensed under the
<a href="creativecommons.html"><img src="cc-by-sa.png"
alt="CC BY-SA"/>Creative Commons License</a>,
which means that everyone is welcome to distribute, modify, translate, etc,
any of them.</p>
```

How to specify "*not a resource; do not copy it and keep its relative URI as is*"?

The automatic resource management of **ebookc** may be turned off globally by setting option `proc.ignoreresources` to "`true`".

If you want to specify that only some of the resources of an ebook are external and as such, should not be processed by **ebookc**, please use

- value "`external-resource`" for standard attribute `rel` (HTML link elements have this attribute);
- proprietary attribute `data-external-resource` for elements like `img` which do not have attribute `rel`.

Example:

```
<p>All the above tutorials are licensed under the
<a href="creativecommons.html"
rel="external-resource"><img src="cc-by-sa.png" alt="CC BY-SA"
data-external-resource=""/>Creative Commons License</a>,
which means that everyone is welcome to distribute, modify, translate, etc,
any of them.</p>
```

In the above example, files "`cc-by-sa.png`" and "`creativecommons.html`" are not processed as resources.

Tip

Option `externalresourcebase` may be used to specify an absolute or relative URI to be prepended to external resources having a relative URI. Example: `-p proc.externalresourcebase "../../samples/"`.

[15] Other than an input HTML page of course.

How to specify "*this is a resource too; copy it to the output directory*"?

By default, XMLmind Ebook Compiler considers that any file referenced by the ebook specification or an input HTML page using an *absolute URI is not a resource*. Example:

```
<p>All the above tutorials are licensed under the
<a href="https://creativecommons.org/creativecommons.html">
<img src="https://creativecommons.org/cc-by-sa.png"
alt="CC BY-SA"/>Creative Commons License</a>,
which means that everyone is welcome to distribute, modify, translate, etc,
any of them.</p>
```

In the above example, files "`https://creativecommons.org/creativecommons.html`" and "`https://creativecommons.org/cc-by-sa.png`" are not processed as resources.

If you want to specify that some files having absolute URIs are in fact resources and as such, should be processed by **ebookc**, please use

- value "`resource`" for standard attribute `rel` (HTML link elements have this attribute);
- proprietary attribute `data-resource` for elements like `img` which do not have attribute `rel`.

Example:

```
<p>All the above tutorials are licensed under the
<a href="https://creativecommons.org/creativecommons.html"
rel="resource">
<img src="https://creativecommons.org/cc-by-sa.png"
data-resource="" alt="CC BY-SA"/>Creative Commons License</a>,
which means that everyone is welcome to distribute, modify, translate, etc,
any of them.</p>
```

In the above example, files "`https://creativecommons.org/creativecommons.html`" and "`https://creativecommons.org/cc-by-sa.png`" are processed as resources.

Sub-resources

In the following example, files "`styles.css`", "`creativecommons.html`" and "`cc-by-sa.png`" are automatically processed as resources:

```
...
<head>
  ...
  <link href="css/styles.css" rel="stylesheet" type="text/css" />
</head>
...
<p>All the above tutorials are licensed under the
<a href="creativecommons.html"><img src="cc-by-sa.png"
alt="CC BY-SA"/>Creative Commons License</a>,
```

```
which means that everyone is welcome to distribute, modify, translate, etc,
any of them.</p>
```

Moreover, if file "`creativecommons.html`" contains XHTML —that is, can be parsed by XMLmind Ebook Compiler— its resources are processed too as if "`creativecommons.html`" were an input HTML page.

This is also the case for resource "`styles.css`". The resources found in a CSS stylesheet (e.g. file "`texture.png`" in "`background-image: url(images/texture.png);`" or file "`core_styles.css`" in "`@import url(lib/core_styles.css);`") are automatically detected and processed by XMLmind Ebook Compiler.

However, if she/he finds this clearer, the ebook author may also explicitly specify the sub-resources of CSS stylesheets using extra `link` elements in the `headcommon` of the ebook specification or in the `head` of an input HTML page. Example:

```
...
<head>
  ...
  <link href="css/images/" rel="resource" type="inode/directory" />
  <link href="css/styles.css" rel="stylesheet" type="text/css" />
</head>
...
<p>All the above tutorials are licensed under the
<a href="creativecommons.html"><img src="cc-by-sa.png"
alt="CC BY-SA"/>Creative Commons License</a>,
which means that everyone is welcome to distribute, modify, translate, etc,
any of them.</p>
```

Notice attribute `rel="resource"` which makes even clearer the purpose of this link. Also notice `type="inode/directory"` which is needed because "`css/images/`" is a folder and not a simple file.

4.3. Conditional processing

XMLmind Ebook Compiler can conditionally exclude some contents found in the ebook specification or in the input HTML pages. To put this feature into use, the ebook author must:

1. Specify one or more `data-*` attributes on the elements to be conditionally excluded. Examples: `data-edition="complete"`, `data-output-format="docx odt pdf rtf wml"`.

   These `data-*` attributes are often called *profiling attributes* because they are used to define several profiles for the same document.

   It's up to the ebook author to choose the names and allowed values for the profiling attributes.

   The ebook author may allow only a single value for a given profiling attribute. Example: attribute `data-edition` may contain only a single value, one of "`complete`" or "`abridged`".

   Or, on the contrary, the ebook author may allow a given profiling attribute to contain several values separated by space characters. Example: attribute `data-output-format` may contain one or more of "`docx`", "`epub`", "`frameset`", "`html`", "`odt`", "`pdf`", "`rtf`", "`webhelp`", "`wml`".

2. Pass one or more `profile.*` parameters to the `ebookc` command-line option. These `profile.*` parameters must match the chosen profiling attributes. Example: `-p profile.edition abridged -p profile.output-format pdf`.

   Note that unless you pass a `profile.*` parameter, the corresponding `data-*` attribute is not given any special meaning by XMLmind Ebook Compiler. For example, without `-p profile.output-format` *`VALUE`*, attribute `data-output-format` is considered to be just an ordinary attribute.

How some elements are conditionally excluded by XMLmind Ebook Compiler is best explained by an example:

Example 4-1. *Example of conditional processing*

```
<p>See YouTube demo:</p>

<p data-edition="complete" data-output-format="epub frameset html webhelp">
<iframe src="https://www.youtube.com/embed/6MgZBZ4XHzU"
height="360" width="640"></iframe></p>

<p data-edition="complete" data-output-format="docx odt pdf rtf wml">
<img src="images/YouTube_play_icon.svg" alt="..."/>
<a href="https://youtu.be/6MgZBZ4XHzU"
target="_blank">https://youtu.be/6MgZBZ4XHzU</a>.</p>
```

See YouTube demo:

iframe

 https://youtu.be/6MgZBZ4XHzU.

For an element to be excluded, suffice for a *single* profiling attribute to be “excluded”. A profiling attribute *`data-X`* is “excluded” if none of the values it contains matches a value contained in the `profile.`*`X`* parameter passed to `ebookc`.

For example, with `-p profile.edition complete -p profile.output-format pdf`, the embedded video

```
<p data-edition="complete" data-output-format="epub frameset html webhelp">
<iframe src="https://www.youtube.com/embed/6MgZBZ4XHzU"
height="360" width="640"></iframe></p>
```

is excluded because despite the fact that `data-edition="complete"` is “included”, `data-output-format="epub frameset html webhelp"` is “excluded”.

Other examples, if you pass `ebookc`

- no `profile.*` parameter at all, the above example will contain both the embedded video and the YouTube link to the video.
- `-p profile.edition abridged`, the above example will contain neither the embedded video nor the YouTube link to the video.

- `-p profile.edition complete`, the above example will contain both the embedded video and the YouTube link to the video.
- `-p profile.output-format epub`, the above example will contain just the embedded video.
- `-p profile.output-format pdf`, the above example will contain just the YouTube link to the video.
- `-p profile.edition abridged -p profile.output-format pdf`, the above example will contain neither the embedded video nor the YouTube link to the video.
- `-p profile.edition complete -p profile.output-format pdf`, the above example will contain just the YouTube link to the video.
- `-p profile.edition complete -p profile.output-format "epub pdf"`, the above example will contain both the embedded video and the YouTube link to the video.

4.4. Transclusion

XMLmind Ebook Compiler has good support for *transclusion*, that is the ability to include contents found in an input HTML page into another input HTML page. This feature is implemented using a standard mechanism called XInclude.

Example, "`page1.html`" contains paragraph having `id="notice"`:

```
<p id="notice" class="important">Interest rates are subject
to fluctuation without notice.</p>
```

Because this paragraph has an `id`, it's possible to include it in "`page2.html`":

```
<p>Paragraph found in page2.html.</p>

<xi:include href="page1.html" xpointer="notice"
  xmlns:xi="http://www.w3.org/2001/XInclude" />[16]

<p>Other paragraph found in page2.html.</p>
```

The corresponding output HTML page will then contain:

```
<p>Paragraph found in page2.html.</p>

<p id="notice" class="important">Interest rates are subject
to fluctuation without notice.</p>

<p>Other paragraph found in page2.html.</p>
```

[16] Creating `xi:include` elements by hand is tedious and error prone. It's strongly recommended to use an XInclude-enabled editor like XMLmind XML Editor to do that. With XMLmind XML Editor, creating an `xi:include` element is as easy as copying a *reference* to an element (**`Ctrl+Shift-C`**) from one page and then pasting it (**`Ctrl-V`**) into another page.

Note that transclusion works fine not only between two input HTML pages, but also:

- within the same input HTML page (see example below),
- between two ebook specifications,
- within the same ebook specification.

However transclusion does not work between an input HTML page and an ebook specification.

Example 4-2. *Transclusion works fine within the same input HTML page*

```
<p id="notice" class="important">Interest rates are subject
to fluctuation without notice.</p>

... ELSEWHERE in page1.html ...

<xi:include href="" xpointer="notice"
  xmlns:xi="http://www.w3.org/2001/XInclude" />
```

Notice `href=""` to refer to the same file.

Transclusion is most often used between the input HTML pages and a “utility HTML page” which is not an input HTML page but which contains useful “snippets”.

Example, excerpts from "`snippets.html`":

```
<ul>
  <li><span id="ebookc">XMLmind Ebook Compiler</span>.</li>

  <li><span id="xxe">XMLmind XML Editor</span>.</li>

  <li><a href="http://www.xmlmind.com/" id="xmlmind"
         target="_blank">XMLmind</a>.</li>
</ul>
```

Now, including snippets in an input HTML page:

```
<p><xi:include href="snippets.html" xpointer="ebookc"
xmlns:xi="http://www.w3.org/2001/XInclude" /> is free, open source software
developed by <xi:include href="snippets.html" xpointer="xmlmind"
xmlns:xi="http://www.w3.org/2001/XInclude" />.</p>

<p><xi:include href="snippets.html" xpointer="xxe"
xmlns:xi="http://www.w3.org/2001/XInclude" /> is a commercial product
developed by <xi:include href="snippets.html" xpointer="xmlmind"
xmlns:xi="http://www.w3.org/2001/XInclude" />.</p>
```

Part II. Reference

Table of Contents

5. Installation 37
6. Content of a source HTML page 39
6.1. Valid XHTML5 39
6.2. Headings 40
6.3. Examples 40
6.4. Equations 43
6.5. Admonitions 44
6.6. Footnotes 45
6.7. Cross-references 46
6.8. Index terms 48
7. Reference of ebook elements 51
7.1. Element `appendices` 51
7.2. Element `appendix` 52
7.3. Element `backmatter` 53
7.4. Element `body` 53
7.5. Element `book` 54
7.6. Element `chapter` 62
7.7. Element `content` 63
7.8. Element `frontmatter` 63
7.9. Element `head` 64
7.10. Element `headcommon` 66
7.11. Element `index` 67
7.12. Element `loe` 67
7.13. Element `lof` 68
7.14. Element `lot` 69
7.15. Element `lox` 70
7.16. Element `part` 71
7.17. Element `related` 72
7.18. Element `section` 73
7.19. Element `title` 74
7.20. Element `toc` 75
7.21. Common attributes 76
8. How it works 79
9. The `ebookc` command-line utility 81
10. XSLT stylesheets parameters 90
10.1. Parameters of the XSLT stylesheets used to convert an ebook specification to EPUB 90
10.2. Parameters of the XSLT stylesheets used to convert an ebook specification to Web Help 90
10.3. Parameters of the XSLT stylesheets used to convert an ebook specification to XSL-FO 96
10.3.1. Specifying a header or a footer 105

Chapter 5. Installation

System requirements

Make sure to have a Java™ 1.8+ runtime installed on your machine. To check this, please open a command window and type "`java -version`" followed by **`Enter`**. You should get something looking like this:

```
C:\> java -version
openjdk version "20.0.2" 2023-07-18
OpenJDK Runtime Environment (build 20.0.2+9-78)
OpenJDK 64-Bit Server VM (build 20.0.2+9-78, mixed mode)
```

Installation

Simply unzip `ebookc-`*`X_Y_Z`*`.zip` in any directory.

After that, you can run command-line utility `ebookc` by simply executing *`ebookc_install_dir`*`/bin/ebookc.bat` (*`ebookc_install_dir`*`/bin/ebookc` on the Mac and on Linux).

```
C:\> mkdir XMLmind
C:\> cd XMLmind
C:\XMLmind> unzip ebookc-1_9_0.zip
C:\XMLmind> dir ebookc-1_9_0
... <DIR> bin
... <DIR> doc
... <DIR> docsrc
... <DIR> LEGAL
...
C:\XMLmind> ebookc-1_9_0\bin\ebookc.bat
ebookc: ERROR: Usage: ebookc [option]* in_ebook_file out_file_or_directory
...
```

Contents of the installation directory

`bin/`, `bin/ebookc`, `bin/ebookc.bat`
: Contains the `ebookc` command-line utility. Use shell script `ebookc` on Linux and on the Mac. Use `ebookc.bat` on Windows.

`doc/`, `doc/index.html`
: Contains the documentation of XMLmind Ebook Compiler.

`docsrc/`, `docsrc/manual.xml`
: Contains the documentation of XMLmind Ebook Compiler in **ebookc** format. File `docsrc/manual.ebook` contains an ebook specification. You may want to use this sample ebook specification to experiment with the `ebookc` command-line utility.

`LEGAL/`, `LEGAL.txt`
: Contains legal information about XMLmind Ebook Compiler and about third-party components used in XMLmind Ebook Compiler.

`lib/*.jar`
: All the Java™ class libraries needed to run XMLmind Ebook Compiler. For example, `lib/ebookc.jar` contain the code of XMLmind Ebook Compiler.

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

`schema/`
: Contains a W3C XML schema and a Schematron which may be used to check an ebook specification for validity. File `schema/catalog.xml` contains an XML catalog which points to these schemas.

`src/`, `src/build.xml`
: Contains the Java™ source code of XMLmind Ebook Compiler. `src/build.xml` is an ant build file which allows to rebuild `lib/ebookc.jar`.

`whc_template/`
: Contains the template directory of XMLmind Web Help Compiler.

`xsl/`
: Contains the XSLT 2.0 stylesheets used to convert ebook specifications to a variety of formats.

Chapter 6. Content of a source HTML page

Table of Contents

6.1. Valid XHTML5 39
6.2. Headings 40
6.3. Examples 40
6.4. Equations 43
6.5. Admonitions 44
6.6. Footnotes 45
6.7. Cross-references 46
6.8. Index terms 48

6.1. Valid XHTML5

Your source HTML pages must contain valid[17] XHTML ("plain HTML" cannot be parsed by **ebookc**) and preferably *valid XHTML5*, because **ebookc** anyway generates XHTML5 markup.

The `html` root element must have 1 `head` and 1 `body` child elements. The `head` child element must have 1 `title` child element.

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <meta charset="UTF-8" />
    <title>...</title>
  </head>
  <body>
  ...
  </body>
</html>
```

Tip

If you plan to use XMLmind XML Editor as your ebook authoring tool, do not forget to add attribute `class="role-ebook-page"` to the `html` root element of your source HTML pages. XMLmind XHTML Editor detects this attribute and this will cause the editor to replace its stock "**XHTML**" menus and toolbars by extended "**XHTML**" menus and toolbars.

[17] Note that the validity of the source HTML pages is currently not checked by **ebookc**. It's only the validity of the ebook specification which is checked against W3C XML Schema *`ebookc_install_dir`*`/schema/ebook.xsd`.

6.2. Headings

You may use headings (`h1`, `h2`, `h3`, etc) normally, without worrying about the role as a book division (chapter, section, etc) that will be given to your input HTML page.

By default, `book` attribute `adjustuserheadings="true"` specifies that the levels of your headings are to be automatically adjusted to make them consistent with the level of the title of the book division.

Example, input HTML page contains:

```
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <meta charset="UTF-8" />
    <title>Troubleshooting</title>
  </head>
  <body>
  <h1>Symptoms</h1>
  ...
  <h2>Intermittent symptoms</h2>
  ...
  <h1>Most common causes</h1>
  ...
  </body>
</htm>
```

The above input HTML is referenced as a subsection of a chapter in the book. Therefore the title of the subsection is represented by an `h3` element. The output HTML page containing the subsection then looks like:

```
<section class="role-section2">
  <h3 class="role-section2-title">Troubleshooting</h3>
  <h4>Symptoms</h4>
  ...
  <h5>Intermittent symptoms</h5>
  ...
  <h4>Most common causes</h4>
  ...
```

If you want to prevent this from happening, please add attribute `adjustuserheadings="false"` to your root `book` element or add a `class` attribute to some or all of your headings. A heading having a `class` attribute is understood by XMLmind Ebook Compiler as being “not an ordinary heading which could be modified”.

6.3. Examples

An example is a `figure` element which has a `class` attribute containing "`role-example`". This kind of figure is listed in the "**List of Examples**" (that is, `book` element `lox`) only if it also has a `figcaption` child element. Example:

```
<figure class="role-example">
  <figcaption>"Hello World" program in the C language</figcaption>
```

```
  <pre>/* Hello World */
#include &lt;stdio.h&ght;

int main()
{
    printf("Hello World\n");
    return 0;
}</pre>
</figure>
```

is rendered as:

Example 6-1. *"Hello World" program in the C language*

```
/* Hello World */
#include <stdio.h>

int main()
{
    printf("Hello World\n");
    return 0;
}
```

Program listings

A program listing can have its lines automatically numbered and/or can feature syntax highlighting. This is done by adding "`role-listing-`*`NUMBER`*`-`*`LANGUAGE`*`-tab`*`WIDTH`*" to the `class` attribute of a `pre` element. Options *NUMBER*, *LANGUAGE*, `tab`*`WIDTH`*, may be specified in any order. Moreover some or all of these options may be omitted.

- *`NUMBER`*, a strictly positive integer, specifies the number of the first line of the program listing. This option may be omitted if you don't want automatic line numbering.
- *`LANGUAGE`*, one of (case-insensitive): "`bourne`" (or "`shell`" or "`sh`"), "`c`", "`cmake`" (or "`make`" or "`make-file`"), "`cpp`", "`csharp`", "`css21`" (or "`css`"), "`delphi`", "`ini`", "`java`", "`javascript`", "`lua`", "`m2`" (Modula 2), "`perl`", "`php`", "`python`", "`ruby`", "`sql1999`", "`sql2003`", "`sql92`" (or "`sql`"), "`tcl`", "`upc`" (Unified Parallel C), "`html`", "`xml`", specifies the programming language or markup language used in the program listing. This option may be omitted if you don't want syntax highlighting.
- `tab`*`WIDTH`* where *WIDTH* is a positive integer, specifies whether tab characters should be expanded to a number of space characters. *WIDTH* is the maximum number of space characters for an expanded tab character, hence this value specifies the location of “tab stops”. Example: `<pre class="role-listing-1-java-tab4">` means expand tabs to up to 4 space characters in this line-numbered Java listing. Other example: `<pre class="role-listing-tab0-shell">` means: do not replace tabs in this Bourne shell listing. When `tab`*`WIDTH`* is omitted, it is equivalent to having an implicit `tab8`.

Example 1 (in the following C program, lines are indented using tab characters):

```
<pre class="role-listing-1-c-tab4">/* Hello World */
#include &lt;stdio.h&ght;

int main()
{
        printf("Hello World\n");
        return 0;
}</pre>
```

is rendered as:

```c
/* Hello World */
#include <stdio.h>

int main()
{
    printf("Hello World\n");
    return 0;
}
```

Superfluous indentation is removed too

Note that in addition to replacing tab characters by a number of space characters, the `tab`*`WIDTH`* facility also removes the space characters which are common to the beginning of all text lines. That is, it removes the superfluous “indentation” in the program listing, if any.

Moreover, the `tab`*`WIDTH`* facility also removes the (useless) space characters found just before a newline character.

See example 2 below in which the indentation is automatically removed.

Example 2 (implicit `role-listing-1-tab8`; first line " `/tmp/`" starts with 4 space characters):

```
<pre class="role-listing-1">    /tmp/
    /usr/
        bin/
        lib/
        <b>local/</b>
                <b>bin/</b>
                <b>lib/</b>
                <b>src/</b>
        src/
    /var/
</pre>
```

is rendered as:

```
/tmp/
/usr/
    bin/
    lib/
    local/
            bin/
            lib/
            src/
    src/
/var/

```

6.4. Equations

An example is a `figure` element which has a `class` attribute containing "`role-equation`". This kind of figure is listed in the "**List of Equations**" (that is, `book` element `loe`) only if it also has a `figcaption` child element. Example:

```
<figure class="role-equation">
  <figcaption>Special relativity</figcaption>
  <div>
    <math display="block"
          xmlns="http://www.w3.org/1998/Math/MathML">
      <mrow>
        <mrow>
          <mi>t</mi>
          <mo>’</mo>
        </mrow>
        <mo>=</mo>
        <mrow>
          <mi>t</mi>
          <mo>&#x2062;</mo>
          <mfrac>
            <mn>1</mn>
            <msqrt>
              <mrow>
                <mn>1</mn>
                <mo>-</mo>
                <mfrac>
                  <msup>
                    <mi>v</mi>
                    <mn>2</mn>
                  </msup>
                  <msup>
                    <mi>c</mi>
```

```
                    <mn>2</mn>
                  </msup>
                </mfrac>
              </mrow>
            </msqrt>
          </mfrac>
        </mrow>
      </mrow>
    </math>
  <div>
</figure>
```

is rendered as:

Equation 6-1. *Special relativity*

Tip

Few web browsers natively support MathML, so it's recommended to add a link to the MathJax script to your input HTML pages containing equations[18]. This typically done as follows (this loads latest 3.x version of the MathJax `mml-chtml` component):

```
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <meta charset="UTF-8" />
    <title>...</title>

    <script async="async" id="MathJax-script"
            src="https://cdn.jsdelivr.net/npm/mathjax@3/es5/mml-chtml.js"
            type="text/javascript"></script>

  </head>
  ...
```

6.5. Admonitions

An admonition, that is, a warning, a tip, a notice, etc, is a `blockquote` element which has a `class` attribute containing "`role-`*`ADMONITION`*", where `role-`*`ADMONITION`* is one of the following class names:

[18] Even simpler, add the link to MathJax script to the `headcommon` element of your `book`.

Table 6-1. *Admonition classes*

Class name	Description
role-note	This is just a note.
role-attention	Please pay extra attention to this note.
role-caution	Care is required when proceeding.
role-danger	Important! Be aware of this before doing anything else.
role-fastpath	This note will speed you on your way.
role-important	This note is important.
role-notice	Indicates a potential situation which, if not avoided, might result in an undesirable result or state.
role-remember	Don't forget to do what this note says.
role-restriction	You can't do what this note says.
role-tip	This is a fine little tip.
role-trouble	Provides information about how to remedy a trouble situation.
role-warning	Indicates a potentially hazardous situation.

Example:

```
<blockquote class="role-important">
  <h4>How to check your oil</h4>
  <p>You need to check your car’s oil before any long trip
  to avoid major damage.</p>
  <p>The process for how to check your oil is simple and involves
  using the dip stick to see levels and test quality.</p>
</blockquote>
```

is rendered as:

How to check your oil

You need to check your car’s oil before any long trip to avoid major damage.

The process for how to check your oil is simple and involves using the dip stick to see levels and test quality.

6.6. Footnotes

Simple footnotes

This first and simplest form for a footnote is a `span` element which has a `class` attribute containing "`role-footnote`".

Example:

```
<p>Yoko<span class="role-footnote">Written with kanji <i>ko</i>, meaning
"child". The syllable <i>ko</i> is not generally found at the end of
masculine names.</span> is a Japanese feminine given name.</p>
```

is rendered as:

Yoko[19] is a Japanese feminine given name.

General footnotes

When you need a footnote to contain paragraphs, lists or tables or when you need to reuse the same footnote at different locations in your document, you'll have to use the second, more general, form for a footnote.

This second form is a `div` element which has a `class` attribute containing "`role-footnote`" and an id attribute.

Moreover, you'll also have to insert an `a` element at the location where you want the footnote marker to be displayed. This `a` element, which points to the footnote `div`, must have a `class` attribute containing "`role-footnote-ref`".

Example:

```
<p>Yoko<a class="role-footnote-ref" href="#ko"></a>is a Japanese
feminine given name.</p>

<div class="role-footnote" id="ko">Written with kanji <i>ko</i>,
meaning "child". The syllable <i>ko</i> is not generally found
at the end of masculine names.</div>
```

is rendered as:

Yoko[20] is a Japanese feminine given name.

6.7. Cross-references

No need to specify the text of a link when this link points to a book division (chapter, section, etc) or to a table, figure, example, or equation having a caption.

Example, the following empty links point respectively to section "*Admonitions*" and to table "*Admonition classes*" found in this section:

```
<p><a href="admonitions.html"></a> contains
<a href="admonitions.html#admonition_classes"></a>.</p>
```

are rendered as:

Section 6.5. Admonitions contains Table 6-1. Admonition classes.

[19] Written with kanji *ko*, meaning "child". The syllable *ko* is not generally found at the end of masculine names.

[20] Written with kanji *ko*, meaning "child". The syllable *ko* is not generally found at the end of masculine names.

The text which is automatically generated for these empty links may be configured using attribute `xreflabels` of element `book`.

Links specified using attribute `data-xml-id-ref`

It's also possible to create links using the `a` element and proprietary attribute `data-xml-id-ref` rather than (or in addition to) standard attribute `href`.

Attribute `data-xml-id-ref` must contain the value of the `xml:id` attribute of a book division found in the ebook specification. This allows the creation of links to locations that do not exist in the input HTML pages, but which will be created in the output HTML pages.

Example, `<a data-xml-id-ref="ch04"/>` points to the following chapter:

```
<chapter xml:id="ch04">
  <head><title>...</title></head>
  <section href="ch4/s1.html"/>
  <section href="ch4/s2.html"/>
</chapter>
```

In input HTML page "`ch4/s2.html`", you may refer to the first section of the chapter by writing `<a href="s1.html"/>`. But how to refer to the chapter itself? Notice that this chapter has no input HTML page to refer to.

The solution to this problem is to add proprietary attribute `data-xml-id-ref` to an `a` element. For the above example, it's `<a data-xml-id-ref="ch04"/>`.

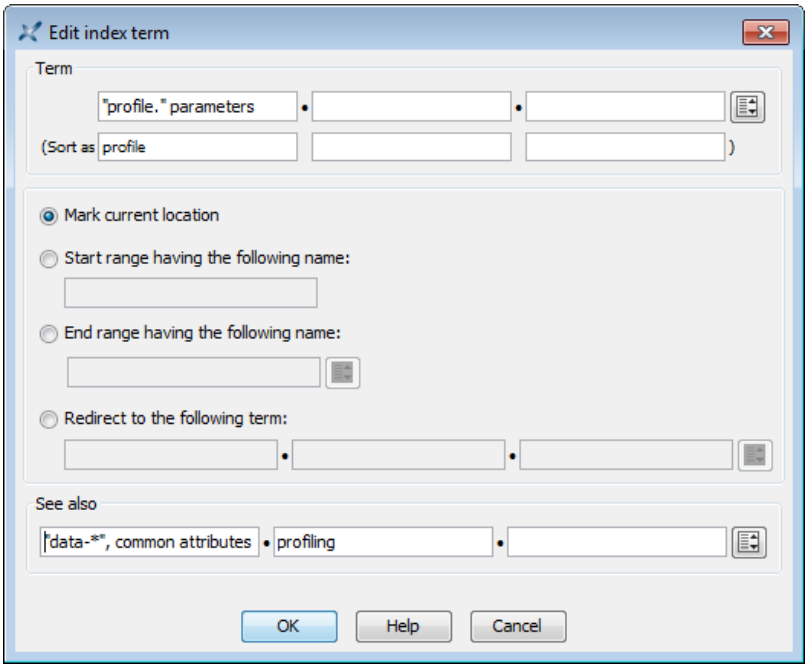
Note that writing `<a href="s1.html" data-xml-id-ref="ch04"/>` is an even better option because `href="s1.html"` is used as a fallback link target in case `xml:id="ch04"` is not defined in the ebook specification.

6.8. Index terms

Tip

Creating index terms by hand (other than copying an index term to paste it elsewhere) is tedious and error prone. It's strongly recommended to use the specialized dialog box of XMLmind XML Editor to do that.

Figure 6-1. *The "**Edit index term**" dialog box of XMLmind XML Editor*



An index term is represented by a `a` element having attribute `class="role-index-term"` containing text —the primary word or phrase in an index term— and possibly nested `span` elements having the following roles: "`role-term`", "`role-see`", "`role-see-also`".

```
index_term -> end_of_range | term

end_of_range -> <a class="role-index-term" data-end-range="range_name"/>

term -> <a class="role-index-term" term_attributes>term_content</a>

term_attributes -> [ data-sort-as="text" ]?
                   [ data-start-range="range_name" ]?

term_content -> rich_text term_childs

term_childs -> [ sub_term ]? | [ see ]* | [ see_also ]*
```

sub_term -> **<span class="role-term" *term_attributes*>***term_content*****

see -> *****see_content*****

see_also -> *****see_content*****

see_content -> *rich_text see_child*

see_child -> [*****rich_text see_child*****]?

In the above grammar:

- "Rich text" means the mix of text and phrase elements (`b`, `i`, `em`, etc) allowed in `a` and `span` elements.
- Though the grammar allows `<span class="role-term">` to be nested to an arbitrary depth, a `<a class="role-index-term">` may contain only up *two* nested `<span class="role-term">`, corresponding respectively to the secondary word and tertiary word of an index term. The same limit applies to `<span class="role-see">` and to `<span class="role-see-also">`.

Examples:

- Simplest index term containing just a phrase:

```
<a class="role-index-term">Dog, man's best friend</a>
```

- “Sort-as” example:

```
<a class="role-index-term"
   data-sort-as="percent">%</a>
```

- Index terms having primary, secondary and tertiary terms:

```
<a class="role-index-term"><b>Pet</b>
  <span class="role-term">Cat</span>
</a>
...
<a class="role-index-term"><b>Pet</b>
  <span class="role-term">Cat
    <span class="role-term">Siamese</span>
  </span>
</a>
...
<a class="role-index-term"><b>Pet</b>
  <span class="role-term">Cat
    <span class="role-term">Burmese</span>
  </span>
</a>
```

- Start of the "`dogs`" range:

```
<a class="role-index-term"><b>Pet</b>
  <span class="role-term" data-start-range="dogs">Dog</span>
</a>
```

- End of the above "`dogs`" range. The end of a range must be found *after* the corresponding start of range in the same input HTML page or in a different input HTML page:

```
<a class="role-index-term" data-end-range="dogs"></a>
```

  Notice that an end of range index term does not contain text nor any child element. It just has a "`data-end-range`" attribute.

- “See” example:

```
<a class="role-index-term"><i lang="la">Felis catus</i>
  <span class="role-see">Pet
    <span class="role-term">Cat</span>
  </span>
</a>
```

- “See also” example:

```
<a class="role-index-term"><i lang="la">Canis lupus</i>
  <span class="role-see-also">Dog, man's best friend</span>
  <span class="role-see-also">Pet
    <span class="role-term">Dog</span>
  </span>
</a>
```

Chapter 7. Reference of ebook elements

Table of Contents

7.1. Element appendices 51
7.2. Element appendix 52
7.3. Element backmatter 53
7.4. Element body 53
7.5. Element book 54
7.6. Element chapter 62
7.7. Element content 63
7.8. Element frontmatter 63
7.9. Element head 64
7.10. Element headcommon 66
7.11. Element index 67
7.12. Element loe 67
7.13. Element lof 68
7.14. Element lot 69
7.15. Element lox 70
7.16. Element part 71
7.17. Element related 72
7.18. Element section 73
7.19. Element title 74
7.20. Element toc 75
7.21. Common attributes 76

7.1. Element appendices

Specifies the group of appendices of the ebook.

Content model

```
(head? , body? , related* , appendix+)
```

Attributes

Name	Data type	Default value
href	anyURI min. length: 1	
pagename	token min. length: 1	
samepage	boolean	"false"
xml:base	anyURI	
xml:id	ID	
xml:lang	language or "" (the empty string)	.

Other attributes: XHTML5 global attributes, including any attribute having a name starting with "data-".

Parents

The following elements contain appendices: book.

Children

The following elements occur in appendices: appendix, body, head, related.

Example

```
<appendices pagename="Appendixes">
  <appendix href="pages/known_problems.html"/>
  <appendix href="pages/error_list.html">
    <section href="pages/report_error.html"/>
  </chapter>
</part>
```

7.2. Element appendix

Specifies an appendix of the ebook.

Content model

```
(head? , body? , related* , section*)
```

Attributes

Name	Data type	Default value
href	anyURI min. length: 1	
pagename	token min. length: 1	
samepage	boolean	"false"
xml:base	anyURI	
xml:id	ID	
xml:lang	language or "" (the empty string)	.

Other attributes: XHTML5 global attributes, including any attribute having a name starting with "data-".

Parents

The following elements contain appendix: appendices, book.

Children

The following elements occur in appendix: body, head, related, section.

Example

```
<appendices pagename="Appendixes">
  <appendix href="pages/known_problems.html"/>
  <appendix href="pages/error_list.html">
    <section href="pages/report_error.html"/>
  </chapter>
</part>
```

7.3. Element backmatter

Specifies the back matter of the ebook.

Content model

```
(toc | index | lot | lof | loe |
 lox | section)+
```

Attributes

Name	Data type	Default value
pagename	token min. length: 1	
samepage	boolean	"false"
xml:base	anyURI	
xml:id	ID	
xml:lang	language or "" (the empty string)	.

Other attributes: any attribute having a name starting with "data-".

Parents

The following elements contain backmatter: book.

Children

The following elements occur in backmatter: index, loe, lof, lot, lox, section, toc.

Example

```
<backmatter>
  <section href="glossary.html"/>
  <index/>
</backmatter>
```

7.4. Element body

Specifies the content of a book division (part, chapter, section, etc).

When the parent of body is element book then body specifies the content of the “title page” of the book.

It's possible for a book division to have both an `href` attribute and a `body` child element. In such case, the content "pulled" using the `href` attribute is inserted before the content specified by the `body` child element.

Content model

```
content+
```

Attributes

Name	Data type	Default value
`xml:base`	`anyURI`	
`xml:id`	`ID`	
`xml:lang`	`language` or "" (the empty string)	.

Other attributes: XHTML5 global attributes, including any attribute having a name starting with "`data-`".

Parents

The following elements contain `body`: `appendices`, `appendix`, `book`, `chapter`, `part`, `section`.

Children

The following elements occur in `body`: `content`.

Example

```
<chapter>
  <head>
    <title>Using Widget</title>
  </head>
  <body>
    <content href="using1.html"/>
    <content href="using2.html"/>
  </body>
</chapter>
```

7.5. Element book

Specifies a complete ebook.

Content model

```
(headcommon? ,
 head? ,
 body? ,
 related* ,
 frontmatter? ,
 ((part+ , appendices?) |
```

```
  (chapter+ , appendix*)) ,
 backmatter?)
```

Attributes

Name	Data type	Default value
adjustuserheadings	boolean	"true"
appendicestocdepth	nonNegativeInteger	"0"
appendixnumber	normalizedString	"%A"
appendixtocdepth	nonNegativeInteger	"0"
booklistlabels	none | all | (part | chapter | appendix | section | figure | table | example | equation)+	"none"
chapternumber	normalizedString	"%1"
chaptertocdepth	nonNegativeInteger	"0"
equationnumber	normalizedString	"%n-%1"
examplenumber	normalizedString	"%n-%1"
figurenumber	normalizedString	"%n-%1"
footnotenumber	normalizedString	"[%1]"
headoverridedefault	boolean	"false"
href	anyURI min. length: 1	
includebasestylesheet	boolean	"false"
labelseparator	normalizedString	". "
pagename	token min. length: 1	
partnumber	normalizedString	"%I"
parttocdepth	nonNegativeInteger	"0"
preventlonelyheading	boolean	"true"
section1number	normalizedString	"%n.%1"
section2number	normalizedString	"%n.%1"
section3number	normalizedString	"%n.%1"
section4number	normalizedString	"%n.%1"
section5number	normalizedString	"%n.%1"
section6number	normalizedString	"%n.%1"
section7number	normalizedString	"%n.%1"
section8number	normalizedString	"%n.%1"

Name	Data type	Default value
section9number	normalizedString	"%n.%1"
tablenumber	normalizedString	"%n-%1."
titlelabels	none | all | (part | chapter | appendix | section | figure | table | example | equation)+	"part chapter appendix figure table example equation"
tocdepth	positiveInteger	"10"
xml:base	anyURI	
xml:id	ID	
xml:lang	language or "" (the empty string)	.
xreflabels	none | all | (part | part-number | chapter | chapter-number | appendix | appendix-number | section | section-number | figure | figure-number | table | table-number | example | example-number | equation | equation-number)+	"all"

Other attributes: XHTML5 global attributes, including any attribute having a name starting with "data-".

adjustuserheadings
: If set to true, change the level of user-specified headings (h1, h2, h3, etc) to be consistent with the level of automatically generated headings. If set to false, do not change any user-specified headings. Example:

```
<chapter href="ch01.html" pagename="first_chapter">
  <section href="s01.html" pagename="first_section">
    <section href="s01_01.html" pagename="nested_section">
    ...
```

where input HTML file "s01_01.html" starts with a user-specified h1.

With adjustuserheadings="false", output HTML file "nested_section.html" contains:

```
<section class="role-section2>
  <h3 class="role-section2-title">Title of the section copied
                                  from "s01_01.html"<h3>
    <h1>User-specified heading found in "s01_01.html"</h1>
     ...
```

With adjustuserheadings="true", output HTML file "nested_section.html" contains:

```
<section class="role-section2>
  <h3 class="role-section2-title">Title of the section copied
                                  from "s01_01.html"<h3>
```

```
4     <h4>User-specified heading found in "s01_01.html"</h4>
5     ...
```

Tip

Note that `adjustuserheadings="true"` has no effect on headings having a `class` attribute. A heading having a user-specified `class` attribute is understood by XMLmind Ebook Compiler as being “not an ordinary heading which could be modified”.

`appendicestocdepth`
: If set to an integer larger than 0, instructs **ebookc** to automatically generate a Table of Contents (**TOC**) having specified depth at the beginning of the appendices division of the book.

`appendixnumber`
: Specifies the format of the number automatically added to the title of an appendix. See Number format.

`appendixtocdepth`
: If set to an integer larger than 0, instructs ebookc to automatically generate a Table of Contents (**TOC**) having specified depth at the beginning of each appendix of the book.

`booklistlabels`
: Specifies the kind of numbered book divisions (`part`, `chapter`, `appendix`, `section`) and numbered figure objects (figure, table, equation, example) for which to add *labels*. This option applies to book list entries (`toc`, `lof`, `lot`, `loe`, `lox`).

What is a *label*?

A *label* is a localized message containing the type of the book division or figure object. For example, with `chapternumber="%1"`, `labelseparator=") "`, `booklistlabels="none"`, a **TOC** entry for a chapter looks like: "**1) Introduction**". With `booklistlabels="chapter"` (or `booklistlabels="all"`), this **TOC** entry looks like: "**Chapter 1) Introduction**".

Note that labels are added only to *numbered* book divisions or figure objects. For example, with `chapternumber="%1"`, `booklistlabels=""`, a **TOC** entry for a chapter will look like: "**Introduction**".

`chapternumber`
: Specifies the format of the number automatically added to the title of a chapter. See Number format.

`chaptertocdepth`
: If set to an integer larger than 0, instructs **ebookc** to automatically generate a Table of Contents (**TOC**) having specified depth at the beginning of each chapter of the book.

`equationnumber`
: Specifies the format of the number automatically added to the caption of an equation. See Number format.

`examplenumber`
: Specifies the format of the number automatically added to the caption of an example. See Number format.

`figurenumber`
: Specifies the format of the number automatically added to the caption of an figure. See Number format.

`footnotenumber`
: Specifies the format of the number automatically added to footnotes (`<span class="role-footnote">` or `<div class="role-footnote">`) and footnote callouts (`<a class="role-footnote-ref">`).

`includebasestylesheet`
: If set to "`true`", include *`ebookc_install_dir`*`/xsl/common/resources/base.css` in all the output HTML pages.

  Using the `base.css` stock CSS stylesheet is the simplest, easiest, mean to create a nicely formatted book. More information about this attribute in *Leveraging `base.css`, the stock CSS stylesheet*.

Tip

When `includebasestylesheet="true"`, `base.css` is included *before* the other CSS stylesheets referenced in the `headcommon` (if any).

If you want to control where `base.css` is included, do not set `includebasestylesheet` to "`true`", instead add a `headcommon` similar to the one in the following example:

```
<headcommon>
  <html:link href="corporate_styles.css" rel="stylesheet"
             type="text/css"/>
  <html:link href="ebookc-home:xsl/common/resources/base.css"
             rel="stylesheet resource" type="text/css"/>
</headcommon>
```

The "`ebookc-home:`" prefix works because stock XML catalog *`ebookc_install_dir`*`/schema/catalog.xml` contains:
`<rewriteURI uriStartString="ebookc-home:" rewritePrefix="../"/>`.

`headoverridedefault`
: Specifies the default value of attribute `override` of element `head`.

`labelseparator`
: Specifies the string which is appended to the *label* automatically generated at the beginning of the title of a book division (`part`, `chapter`, `appendix`, `section`) or figure object (figure, table, equation, example). Example: with `labelseparator=") "`, the output HTML element generated for the following chapter is:

```
<chapter href="ch01.html">
```

is:

```
1 <section class="role-chapter>
2   <h1 class="role-chapter-title">
```

```
    <span class="role-label">Chapter
      <span class="role-number">1</span>) </span>
    Title of the chapter copied from "ch01.html"<h1>
```

`partnumber`
: Specifies the format of the number automatically added to the title of a part of the book. See Number format.

`parttocdepth`
: If set to an integer larger than 0, instructs **ebookc** to automatically generate a Table of Contents (**TOC**) having specified depth at the beginning of each part of the book.

`preventlonelyheading`
: If set to `true`, prevent an output HTML page from containing only a title. Example:

```
<chapter pagename="chapter1">
  <head>
    <title>First chapter</title>
  </head>
  <section href="s01.html"/>
  ...
```

With `preventlonelyheading="false"`, output HTML page "*`output_directory`*`/chapter1.html`" contains just the title of the chapter "**First chapter**", which may be surprising for the reader of the book.

With `preventlonelyheading="true"`, output HTML page "*`output_directory`*`/chapter1.html`" contains the title of the chapter "**First chapter**" and also the content of input HTML page "`s01.html`"[21].

`section1number`
: Specifies the format of the number automatically added to the title of a top level section. See Number format.

`section2number`
: Specifies the format of the number automatically added to the title of a section having a nesting level equal to 2 (subsection of a top level section). See Number format.

`section3number`
: Specifies the format of the number automatically added to the title of a section having a nesting level equal to 3. See Number format.

`section4number`
: Specifies the format of the number automatically added to the title of a section having a nesting level equal to 4. See Number format.

`section5number`
: Specifies the format of the number automatically added to the title of a section having a nesting level equal to 5. See Number format.

`section6number`
: Specifies the format of the number automatically added to the title of a section having a nesting level equal to 6. See Number format.

[21] As if attribute `samepage="true"` were automatically added to the `section` element.

`section7number`
: Specifies the format of the number automatically added to the title of a section having a nesting level equal to 7. See Number format.

`section8number`
: Specifies the format of the number automatically added to the title of a section having a nesting level equal to 8. See Number format.

`section9number`
: Specifies the format of the number automatically added to the title of a section having a nesting level equal to 9. See Number format.

`tablenumber`
: Specifies the format of the number automatically added to the caption of an table. See Number format.

`titlelabels`
: Specifies the kind of numbered book divisions (`part`, `chapter`, `appendix`, `section`) and numbered figure objects (figure, table, equation, example) for which to add *labels*. This option applies to titles or captions.

  For example, with `chapternumber="%1"`, `labelseparator=") "`, `titlelabels="none"`, the title of a chapter looks like: "**1) Introduction**". With `titlelabels="chapter"` (or `titlelabels="all"`), this title looks like: "**Chapter 1) Introduction**".

`tocdepth`
: Specifies the depth of the main Table of Contents (**TOC**) (see `toc` element).

`xml:lang`
: Specifies the main language of the book. This language is used to automatically generate some titles (e.g. "**Table of Contents**", "**List of Figures**") and also to sort index entries.

Tip

Unlike `lang`, which is a XHTML5 global attribute, `xml:lang` is *not* copied to the output HTML element corresponding to the `book` element.

However, explicitly setting attribute `xml:lang` on the `book` element is a convenient way to ensure that all the output HTML pages have a `lang` attribute.

`xreflabels`
: Specifies the kind of numbered book divisions (`part`, `chapter`, `appendix`, `section`) and numbered figure objects (figure, table, equation, example) for which to add *labels*. This option applies to automatically generated link text.

  For example, with `chapternumber="%1"`, `labelseparator=") "`, `xreflabels="none"`, the text automatically generated for empty link to chapter `<a href="intro.html"/>` looks like: "**1) Introduction**". With `xreflabels="chapter"` (or `xreflabels="all"`), this text looks like: "**Chapter 1) Introduction**".

  With `xreflabels="chapter-number"`, this text looks like: "**Chapter 1**", that is, no chapter title, just the label without any label separator. Note that this `"-number"` suffix is supported only by `xreflabels`.

Number format

`%1`
: Decimal numbers, beginning with 1.

`%a`
: Lowercase ASCII letters (a, b, c, ... z).

`%A`
: Uppercase ASCII letters (A, B, C, ... Z).

`%i`
: Lowercase roman numerals (i, ii, iii, iv, v, etc).

`%I`
: Uppercase roman numerals (I, II, III, IV, V, etc).

`%n`
: Number of parent element. Example: prepend the number of the `chapter` parent to the number of a top level `section` element: "`%n.%1`".

  In the case of a figure, table, equation or example, `%n` is the number of the ancestor `chapter` or `appendix` element.

An empty string may be used to specify that the book division or figure object is not numbered.

Restriction

- There is no automatic numbering inside `frontmatter` and `backmatter` elements.

  There is no automatic numbering *directly inside* `part` and `appendices` elements.

  That's why section numbers like "`%n.%1`" and figure numbers like "`%n-%1`" work in all cases.

- Sections having a nesting level greater than 9 cannot be numbered.

- An ebook specification can only have a single `appendices` division. That's why an `appendices` division cannot be numbered (i.e. no `appendicesnumber` attribute).

Children

The following elements occur in `book`: `appendices`, `appendix`, `backmatter`, `body`, `chapter`, `frontmatter`, `head`, `headcommon`, `part`, `related`.

Example

```
<book appendixtocdepth="100" chaptertocdepth="100"
      section7number="" section8number="" section9number=""
      labelseparator=" "
      booklistlabels="chapter appendix"
      xreflabels="chapter appendix section
                  figure-number table-number equation-number example-number"
      xml:lang="en-US"
      xmlns="http://www.xmlmind.com/schema/ebook"
      xmlns:html="http://www.w3.org/1999/xhtml">
  <head>
    <title>Widget User Guide</title>
    <html:meta content="John Smith" name="author"/>
    <html:meta content="2017-08-25" name="dc.date"/>
  </head>
  ...
</book>
```

7.6. Element `chapter`

Specifies a chapter of the ebook.

Content model

```
(head? , body? , related* , section*)
```

Attributes

Name	Data type	Default value
`href`	`anyURI` min. length: 1	
`pagename`	`token` min. length: 1	
`samepage`	`boolean`	`"false"`
`xml:base`	`anyURI`	
`xml:id`	`ID`	
`xml:lang`	`language` or "" (the empty string)	.

Other attributes: XHTML5 global attributes, including any attribute having a name starting with "`data-`".

Parents

The following elements contain chapter: book, part.

Children

The following elements occur in chapter: body, head, related, section.

Example

```
<part>
  <chapter href="pages/install.html">
    <section href="pages/requirements.html" samepage="true"/>
  </chapter>
  <chapter href="pages/quick_start.html"/>
</part>
```

7.7. Element `content`

Instructs XMLmind Ebook Compiler to copy to the output HTML page all the elements found in the `html:body` of the input HTML page pointed to by the `href` attribute.

Content model

EMPTY

Attributes

Name	Data type	Default value
`href`	`anyURI` min. length: 1	REQUIRED
`xml:base`	`anyURI`	
`xml:id`	`ID`	
`xml:lang`	`language` or "" (the empty string)	.

Other attributes: any attribute having a name starting with "`data-`".

Parents

The following elements contain `content`: `body`.

Example

```
<chapter>
  <head>
    <title>Using Widget</title>
  </head>
  <body>
    <content href="using1.html"/>
    <content href="using2.html"/>
  </body>
</chapter>
```

7.8. Element `frontmatter`

Specifies the front matter of the ebook.

Content model

```
(toc | index | lot | lof | loe |
 lox | section)+
```

Attributes

Name	Data type	Default value
pagename	token min. length: 1	
samepage	boolean	"false"
xml:base	anyURI	
xml:id	ID	
xml:lang	language or "" (the empty string)	.

Other attributes: any attribute having a name starting with "data-".

Parents

The following elements contain frontmatter: book.

Children

The following elements occur in frontmatter: index, loe, lof, lot, lox, section, toc.

Example

```
<frontmatter>
  <toc/>
  <lof/>
  <lot/>
  <lox/>
  <loe/>
  <section href="intro.html"/>
</frontmatter>
```

7.9. Element head

Specifies the content of the html:head element of an output HTML page.

By default, this html:head element is simply a copy of the html:head element found in the content “pulled” using the href attribute of a book division. But when a head child element of a book division is specified,

1. Its title child element is used to specify the html:title of the output HTML page.
2. All its other child elements and also all its XHTML5 global attributes are copied to the html:head of the output HTML page.

Content model

```
(title? ,
 (html:base | html:link | html:meta | html:script | html:style |
  html:template)*)
```

Attributes

Name	Data type	Default value
override	boolean	Specified by attribute headoverridedefault of element book.
xml:base	anyURI	
xml:id	ID	
xml:lang	language or "" (the empty string)	.

Other attributes: XHTML5 global attributes, including any attribute having a name starting with "data-".

override
: When set to true, the child elements and XHTML5 global attributes found in the head element *completely replace* the child elements and XHTML5 global attributes found in the html:head element of an input HTML page.

  When set to false, the child elements and XHTML5 global attributes found in the head element are merged with the child elements and XHTML5 global attributes found in the html:head element of an input HTML page.

Parents

The following elements contain head: appendices, appendix, book, chapter, part, section.

Children

The following elements occur in head: html:base, html:link, html:meta, html:script, html:style, html:template, title.

Example

Element head is most often used to give a “rich” title to a book division.

```
<appendix href="ssh_key.html">
  <head>
    <title>Generating Your <html:b>SSH</html:b> Public Key</title>

    <html:style>
.error {
    font-weight: bold;
    font-style: italic;
    color: #800000;
}
    </html:style>
```

```
  </head>
</appendix>
```

See also

- Section 7.10. Element `headcommon`

7.10. Element `headcommon`

Specifies some *common content* for the `html:head` elements of all the output HTML pages.

Note that the XHTML5 global attributes found on element `headcommon` are also copied to the `html:head` elements of all the output HTML pages.

Content model

```
(html:base | html:link | html:meta | html:script | html:style |
 html:template)*
```

Attributes

Name	Data type	Default value
`xml:base`	`anyURI`	
`xml:id`	`ID`	
`xml:lang`	`language` or "" (the empty string)	.

Other attributes: XHTML5 global attributes, including any attribute having a name starting with "`data-`".

Parents

The following elements contain `headcommon`: `book`.

Children

The following elements occur in `headcommon`: `html:base`, `html:link`, `html:meta`, `html:script`, `html:style`, `html:template`.

Example

Element `headcommon` is typically used to give a common CSS stylesheet to all the output HTML pages.

```
<book>
  <headcommon>
    <html:link href="../resources/styles.css" rel="stylesheet"
               type="text/css"/>
  </headcommon>
  ...
</book>
```

See also

- Section 7.9. Element `head`

7.11. Element `index`

Instructs XMLmind Ebook Compiler to automatically generate an index.

Remember

- The language used to automatically sort generated index entries is taken from the `xml:lang` attribute of the `book` element.
- An index term is a `a` element without an `href` attribute having `class` attribute containing "`role-index-term`". See Section 6.8. Index terms.

Content model

EMPTY

Attributes

Name	Data type	Default value
`pagename`	`token` min. length: 1	
`samepage`	`boolean`	"`false`"
`xml:base`	`anyURI`	
`xml:id`	`ID`	
`xml:lang`	`language` or "" (the empty string)	.

Other attributes: any attribute having a name starting with "`data-`".

Parents

The following elements contain `toc`: `backmatter`, `frontmatter`.

Example

```
<backmatter>
  <section href="glossary.html"/>
  <index/>
</backmatter>
```

7.12. Element `loe`

Instructs XMLmind Ebook Compiler to automatically generate a List of Equations (**LOE**).

Remember

An equation listed in the **LOE** is a `html:figure` element having a `html:figcaption` and a `class` attribute containing "`role-equation`". See Section 6.4. Equations.

Content model

EMPTY

Attributes

Name	Data type	Default value
`pagename`	`token` min. length: 1	
`samepage`	`boolean`	"`false`"
`xml:base`	`anyURI`	
`xml:id`	`ID`	
`xml:lang`	`language` or "" (the empty string)	.

Other attributes: any attribute having a name starting with "`data-`".

Parents

The following elements contain `loe`: `backmatter`, `frontmatter`.

Example

```
<frontmatter>
  <toc/>
  <lof/>
  <lot/>
  <lox/>
  <loe/>
  <section href="intro.html"/>
</frontmatter>
```

See also

- Section 7.14. Element `lot`
- Section 7.13. Element `lof`
- Section 7.15. Element `lox`

7.13. Element `lof`

Instructs XMLmind Ebook Compiler to automatically generate a List of Figures (**LOF**).

Remember

A plain figure listed in the **LOF** is a `html:figure` having a `html:figcaption` and no `class` attribute or a `class` attribute not containing "`role-equation`" or "`role-example`".

Content model

EMPTY

Attributes

Name	Data type	Default value
`pagename`	`token` min. length: 1	
`samepage`	`boolean`	"`false`"
`xml:base`	`anyURI`	
`xml:id`	`ID`	
`xml:lang`	`language` or "" (the empty string)	.

Other attributes: any attribute having a name starting with "`data-`".

Parents

The following elements contain `lof`: `backmatter`, `frontmatter`.

Example

```
<frontmatter>
  <toc/>
  <lof/>
  <lot/>
  <lox/>
  <loe/>
  <section href="intro.html"/>
</frontmatter>
```

See also

- Section 7.14. Element `lot`
- Section 7.12. Element `loe`
- Section 7.15. Element `lox`

7.14. Element `lot`

Instructs XMLmind Ebook Compiler to automatically generate a List of Tables (**LOT**).

Remember

A table listed in the **LOT** is a `html:table` having a `html:caption`.

Content model

EMPTY

Attributes

Name	Data type	Default value
`pagename`	`token` min. length: 1	
`samepage`	`boolean`	`"false"`
`xml:base`	`anyURI`	
`xml:id`	`ID`	
`xml:lang`	`language` or "" (the empty string)	.

Other attributes: any attribute having a name starting with "`data-`".

Parents

The following elements contain `lot`: `backmatter`, `frontmatter`.

Example

```
<frontmatter>
  <toc/>
  <lof/>
  <lot/>
  <lox/>
  <loe/>
  <section href="intro.html"/>
</frontmatter>
```

See also

- Section 7.13. Element `lof`
- Section 7.12. Element `loe`
- Section 7.15. Element `lox`

7.15. Element `lox`

Instructs XMLmind Ebook Compiler to automatically generate a List of Examples (**LOX**).

Remember

An example listed in the **LOX** is a `html:figure` element having a `html:figcaption` and a `class` attribute containing "`role-example`". See Section 6.3. Examples.

Content model

EMPTY

Attributes

Name	Data type	Default value
`pagename`	`token` min. length: 1	
`samepage`	`boolean`	"`false`"
`xml:base`	`anyURI`	
`xml:id`	`ID`	
`xml:lang`	`language` or "" (the empty string)	.

Other attributes: any attribute having a name starting with "`data-`".

Parents

The following elements contain `lox`: `backmatter`, `frontmatter`.

Example

```
<frontmatter>
  <toc/>
  <lof/>
  <lot/>
  <lox/>
  <loe/>
  <section href="intro.html"/>
</frontmatter>
```

See also

- Section 7.14. Element `lot`
- Section 7.13. Element `lof`
- Section 7.12. Element `loe`

7.16. Element `part`

Specifies a part —a group of chapters— of the ebook.

Content model

```
(head? , body? , related* , chapter+)
```

Attributes

Name	Data type	Default value
href	anyURI min. length: 1	
pagename	token min. length: 1	
samepage	boolean	"false"
xml:base	anyURI	
xml:id	ID	
xml:lang	language or "" (the empty string)	.

Other attributes: XHTML5 global attributes, including any attribute having a name starting with "data-".

Parents

The following elements contain part: book.

Children

The following elements occur in part: body, chapter, head, related.

Example

```
<part>
  <chapter href="pages/install.html">
    <section href="pages/requirements.html" samepage="true"/>
  </chapter>
  <chapter href="pages/quick_start.html"/>
</part>
```

7.17. Element related

Instructs XMLmind Ebook Compiler to generate a list of links.

The targets of these links are the book divisions (part, chapter, section, etc) having an xml:id attribute referenced in the ids attribute of the related element.

The default title of this list of links is "**Related information**". A different title (e.g. "**See also**") may be specified in attribute relation.

Content model

EMPTY

Attributes

Name	Data type	Default value
ids	IDREFS	REQUIRED
relation	token min. length: 1	
xml:base	anyURI	
xml:id	ID	
xml:lang	language or "" (the empty string)	.

ids
: Specifies the IDs of the related book divisions (part, chapter, section, etc). Redundant IDs found in this list are ignored.

relation
: Specifies the title of the automatically generated list of links. By default, it's "**Related information**" translated to the language of the parent element of the automatically generated list of links.

Parents

The following elements contain related: appendices, appendix, book, chapter, part, section.

Example

```
<chapter href="ch01.html" xml:id="ch01">
  <related ids="ch01 ch02 ch03"/>
</chapter>
<chapter href="ch02.html" xml:id="ch02">
  <related ids="ch01 ch02 ch03"/>
</chapter>
...
```

7.18. Element section

Specifies a section of the ebook.

Content model

```
(head? , body? , related* , section*)
```

Attributes

Name	Data type	Default value
href	anyURI min. length: 1	
pagename	token min. length: 1	
samepage	boolean	"false"

Name	Data type	Default value
xml:base	anyURI	
xml:id	ID	
xml:lang	language or "" (the empty string)	.

Other attributes: XHTML5 global attributes, including any attribute having a name starting with "data-".

Parents

The following elements contain section: appendix, backmatter, chapter, frontmatter, section.

Children

The following elements occur in section: body, head, related, section.

Example

```
<part>
  <chapter href="pages/install.html">
    <section href="pages/requirements.html" samepage="true"/>
  </chapter>
  <chapter href="pages/quick_start.html"/>
</part>
```

7.19. Element title

Specifies the “rich” title of a book division (part, chapter, section, etc).

Content model

Element title can contain text and the same XHTML5 child elements as an html:p element (that is,*phrasing content*: html:b, html:img, etc) in any order and in any number.

Attributes

Name	Data type	Default value
xml:base	anyURI	
xml:id	ID	
xml:lang	language or "" (the empty string)	.

Other attributes: XHTML5 global attributes, including any attribute having a name starting with "data-".

Parents

The following elements contain title: head.

Children

The following elements occur in title: the same XHTML5 child elements as an html:p element.

Example

```
<appendix href="ssh_key.html">
  <head>
    <title>Generating Your <html:b>SSH</html:b> Public Key</title>
  </head>
</appendix>
```

7.20. Element `toc`

Instructs XMLmind Ebook Compiler to automatically generate a Table of Contents (**TOC**).

Content model

EMPTY

Attributes

Name	Data type	Default value
`pagename`	`token` min. length: 1	
`samepage`	`boolean`	`"false"`
`xml:base`	`anyURI`	
`xml:id`	`ID`	
`xml:lang`	`language` or "" (the empty string)	.

Other attributes: any attribute having a name starting with "`data-`".

Parents

The following elements contain `toc`: `backmatter`, `frontmatter`.

Example

```
<frontmatter>
  <toc/>
  <lof/>
  <lot/>
  <lox/>
  <loe/>
  <section href="intro.html"/>
</frontmatter>
```

7.21. Common attributes

`href`

Specifies the location of an input HTML file. This file must contain valid XHTML5 (more information in Section 6.1.  Valid XHTML5). The specified URL may not have a fragment identifier (e.g. something like `href="ch09.html#conclusion"` is not supported).

`pagename`

Specifies the base name without any extension of an output HTML file. By default, this name is the same as the name of the corresponding input HTML file. Example:

```
<chapter href="intro.html" pagename="introduction"/>
```

By default, without attribute `pagename`, the page generated for the above chapter would be *`output_directory`*`/intro.html`.

After setting `pagename` to "`introduction`", the page generated for the above chapter is *`output_directory`*`/introduction.html`.

`samepage`

Specifies that the book division (e.g. a section) is to be generated in the same output HTML file as its parent book division (e.g. a chapter). By default, all book divisions are generated by **ebookc** in their own HTML files. Example:

```
1 <chapter href="ch1.html">
2   <section href="ch1/s1.html" samepage="true"/>
3   <section href="ch1/s2.html"/>
4 </chapter>
```

Attribute `samepage="true"` instructs **ebookc** to generate the content of the chapter and the content of the first section in the same HTML file. The second section having an implied `samepage="false"` is created in its own HTML file.

Note that something like:

```
1 <chapter href="ch1.html">
2   <section href="ch1/s1.html"/>
3   <section href="ch1/s2.html" samepage="true"/>
4 </chapter>
```

is an error because there is no way for **ebookc** to generate two sibling sections in the same output HTML file.

`xml:base`

Specifies a base URL which used to resolve the relative URLs found in the ebook specification.

`xml:lang`

Ignored for any element other than `book`, for which it specifies the main language of the book.

`xml:id`

Specifies the unique ID of an element of the ebook specification. Specifying an `xml:id` attribute is useful in the following cases:

- It is required for a book division to be referenced in a `related` element. Example:

```
<chapter href="ch1.html" xml:id="ch01">
  <related ids="ch01 ch02 ch03" xml:id="rel1"/>
</chapter>
```

- It allows the inclusion of ebook elements using XInclude. In the preceding example, `related` element "`rel1`" is defined in first chapter. In the following example, a copy of `related` element "`rel1`" is included in the second chapter:

```
<chapter href="ch2.html" xml:id="ch02">
  <xi:include href="" xpointer="rel1" set-xml-id=""
              xmlns:xi="http://www.w3.org/2001/XInclude"/>
</chapter>
```

- It may be used to control the IDs generated in the output HTML pages. Example:

```
<chapter href="ch3.html" xml:id="going_further">
  <section href="ch3/s1.html" xml:id="requirements" samepage="true"/>
</chapter>
```

  - The `html` element of the output page containing the chapter will have `id="going_further"`. All the elements "pulled" from "`ch3.html`" will have their IDs prefixed with "`going_further__`".
  - The `section` element containing the section will have `id="requirements"`. All the elements "pulled" from "`ch3/s1.html`" will have their IDs prefixed with "`requirements__`".

- Referencing the value of an `xml:id` attribute in proprietary attribute `data-xml-id-ref` may be used to create links to locations that do not exist in the input HTML pages, but which will be created in the output HTML pages. Example:

```
<chapter xml:id="ch04">
  <head><title>...</title></head>
  <section href="ch4/s1.html"/>
  <section href="ch4/s2.html"/>
</chapter>
```

  In input HTML page "`ch4/s2.html`", you may refer to the first section of the chapter by writing `<a href="s1.html"/>`. But how to refer to the chapter itself? Notice that this chapter has no input HTML page to refer to.

  The solution to this problem is to add proprietary attribute `data-xml-id-ref` to an `a` element. For the above example, it's `<a data-xml-id-ref="ch04"/>`.

  Note that writing `<a href="s1.html" data-xml-id-ref="ch04"/>` is an even better option because `href="s1.html"` is used as a fallback link target in case `xml:id="ch04"` is not defined in the ebook specification.

Any XHTML5 global attribute, including any attribute having a name starting with "`data-`"
These attributes (e.g. `class`, `dir`, `lang`, `onclick`, `style`) are copied to the output HTML element corresponding to the book division. Example: the output HTML element corresponding to the following appendix:

```
<appendix href="a2.html" samepage="true" class="disclaimer" lang="fr-FR"/>
```

is:

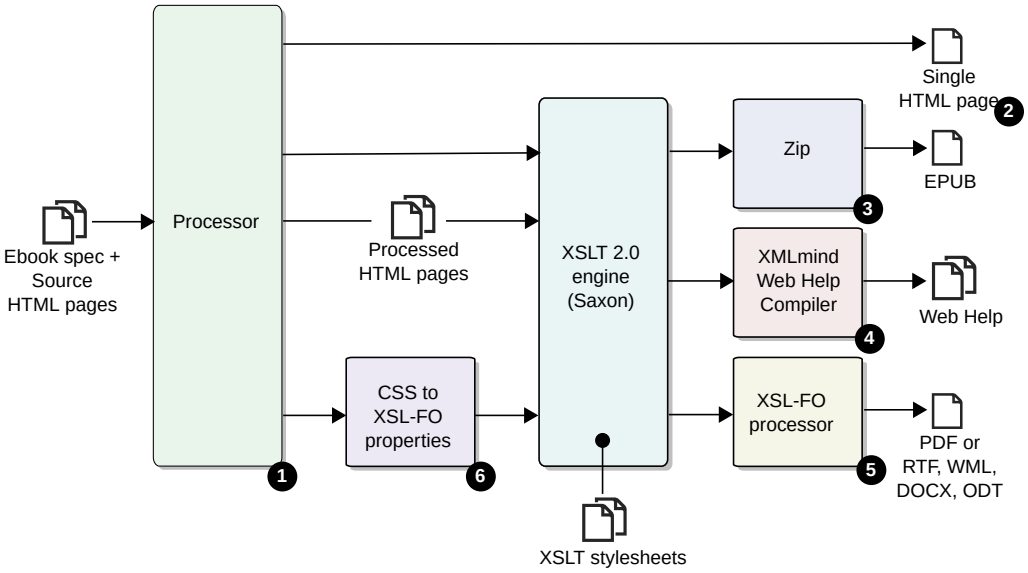
```
<html:section class="role-appendix disclaimer" lang="fr-FR"/>
```


WARNING

Specifying an `id` attribute for a book division is likely to cause broken links in the output HTML files.

Chapter 8. How it works

Figure 8-1. *XMLmind Ebook Compiler components*



1. The main component of XMLmind Ebook Compiler
2. Single HTML page output
3. EPUB ouput
4. Web Help ouput
5. XSL-FO based output
6. CSS styles to XSL-FO properties

1. The *Processor* is the main component of XMLmind Ebook Compiler. It processes an ebook specification referencing a number of valid XHTML5 pages. It generates processed valid XHTML5 pages and generally also, a subdirectory (called "`_res/`" by default) containing all the resources referenced by the processed pages.

   Whatever the file layout of the input HTML pages and their resources, all the files and directories are always created in a single output directory, which makes this output directory self-contained.

   In addition to the processed pages, the Processor automatically creates an HTML page (called "`_toc_frame.html`" by default) containing a table of contents and the manifest of all the resources found in the resource directory (in the form of `<link href="`*`XXX`*`" rel="resource" type="`*`YYY`*`"/>` elements).

   The Processor also automatically creates an HTML page (called "`_frameset.html`" by default) containing a `frameset`. The only purpose of this `frameset` is to be able to quickly navigate the output of the Processor when testing and debugging.

2. Generating a single HTML page out of an ebook specification does not involve any further processing steps. The Processor is simply instructed to generate a single page and files "`_toc_frame.html`" and "`_frameset.html`" are discarded.

3. Generating an EPUB file requires transforming "`_toc_frame.html`" by the means of the `xsl/epub/epub.xsl` stylesheet and then archiving[22] the contents of the output directory.

4. Generating a Web Help requires transforming "`_toc_frame.html`" by the means of the `xsl/webhelp/webhelp.xsl` stylesheet and then processing the contents of the output directory using XMLmind Web Help Compiler.

5. Generating PDF, DOCX, ODT, etc, requires first generating an intermediate format called XSL-FO. This is done by the means of the `xsl/fo/fo.xsl` stylesheet. After that, it's up to an XSL-FO processor — Apache FOP, RenderX XEP or Antenna House Formatter for the PostScript and PDF formats, XMLmind XSL-FO Converter for the RTF, WML, DOCX and ODT formats— to create the output file.

6. The CSS styles specified in the ebook specification and in the source HTML pages are also used when generating output formats based on XSL-FO. However for this to work, these CSS styles need to be translated to directly usable XSL-FO properties (see `apply-css-styles`) and stored in processing-instructions (`<?css-styles?>`) prior to be transformed by the `xsl/fo/fo.xsl` stylesheet. This preparatory step is implemented by the "CSS to XSL-FO properties" component depicted in the above figure.

[22] An EPUB is a zip archive.

Chapter 9. The ebookc command-line utility

Command-line usage

```
ebookc [option]* in_ebook_file out_file_or_directory
```

Converts specified ebook input file and saves the result of the conversion to specified output file or directory.

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

Output formats `webhelp`, `html` and `frameset` require *output_file_or_directory* to be a directory. Other output formats require *output_file_or_directory* to be a file.

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

Example: convert `userguide.ebook` to *Web Help*:

```
C:\docsrc> ebookc -f webhelp userguide.ebook out\wh
```

Example: convert `userguide.ebook` to PDF using RenderX XEP:

```
C:\docsrc> ebookc -xep C:\xep\xep.bat userguide.ebook out\userguide.pdf
```

Commonly used command-line options

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

`-p` *param_name param_value*
`-param` *param_name param_value*
: Specifies a conversion parameter, generally an XSLT stylesheet parameter.

"profile." parameters

A *param_name* starting with "`profile.`" specifies a profiling attribute. Example: `-p profile.data-output-format html` or more simply `-p profile.output-format html` (the "`data-`" attribute name prefix being implied). See Section 4.3. Conditional processing.

"load.*page_loader_name*." parameters

A *param_name* starting with "`load.`*`page_loader_name`*`.`" specifies an option which is passed to the alternate page loader called *page_loader_name*. For example, `-p load.markdown.autolink true` turns on the **autolink** extension in the Markdown loader. See Supported Markdown extensions.

"proc." parameters

A *param_name* starting with "`proc.`" specifies a low-level option which is passed to the first pass of `ebookc`. This first pass, called the *Processor*, compiles the input ebook specification to multi-page XHTML5 with a `frameset` and a "TOC frame"[23], see Chapter 8. How it works. Example: `-p proc.resourcedirname resources`.

[23] In other words, when using option `-f frameset`, `ebookc` stops after its first pass.

Setting these low-level options “by hand” is almost never needed, it's best not to fiddle with these.

Table 9-1. *Low-level processor options*

Option	Value	Description
`copiablelinks`	List of values separated by whitespace. Allowed values are: `'part'`, `'chapter'`, `'section'`, `'figure'`, `'table'`, `'heading'`, `'all'` (equivalent to `'chapter section figure table'`; `'part'` and `'heading'` *not* included in this list). Default value: `''` (do not add copiable links).	Adds a *copiable link* to the heading or caption child elements of specified “formal elements”. • If the “formal element” is numbered (e.g. has a "**Chapter 1.**" automatically generated label), then the automatically generated label is converted to a link. This link points to the formal element (e.g. a link to the `section` having a chapter role). • Otherwise (e.g. a `table` which is not numbered), a link containing the section symbol, "**§**", is added to the heading or caption. This link points to the formal element (e.g. a link to the `table`). This automatically generated link to a formal element is intended to be copied using the "**Copy Link**" entry found in the contextual menu of all web browsers in order to be shared with others. For example, send this link by email. **Restriction** • For this facility to work, the formal element must have an `id` attribute, whether specified by the author or automatically generated by `ebookc`. • It does not make sense to use this parameter when generating EPUB or any XSL-FO based output format (PDF, RTF, etc). Use it only when generating HTML or Web Help.

Option	Value	Description
`debug`	`true` | `false` Default: `false`.	Print low-level debugging info.
`externalresource-base`	Absolute or relative URI ending with `'/'`. Default: `''` (no base).	Specifies an absolute or relative URI to be prepended to external resources having a relative URI.
`framesetfilename`	File basename without any extension. Default: "`_frameset`".	Specifies the name of the `frameset` file generated by first pass.
`htmlcharset`	A valid charset. Default: "`UTF-8`".	Specifies which charset to use for the generated HTML files.
`htmlextension`	File extension (without a leading period). Default: "`html`".	Specifies which file extension to use for the generated HTML files.
`ignoreresources`	`true` | `false` Default: `false`.	If set to `true`, do not process resources. That is, treat all resources as if they were external resources.
`indexfilename`	File basename without any extension. Default: none.	Specifies that the index is to be generated in a separate HTML file. This option specifies the name of this separate file. Setting this option generally also requires setting `suppressindex` to `true`. Ignored unless the ebook as specified by the user actually contains an `<index/>` descendant.
`pagenavigation`	`none` | `header` | `footer` | `both` Default: `none`.	Specifies whether page navigation headers and/or footers are to be added to the output HTML pages. The page navigation headers and footers are styled using CSS stylesheet `pageNavigation.css` found in *`ebookc_install_dir`*`/xsl/common/resources/`.
`reservedfilenames`	One or more file basenames (without any extension) separated by newline characters.	Do not generate HTML files having any of the specified names.

Option	Value	Description
	Default: none.	
`resourcedirname`	File basename without any extension. Default: "`_res`".	Specifies the name of the directory where all the resources (e.g. image files, CSS files) referenced in the output HTML pages are stored.
`resourcedirnamefor`	URL or file path.	Same as `resourcedirname` except that the name of the resource directory is computed out of the option value. For example, sets the name of the resource directory to "`my doc_files`" when passed "`file:/tmp/my%20doc.epub`" or "`C:\temp\my doc.epub`".
`singlepage`	`true` | `false` Default: `false`.	Generate a single HTML page.
`suppressindex`	`true` | `false` Default: `false`.	Suppress `<index/>` from the ebook specification before generating the output HTML pages. Setting `suppressindex` to `true` is generally needed when `indexfilename` is also specified.
`suppresstoc`	`true` | `false` Default: `false`.	Suppress <toc/> from the ebook specification before generating the output HTML pages.
`tocframefilename`	File basename without any extension. Default: "`_toc_frame`".	Specifies the name of the “TOC frame” file generated by first pass.
`validate`	`true` | `false` Default: `true` when invoked by the `ebookc` command-line utility, `false` otherwise.	Validate the ebook specification against the W3C XML schema found in *`ebookc_install_dir`*`/schema/ebook.xsd`.

`-t` *XSLT_stylesheet_URL_or_file*
`-xslt` *XSLT_stylesheet_URL_or_file*
Use the specified custom XSLT stylesheet rather than the stock one.

`-f` html1 | html | webhelp | epub | ps | pdf | rtf | odt | wml | docx | fo | frameset
`-format` html1 | html | webhelp | epub | ps | pdf | rtf | odt | wml | docx | fo | frameset
Explicitly specifies the output format. By default, the output format is determined using the extension of *output_file_or_directory*.

Table 9-2. *Output formats*

Output format	Description
`html1`	Single XHTML5 page. Automatically detected filename extensions are: "`html`", "`htm`", "`xhtml`", "`xhtm`" or "`xht`".
`html`	Multiple XHTML5 pages.
`webhelp`	*Web Help*
`epub`	EPUB 3
`ps`	PostScript[24]
`pdf`	PDF[24]
`rtf`	RTF (can be opened in Word 2000+)[25]
`wml`	WordprocessingML (can be opened in Word 2003+)[25]
`docx`	Office Open XML (`.docx`, can be opened in Word 2007+)[25]
`odt`	OpenOffice (`.odt`, can be opened in OpenOffice/LibreOffice 2+)[25]
`fo`	XSL-FO. Mainly used for debugging and testing purposes.
`frameset`	Multi-page XHTML5 with a `frameset` and a “TOC frame”. Mainly used for debugging and testing purposes.

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

Example:

```
$ ebookc -v -o go.options go.ebook go.epub
```

If `go.options` contains:

[24] Requires an XSL-FO processor such as Apache FOP, RenderX XEP or Antenna House Formatter to have been installed and registered with XMLmind Ebook Compiler (for example, using option `-foconverter`).

[25] Requires XMLmind XSL-FO Converter to have been installed and registered with XMLmind Ebook Compiler (using option `-xfc`).

```
-p epub-identifier urn:isbn:0451450523
-p cover-image /home/john/artwork/playing_go.png
```

then this is equivalent to running:

```
$ ebookc -v -p epub-identifier urn:isbn:0451450523 \
    -p cover-image /home/john/artwork/playing_go.png \
    go.ebook go.epub
```

`-v`
`-vv`
`-vvv`
Turn verbosity on. More Vs means more verbose.

Command-line options used to configure ebookc

`-fop` *executable_file*
Specifies the location of the `fop` shell script (`fop.bat` on Windows).

Shorthand for:

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

`-xep` *executable_file*
Specifies the location of the `xep` shell script (`xep.bat` on Windows).

Shorthand for:

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

`-ahf` *executable_file*
Specifies the location of `AHFCmd.exe` (`run.sh` on platforms other than Windows).

Shorthand for:

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

`-xfc` *executable_file*
Specifies the location of the `fo2rtf` shell script (`fo2rtf.bat` on Windows).

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

Shorthand for:

```
-foconverter XFC rtf "fo2rtf_executable_file" "%I" "%O"
-foconverter XFC wml "fo2wml_executable_file" "%I" "%O"
```

```
-foconverter XFC docx "fo2docx_executable_file" "%I" "%O"
-foconverter XFC odf "fo2odt_executable_file" "%I" "%O"
```


WARNING

XMLmind XSL-FO Converter Evaluation Edition (download page) generates output containing *random duplicate letters*. This makes this edition useless for any purpose other than evaluating XMLmind XSL-FO Converter. Of course, this does not happen with XMLmind XSL-FO Converter Professional Edition!

`-foconverter` *processor_name target_format command*
: Register specified XSL-FO converter with ebookc, a lower-level alternative to using `-xep`, `-fop`, `-ahf` or `-xfc`. Example:

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

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

This low-level option may be used for example to specify a configuration file for Apache FOP:

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

Command-line options used to debug ebookc

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

`-errout`
: Output all messages, including errors and warnings, to `stdout`.

`-ignoreoptionsfile`
: Do not load the `ebookc.options` options file. See below The `ebookc.options` file.

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

`-keepforesources` `true`|`yes`|`on`|`1` | `false`|`no`|`off`|`0`
: When generating XSL-FO, PDF, RTF, DOCX, etc, do not delete the generated resource directory.
By default, `-keepfo` implies `-keepforesources true`.

`-version`
: Print version number and exit.

The `ebookc.options` file

It is also possible to specify command-line options in the `ebookc.options` options file. The content of this plain text file, encoded in the native encoding of the platform (e.g. Windows-1252 on a Western Windows PC), is auto-

matically loaded by `ebookc` each time this command is executed. The content of this file, command-line options separated by whitespace, is *prepended* to the options specified in the command-line.

Example: If `ebookc.options` contains:

```
-v -xep C:\xep\xep.bat
```

Running:

```
~/docsrc$ ebookc userguide.ebook out\userguide.pdf
```

is equivalent to running:

```
~/docsrc$ ebookc -v -xep C:\xep\xep.bat userguide.ebook out\userguide.pdf
```

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

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

The `ebookc.options` options file is mainly useful to configure `ebookc` once for all by specifying values for the `-fop`, `-xep`, `-xfc`, `-ahf` options.

Example:

```
-v
-xep E:\opt\xep\xep.bat
-fop E:\opt\fop-2.9\fop\fop.bat
-xfc "E:\opt\xfc_eval_java-6_4_1\bin\fo2rtf.bat"
```

Remember

- Relative filenames found in this file are relative to the current working directory, and not to the `ebookc.options` options file. Therefore it is recommended to always specify absolute filenames.
- No comments (e.g. lines starting with '#') are allowed in `ebookc.options`. Options must be separated by whitespace.
- In the above example, FOP is declared *after* XEP. This implies that it is FOP and not XEP, which will be used by ebookc to generate PDF and PostScript®.
- An XSL-FO processor tend to consume a lot of memory. If the ebook compilation fails with an out-of-memory error, you need to edit the `xep` (`xep.bat`), `fop` (`fop.bat`), `fo2`*`xxx`* (`fo2`*`xxx`*`.bat`) scripts in order to increase the maximum amount of memory that the Java™ runtime may allocate. This is done by using the `-Xmx` option of the Java™ command-line. Example: "`java ... -Xmx512m ...`".
- Starting from Java™ 1.6.0_23, converting XML documents to PDF using RenderX XEP randomly fails with false XSL-FO errors (e.g. attribute "space-before" may not be empty). This problem seems specific to the 64-bit runtime.

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

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

Chapter 10. XSLT stylesheets parameters

Table of Contents

10.1. Parameters of the XSLT stylesheets used to convert an ebook specification to EPUB 90
10.2. Parameters of the XSLT stylesheets used to convert an ebook specification to Web Help 90
10.3. Parameters of the XSLT stylesheets used to convert an ebook specification to XSL-FO 96
10.3.1. Specifying a header or a footer 105

10.1. Parameters of the XSLT stylesheets used to convert an ebook specification to EPUB

Parameter	Value	Default Value	Description
`cover-image`	URI. If the URI is relative, it is relative to the current working directory of the user.	None.	Specifies an image file which is to be used as the cover page of the EPUB file. This image must be a PNG or JPEG image. Its size must not exceed 1000x1000 pixels. In theory, EPUB 3 also accepts SVG 1.1 cover images.
`epub-identifier`	String	Dynamically generated UUID URN.	A globally unique identifier for the generated EPUB document (typically the permanent URL of the EPUB document).
`epub2-compatible`	`'no'` | `'yes'`	`'yes'`	By default, the EPUB 3 files generated by **ebookc** are made compatible with EPUB 2 readers. Specify `'no'` if you don't need this compatibility.
`omit-toc-root`	`'no'` | `'yes'`	`'yes'`	Specify `'yes'` if you want the title of the book to be the root of the EPUB TOC.

10.2. Parameters of the XSLT stylesheets used to convert an ebook specification to Web Help

Note

Parameters starting with "`wh-`" are *pseudo-parameters*. They may or may not be passed to the XSLT stylesheets, but the important thing to remember is that they are also interpreted by **ebookc** itself. Consequently you cannot specify them in an XSLT stylesheet which customizes the stock ones.

Parameter	Value	Default Value	Description
`omit-toc-root`	`'no'` | `'yes'`	`'yes'`	Specify `'yes'` if you want the title of the book to be the root of the Web Help TOC.

Parameter	Value	Default Value	Description

`wh---`*`CSS_VAR_NAME`*	String. A valid CSS property value.	No default.	This kind of parameter may be used to override any of the default values of the CSS variables specified in any of the *`NN`*`theme.css` template files (all found in *`ebookc_install_dir`*`/whc_template/_wh/`).
			
			For example, the main NNtheme.css template file:
			
			`body {`
			`  ...`
			`  --navigation-width: 33%;`
			`  ...`
			`}`
			
			The `wh---navigation-width` CSS variable is used as follows in *`NN`*`common.css`, another CSS template file:
			
			`#wh-navigation {`
			`  ...`
			`  width: var(--navigation-width);`
			`  ...`
			`}`
			
			Therefore parameter `wh---navigation-width` may be used to give the navigation side of the generated Web Help a different initial width. Example: `-p wh---navigation-width "25%"`.
			
			More examples in "*XMLmind Web Help Compiler Manual, Getting started*".

`wh-collapse-toc`	`'no'` | `'yes'`	`'no'`	Specifies whether the Web Help TOC should be initially collapsed.
`wh-index-numbers`	`'no'` | `'yes'`	`'no'`	Specifies whether words looking like numbers are to be indexed. Examples of such number-like words: `3.14`, `3,14`, `3times4equals12`, `+1`, `-1.0`, `3px`, `1,2cm`, `100%`, `1.0E+6`, `1,000.00$`.
`wh-inherit-font-and-colors`	`'no'` | `'yes'`	`'yes'`	When `wh-inherit-font-and-colors` is set to `'no'`, the navigation pane of the generated Web Help uses fonts and colors of its own, which will generally differ from those used for the content of the Web Help. Setting `wh-inherit-font-and-colors` to `'yes'` lets you use for the navigation pane the same fonts and colors as those used for the content of the Web Help.

Parameter	Value	Default Value	Description

			So basically this parameter is a shorthand for:
			`-p wh---navigation-font-family inherit¬`
			`-p wh---navigation-font-size inherit¬`
			`-p wh---navigation-color inherit¬`
			`-p wh---navigation-background-color inherit`
			See above wh---*CSS_VAR_NAME* parameters.

`wh-jquery`	Relative or absolute URI. A relative URI is relative to the URI of a page of the Web Help.	Absolute URI of the corresponding file found on the Google CDN.	Specifies the location of the JavaScript file containing jQuery. Example: `https://code.jquery.com/¬ jquery-3.7.1.slim.min.js`. Specifying an "`https:`" URL is recommended when the generated Web Help is stored on an HTTPS server.
`wh-local-jquery`	`'no'` | `'yes'`	`'no'`	Specifies whether all jQuery files should be copied to `_wh/jquery/`, where `_wh/` is the directory containing the other Web Help files. By default, the jQuery files are accessed from the Web (typically from a **CDN**). Note that this parameter is applied after jQuery has been possibly customized using parameter `wh-jquery`. For example, "`-p wh-jquery https://code.jquery.com/jquery-3.7.1.min.js`" copies a file downloaded from `https://code.jquery.com/ to _wh/jquery/`.
`wh-layout`	The name of a layout.	`'classic'`	Selects a layout for the generated Web Help. For now, only 3 layouts are supported: `'classic'`, `'simple'` and `'corporate'`.
`wh-responsive-ui`	`'no'` | `'yes'`	`'yes'`	Specifies whether the generated Web Help should be "responsive", that is, whether it should adapt its layout to the size of the screen.
`wh-ui-language`	"`browser`" or "`document`" or a language code conforming RFC 3066. Examples: `de`, `fr-CA`.	`'browser'`	Specifies which language should be used for the messages (tab labels, button tool tips, etc) of the generated Web Help. Default value "`browser`" means that this language is the one used by the Web browser for its own messages.

Parameter	Value	Default Value	Description
			This language may often be specified in the user preferences of the Web browser. Value "`document`" means that the language of the document should be used. A language code such as `en`, `en-US`, es, `es-AR`, etc, may be used to explicitly specify which language should be used.
`wh-use-stemming`	`'no'` | `'yes'`	`'yes'`	Specifies whether *stemming* should be used to implement the search facility. By default, stemming is used whenever possible, that is, 1. when the main language of the XHTML pages to be compiled can be determined; 2. when this main language is one of: Danish, Dutch, English, Finnish, French, German, Hungarian, Italian, Norwegian, Portuguese, Russian, Spanish, Swedish, Romanian, Turkish. The main language of the document is specified by the `@xml:lang` attribute found on the root element of the ebook specification being compiled; otherwise, it is assumed to be "`en`".

`wh-user-css`	Filename or absolute URI of a CSS file. A relative filename is relative to the current working directory.	None.	Specifies the user's CSS stylesheet which is to be added to an XHTML page decorated by the compiler.
			
			This file is copied to *`output_directory`*`/_wh/user/`.
			
			Sample user's CSS `wh_resources/header_footer.css` as used in the following example:
			
			`-p wh-user-header¬`
			`wh_resources/header.html`
			`-p wh-user-footer¬`
			`wh_resources/footer.html`
			`-p wh-user-css¬`
			`wh_resources/header_footer.css`
			`-p wh-user-resources¬`
			`wh_resources/header_footer_files`

Parameter	Value	Default Value	Description
`wh-user-footer`	Filename or absolute URI of an XHTML file. A relative filename is relative to the current working directory.	None.	Specifies the user's footer which is to be added to each page of the Web Help. The content of the `body` element of `user-footer` is inserted as is in the `<div id="wh-footer">` found in each page of the Web Help. Same remark as for parameter `user-header` about the resources referenced by a user's footer. Sample user's footer `wh_resources/footer.html` as used in the following example: `-p wh-user-header¬ wh_resources/header.html -p wh-user-footer¬ wh_resources/footer.html -p wh-user-css¬ wh_resources/header_footer.css -p wh-user-resources¬ wh_resources/header_footer_files` More examples in "*XMLmind Web Help Compiler Manual, Getting started*".
`wh-user-header`	Filename or absolute URI of an XHTML file. A relative filename is relative to the current working directory.	None.	Specifies the user's header which is to be added to each page of the Web Help. The content of the `body` element of `user-header` is inserted as is in the `<div id="wh-header">` found in each page of the Web Help. If a user's header references resources (e.g. image files), then these resources must either be referenced using absolute URLs or these resources must be found in a user's resource directory and parameter `user-resources` must be specified. Example: • The user's resource directory is called `header_footer_files/` and contains `header_footer_files/logo200x100.png`. • ebookc is passed parameters `-p user-resources` *`PATH_TO`*`/header_footer_files` and `-p user-header` *`PATH_TO`*`/header.html`. • `header.html` looks like this:

Parameter	Value	Default Value	Description
			<pre><html> ... <body> ... ... </body> </html></pre> Notice the path used to reference `logo200x100.png`. Sample user's header `wh_resources/header.html` as used in the following example: <pre>-p wh-user-header¬ wh_resources/header.html -p wh-user-footer¬ wh_resources/footer.html -p wh-user-css¬ wh_resources/header_footer.css -p wh-user-resources¬ wh_resources/header_footer_files</pre> More examples in "*XMLmind Web Help Compiler Manual, Getting started*".
`wh-user-resources`	Filename or absolute "`file:`" URI of a *directory*. URI schemes other than "`file`" (e.g. "`http`") are not supported for this parameter. A relative filename is relative to the current working directory.	None.	Specifies a user's resource directory which is to be recursively copied to *`output_directory`*`/_wh/user/`. This directory typically contains image files referenced by the user's header, footer or CSS stylesheet. Sample user's resource directory `wh_resources/header_footer_files/` as used in the following example: <pre>-p wh-user-header¬ wh_resources/header.html -p wh-user-footer¬ wh_resources/footer.html -p wh-user-css¬ wh_resources/header_footer.css</pre>

Parameter	Value	Default Value	Description
			`-p wh-user-resources¬ wh_resources/header_footer_files` More examples in "*XMLmind Web Help Compiler Manual, Getting started*".

System parameters

Note

Such system parameters are not intended to be specified by the end-user. Such system parameters are documented here only because the end-user may see them referenced in some dialog boxes, in some configuration files or in the source code of the XSLT stylesheets.

Parameter	Value	Default Value	Description
`whc-index-base-name`	URL base-name	`'__tmp__index.whc_ndx'`	Basename of the Index XML input file of XMLmind Web Help Compiler.
`whc-toc-base-name`	URL base-name	`'__tmp__toc.whc_toc'`	Basename of the TOC XML input file of XMLmind Web Help Compiler.

10.3. Parameters of the XSLT stylesheets used to convert an ebook specification to XSL-FO

Parameter	Value	Default Value	Description
`apply-css-styles`	`'no'` | `'yes'`	`'yes'`	Specifies whether CSS styles specified in XHTML `style` attributes, `style` and `link` elements also apply to the XSL-FO file. Depending on the context, the following CSS properties are converted to their equivalent XSL-FO attributes. The corresponding shorthand CSS properties are supported too. *Any other CSS property is ignored.* Generated content (`:before`, `:after`) is ignored too. • `margin-top`, `margin-right`, `margin-bottom`, `margin-left`. • `padding-top`, `padding-right`, `padding-bottom`, `padding-left`.

Parameter	Value	Default Value	Description
			• `border-top-style`, `border-right-style`, `border-bottom-style`, `border-left-style`. • `border-top-width`, `border-right-width`, `border-bottom-width`, `border-left-width`. • `border-top-color`, `border-right-color`, `border-bottom-color`, `border-left-color`. • `background-color`, `background-image`, `background-repeat`, `background-position`. • `color`. • `font-family`, `font-style`, `font-weight`, `font-size`. • `text-decoration`. • `text-align`. • `text-indent`. • `vertical-align`. • `line-height`. • `list-style-type`, `list-style-position`, `list-style-image`. • `width`, `height`. • `caption-side`. • `border-spacing`. *Note that styles specified this way supersede all the other ways to specify the presentation in the output file*, that is, parameters like `base-font-size` or the presentation attributes (`xsl:attribute-set`) specified in the XSLT stylesheets that generate the XSL-FO file.
`base-font-size`	Length in `pt`	`'10pt'`	The size of the font used for most body elements (paragraphs, tables, lists, etc). All the other font sizes are computed relatively to this font size.
`base-line-height`	A valid line height	`'10'`	The line height used for most body elements (paragraphs, tables, lists, etc). All the line heights are computed relatively to this line height.
`external-href-after`	String	`']'`	Appended after the external URL referenced by an `a` element. Ignored unless `show-external-links='yes'`.
`external-href-before`	String	`' ['`	Separates the text of an `a` element from the external URL it points to. Ignored unless `show-external-links='yes'`.
`font-family`	One or more font families	`'serif'`	The font family used by default for all elements.

Parameter	Value	Default Value	Description
	separated by commas		

`footer-center`	A mix of text and variables.	See next column.	Specifies the contents of the central part of a page footer. See Section 10.3.1. Specifying a header or a footer.
			
			Default value:
			
			`two-sides even:: {{chapter-title}};;`
			`two-sides body odd:: {{section1-title}};;`
			`one-side:: {{chapter-title}}`

`footer-center-width`	String representing an integer larger than or equal to 1.	`'6'`	Specifies the proportional width of the central part of a page footer. See Section 10.3.1. Specifying a header or a footer.
`footer-left`	A mix of text and variables.	See next column.	Specifies the contents of the left part of a page footer. See Section 10.3.1. Specifying a header or a footer. Default value: `two-sides even:: {{page-number}}`
`footer-left-width`	String representing an integer larger than or equal to 1.	`'2'`	Specifies the proportional width of the left part of a page footer. See Section 10.3.1. Specifying a header or a footer.

`footer-right`	A mix of text and variables.	See next column.	Specifies the contents of the right part of a page footer. See Section 10.3.1. Specifying a header or a footer.
			
			Default value:
			
			`two-sides first||odd:: {{page-number}};`
			`one-side:: {{page-number}}`

`footer-right-width`	String representing an integer larger than or equal to 1.	`'2'`	Specifies the proportional width of the right part of a page footer. See Section 10.3.1. Specifying a header or a footer.
`footer-separator`	`'no'|'yes'`	`'yes'`	Specifies whether an horizontal rule should be drawn above the page footer. See Section 10.3.1. Specifying a header or a footer.
`header-center`	A mix of text and variables.	`'{{document-title}}'`	Specifies the contents of the central part of a page header. See Section 10.3.1. Specifying a header or a footer.

Parameter	Value	Default Value	Description
`header-center-width`	String representing an integer larger than or equal to 1.	`'6'`	Specifies the proportional width of the central part of a page header. See Section 10.3.1. Specifying a header or a footer.
`header-left`	A mix of text and variables.	`''`	Specifies the contents of the left part of a page header. See Section 10.3.1. Specifying a header or a footer.
`header-left-width`	String representing an integer larger than or equal to 1.	`'2'`	Specifies the proportional width of the left part of a page header. See Section 10.3.1. Specifying a header or a footer.
`header-right`	A mix of text and variables.	`''`	Specifies the contents of the right part of a page header. See Section 10.3.1. Specifying a header or a footer.
`header-right-width`	String representing an integer larger than or equal to 1.	`'2'`	Specifies the proportional width of the right part of a page header. See Section 10.3.1. Specifying a header or a footer.
`header-separator`	`'no'` | `'yes'`	`'yes'`	Specifies whether an horizontal rule should be drawn below the page header. See Section 10.3.1. Specifying a header or a footer.
`hyphenate`	`'no'` | `'yes'`	`'no'`	Specifies whether words may be hyphenated.
`justified`	`'no'` | `'yes'`	`'no'`	Specifies whether text (e.g. in paragraphs) should be justified (that is, flush left and right) or just left aligned (that is, flush left and ragged right).
`index-column-count`	Positive integer.	`'2'`	The number of columns of index pages.
`index-column-gap`	Length.	`'2em'`	The distance which separates columns in index pages.
`note-icon-height`	Length	`'0.333in'`	The height of a note icon. See parameter `use-note-icon`.
`note-icon-width`	Length	`'0.333in'`	The width of a note icon. See parameter `use-note-icon`.
`page-orientation`	`'portrait'` | `'landscape'`	`'portrait'`	The orientation of the printed page.
`page-ref-after`	String	`']'`	Appended after the page number pointed to by an `a` element. Ignored unless `show-xref-page='yes'`.
`page-ref-before`	String	`' ['`	Separates the text of an `a` element from the page number it points to. Ignored unless `show-xref-page='yes'`.

Parameter	Value	Default Value	Description
`paper-type`	Allowed values are: `'Letter'`, `'Legal'`, `'Ledger'`, `'Tabloid'`, `'A0'`, `'A1'`, `'A2'`, `'A3'`, `'A4'`, `'A5'`, `'A6'`, `'A7'`, `'A8'`, `'A9'`, `'A10'`, `'B0'`, `'B1'`, `'B2'`, `'B3'`, `'B4'`, `'B5'`, `'B6'`, `'B7'`, `'B8'`, `'B9'`, `'B10'`, `'C0'`, `'C1'`, `'C2'`, `'C3'`, `'C4'`, `'C5'`, `'C6'`, `'C7'`, `'C8'`, `'C9'`, `'C10'` (case-insensitive).	`'A4'`	A convenient way to specify the size of the printed page.I t is also possible to specify a custom paper type by ignoring the `paper-type` parameter and directly specifying the `page-width` and `page-height` parameters.
`pdf-outline`	`'no'` | `'yes'`	`'no'`	Specifies whether PDF bookmarks should be generated. Supported by the `'XEP'`, `'FOP'` and `'AHF'` XSL-FO processors. Not relevant, and thus ignored by `'XFC'`.
`show-external-links`	`'no'` | `'yes'`	`'no'`	Specifies whether the external URL referenced by an `a` element should be displayed right after the text contained by this element. Example: `show-external-links='yes'` causes `<a href="http://www.oasis-open.org/">Oasis</a>` to be rendered as follows: `Oasis [http://www.oasis-open.org/]`.
`show-map-links`	`'no'` | `'yes'`	`'yes'`	Specifies whether a numbered list should be generated for a XHTML `map` element, with one list item per `area` element. A list item contains the link specified by the `area` element. No list items are generated for “dead areas” (`area` elements specifying no link at all).
`show-xref-page`	`'no'` | `'yes'`	`'no'`	Specifies whether the page number corresponding to the internal link target referenced by an `a` element

Parameter	Value	Default Value	Description
			should be displayed right after the text contained by this element. Example: `show-xref-page='yes'` causes `<a href="#introduction">Introduction</a>` to be rendered as follows: `Introduction [3]`.
`two-sided`	`'no'` | `'yes'`	`'no'`	Specifies whether the document should be printed double sided.
`ul-li-bullets`	One or more bullet characters separated by spaces	`'• –'`	Specify which bullet character to use for an `ul`/`li` element. Additional characters are used for nested `li` elements. For example, if `ul-li-bullets="* - +"`, "*" will be used for `ul`/`li` elements, "-" will be used for `ul`/`li` elements contained in a `ul`/`li` element and "+" will be used for `ul`/`li` elements nested in two `ul`/`li` elements.
`use-note-icon`	`'no'` | `'yes'`	`'no'`	Specifies whether an icon should be added to `blockquote` elements having a `class` attribute containing `role-note`, `role-attention`, `role-caution`, `role-danger`, `role-fastpath`, `role-important`, `role-notice`, `role-remember`, `role-restriction`, `role-tip`, `role-trouble`, `role-warning`.
`use-note-label`	`'no'` | `'yes'`	`'no'`	Specifies whether a title should be added to `blockquote` elements having a `class` attribute containing `role-note`, `role-attention`, `role-caution`, `role-danger`, `role-fastpath`, `role-important`, `role-notice`, `role-remember`, `role-restriction`, `role-tip`, `role-trouble`, `role-warning`.
`watermark`	Allowed values are one or more of `'blank'`, `'title'`, `'toc'`, `'booklist'`, `'frontmatter'`, `'body'`, `'backmatter'`, `'index'`, `'all'` separated by whitespace.	`'all'`	Specifies which pages in the output document are to be given a watermark. By default, all pages are given a watermark. If for example, parameter `watermark` is set to `'frontmatter body backmatter'`, then only the pages which are part of the front matter, body and back matter of the output document are given a watermark. The title page, TOC pages, etc, are not given a watermark. No effect unless parameter `watermark-image` is specified.

Parameter	Value	Default Value	Description
`watermark-image`	URI. If the URI is relative, it is relative to the current working directory of the user.	No default.	Specifies an image file which is to be used as a watermark in all the pages comprising the output document. See also parameter `watermark`.
`xfc-render-as-table`	A string containing zero or more roles or element names separated by whitespace. Supported roles and element names are: `admonition`, `aside`, `blockquote`, `footer`, `header`, `nav`.	`'admonition aside blockquote'`	Specifies whether XMLmind XSL-FO Converter should render the `fo:block`s representing specified elements as `fo:table`s. This parameter enables a workaround for a limitation of XMLmind XSL-FO Converter: a `fo:block` having a border and/or background color and containing several other blocks, lists or tables is very poorly rendered in RTF, WML, DOCX and ODT.

Tip

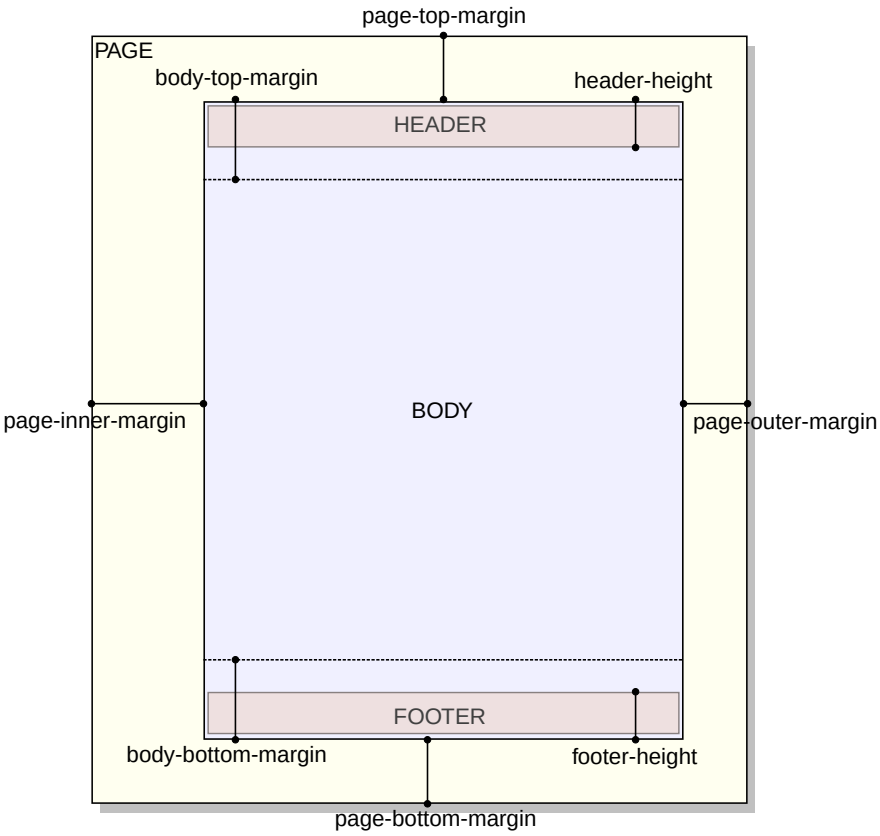
Inserting a `<?pagebreak?>` processing-instruction in the XHTML5 source between paragraphs, notes, tables, lists, etc, may be used to force a page break when generating any of the output formats which uses XSL-FO as an intermediate format (PDF, RTF, DOCX, etc).

Page layout parameters

Parameter	Value	Default Value	Description
`body-bottom-margin`	Length	`'0.5in'`	See Figure 10-1. Page areas below.
`body-top-margin`	Length	`'0.5in'`	See Figure 10-1. Page areas below.
`footer-height`	Length	`'0.4in'`	See Figure 10-1. Page areas below.
`header-height`	Length	`'0.4in'`	See Figure 10-1. Page areas below.
`page-bottom-margin`	Length	`'0.5in'`	See Figure 10-1. Page areas below.

Parameter	Value	Default Value	Description
`page-height`	Length. Example: `'297mm'`.	Depends on parameter `paper-type`.	The height of the printed page.
`page-inner-margin`	Length	If parameter `two-sided` is specified as `'yes'` then `'1.25in'` otherwise `'1in'`.	See Figure 10-1. Page areas below.
`page-outer-margin`	Length	If parameter `two-sided` is specified as `'yes'` then `'0.75in'` otherwise `'1in'`.	See Figure 10-1. Page areas below.
`page-top-margin`	Length	`'0.5in'`	See Figure 10-1. Page areas below.
`page-width`	Length. Example: `'8.5in'`.	Depends on parameter `paper-type`.	The width of the printed page.

Figure 10-1. *Page areas*



System parameters

Note

Such system parameters are not intended to be specified by the end-user. Such system parameters are documented here only because the end-user may see them referenced in some dialog boxes, in some configuration files or in the source code of the XSLT stylesheets.

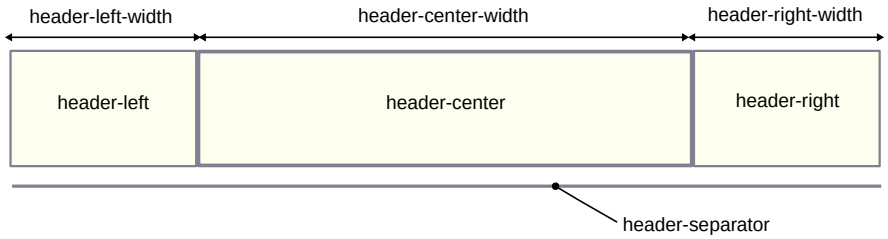
Parameter	Value	Default Value	Description
`foProcessor`	`'FOP'` | `'XEP'` | `'XFC'` | `'AHF'`	Automatically set by the application hosting XMLmind Ebook Compiler	The name of the XSL-FO processor used to convert the XSL-FO file generated by the XSLT stylesheets to the target output format.

Parameter	Value	Default Value	Description
`img-src-path`	URI ending with `'/'`	`''`	If this parameter is not empty and if the value of the `src` attribute is a relative URI, then this parameter is prepended to the value of the `src` attribute. This low-level alternative to `resolve-img-src='yes'` also allows the generation of an XSL-FO file where all the references to graphic files are absolute URIs.
`outputFormat`	String. Examples: `'ps'`, `'pdf'`, `'rtf'`, `'wml'`, `'docx'`, `'odt'`.	Automatically set by the application hosting XMLmind Ebook Compiler	The file extension of the target output file.
`resolve-a-href`	`'no'` | `'yes'`	`'no'`	In the XSL-FO file, convert relative URIs contained in the `href` attribute of `a` elements to absolute URIs. This is done by resolving the relative URI against the base of the `a` element.
`resolve-img-src`	`'no'` | `'yes'`	`'no'`	In the XSL-FO file, convert relative URIs contained in the `src` attribute of `img` elements to absolute URIs. This is done by resolving the relative URI against the base of the `img` element.
`screen-resolution`	Number	`96.0`	Screen resolution in DPI. Used to convert `px` to `pt`.
`xsl-resources-directory`	URL	`'resources/'` resolved against the directory which contains the XSLT stylesheets.	These XSLT stylesheets generate files which reference resources such as note icons. This parameter specifies the directory containing such resources.

10.3.1. Specifying a header or a footer

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

Figure 10-2. *Layout of a header*



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

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

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

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

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

Supported variables are:

`{{document-title}}`
: The title of the document.

`{{document-date}}`
: The publication date of the document.

  The value of this variable comes from the `meta` element having any of the following `name` attributes: "`dc.date`", "`dcterms.issued`", "`dcterms.modified`", "`dcterms.created`", if found in the `head` element of the ebook specification. If the ebook specification does not contain such `meta` elements then the current date is used.

  The value of the `content` attribute of the `meta` element is expected be something like *YYYY-MM-DD*, because it is parsed and then formatted according to the `xml:lang` of the ebook specification. For example, if `content="2017-02-23"`, with `xml:lang="en"`, it gives: "February 02, 2017" and with `xml:lang="fr"`, it gives: "02 février 2017".

`{{chapter-title}}`
: The title of the current part, chapter, appendices or appendix.

`{{section1-title}}`
: The title of the current part, chapter, appendices or appendix *or section 1*.

`{{division-title}}`
: The title of the current document divisions. All the document divisions are guaranteed to have a corresponding `{{division-title}}`. Even automatically generated divisions such as `<toc/>` or `<index/>` have a corresponding `{{division-title}}`.

`{{page-number}}`
: Current page number within the current document part (front matter, body matter or back matter) .

`{{page-count}}`
: Total number of pages of the current document part (front matter, body matter or back matter).

`{{break}}`
: A line break.

`{{image(`*`URI`*`)}}`
: An image having specified URI. A relative URI is resolved against the current working directory. Example: "`{{image(artwork/logo.svg)}}`".

`{{page-sequence}}`
: Not for production use. Inserts in the header/footer the name of the current page sequence . Lets the author learn which name to use in *a conditional header or footer*. See below.

Conditional headers and footers

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

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

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

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

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

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

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

Let's remember that variable `{{division-title}}` is substituted with the title of the current document division, including automatically generated document divisions such as `toc` and `index`.

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

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

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

page_sequence --> "title" |
                  "toc" |
                  "booklist" |
                  "frontmatter" |
                  "body" |
                  "backmatter" |
                  "index"
```

Tip

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

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

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

- The page sequence to which belongs the page being generated.
- Whether the page being generated is part of a one-sided or a two-sided document.
- Whether the page being generated is the first page of its sequence, has an odd page number or has an even page number.

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

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

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

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

Remember

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

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

Therefore `'first body:: ;; toc||index:: {{division-title}};; {{document-title}}'` reads as follows:

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

Note

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

Appendices

Appendix A. Embedding com.xmlmind.ebook.convert.Converter

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

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

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

1. Create the Converter.

```
StyleSheetCache cache = new StyleSheetCache();

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

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

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

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

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

2. Configure the Converter.

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

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

3. Invoke the run method.

```
String[] args = {
    "-v",
    "-p", "pdf-outline", "yes",
    inFile.getPath(),
    outFile.getPath()
};
```

```
7
8 return converter.run(args);
```

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

Environment required for running this kind of embedding

Aside `".jar"` files like `ebookc.jar`, `xmlresolver.jar`, `saxon11.jar`, etc, which are all listed in *`ebookc_install_dir`*`/doc/manual/embed/build.xml` (see below), this kind of embedding also needs to access:

- The W3C XML schemas, schematrons and XML catalogs normally found in *`ebookc_install_dir`*`/schema/`.
- The XSL stylesheets normally found in *`ebookc_install_dir`*`/xsl/`.

Therefore the requirements for running this kind of embedding are:

1. Use system property `xml.catalog.files` to point to *`ebookc_install_dir`*`/schema/catalog.xml` or to an equivalent of this XML catalog.

2. Stock *`ebookc_install_dir`*`/schema/catalog.xml` contains the following entry:

   ```
   <rewriteURI uriStartString="ebookc-home:" rewritePrefix="../" />
   ```

   This `rewriteURI` entry is needed to find the `ebook.xsd` schema and the location of the directory containing the XSL stylesheets. Make sure that this entry exists in your XML catalogs and that it points to the actual location of the directory containing both the `schema/` and `xsl/` subdirectories.

Compiling and executing the `Embed1` sample

Compile the `Embed1` sample by running `ant` in *`ebookc_install_dir`*`/doc/manual/html/embed/`.

Execute the `Embed1` sample by running "`ant embed1`" in *`ebookc_install_dir`*`/doc/manual/html/embed/`. This will convert `ebookc_install_dir/docsrc/manual/manual.ebook` to *`ebookc_install_dir`*`/doc/manual/html/embed/manual.pdf`, using Apache FOP.

Note that `Embed1.java` contains “hardwired filenames”, like "`/opt/fop/fop`". This means that, without modifications, this sample cannot be run from elsewhere than *`ebookc_install_dir`*`/doc/manual/html/embed/` and that you'll almost certainly need to modify the source code in order to specify the actual location of the `fop` (`fop.bat`) script.

Index

A

adjustuserheadings, attribute of element book, 56
-ahf, option, 86
AHF, XSL-FO Processor, 80, 86
Antenna House Formatter. *See* AHF, XSL-FO Processor
Apache FOP. *See* FOP, XSL-FO processor
appendices, ebook element, 51
appendicestocdepth, attribute of element book, 57
appendix, ebook element, 52
appendixnumber, attribute of element book, 57
appendixtocdepth, attribute of element book, 57
apply-css-styles, parameter, 96

B

backmatter, ebook element, 53
base-font-size, parameter, 97
base-line-height, parameter, 97
body, ebook element, 53
body-bottom-margin, parameter, 102
body-top-margin, parameter, 102
book, ebook element, 54
booklistlabels, attribute of element book, 57
break, page header/footer variable, 107

C

chapter, ebook element, 62
chapternumber, attribute of element book, 57
chapter-title, page header/footer variable, 106
chaptertocdepth, attribute of element book, 57
compatible, parameter, 90
Conditional processing, 32
content, ebook element, 63
copiablelinks, "proc." parameter, 82
cover-image, parameter, 90

D

"data-*", common attributes, 78
    profiling, 32
data-external-resource, attribute, 30
data-resource, attribute, 31
data-xml-id-ref, another method to create links, 47, 77
debug, "proc." parameter, 83
division-title, page header/footer variable, 107
document-date, page header/footer variable, 106
document-title, page header/footer variable, 106
docx, output format, 85
-dryrun, option, 87

E

ebookc, command-line tool, 13, 37, 81
ebookc.options, options file, 14, 87
epub, output format, 85
epub-identifier, parameter, 90
equationnumber, attribute of element book, 57
-errout, option, 87
examplenumber, attribute of element book, 57
external-href-after, parameter, 97
external-href-before, parameter, 97
external-resource, value of attribute rel, 30
externalresourcebase, "proc." parameter, 83

F

-f, option, 85
figurenumber, attribute of element book, 58
fo, output format, 85
-foconverter, option, 87
font-family, parameter, 97
footer-center, parameter, 98, 106
footer-center-width, parameter, 98, 106
footer-height, parameter, 102
footer-left, parameter, 98, 106
footer-left-width, parameter, 98, 106
footer-right, parameter, 98, 106
footer-right-width, parameter, 98, 106
footer-separator, parameter, 98
footnotenumber, attribute of element book, 58
-fop, option, 86
FOP, XSL-FO processor, 38, 80, 86, 89
foProcessor, parameter, 104
-format, option, 85
frameset, output format, 85
framesetfilename, "proc." parameter, 83
frontmatter, ebook element, 63

H

head, ebook element, 64
headcommon, ebook element, 66
header-center, parameter, 98, 106
header-center-width, parameter, 99, 106
header-height, parameter, 102

header-left, parameter, 99, 106
header-left-width, parameter, 99, 106
header-right, parameter, 99, 106
header-right-width, parameter, 99, 106
header-separator, parameter, 99
headoverridedefault, attribute of element book, 58
href, common attribute, 76
html, output format, 85
html1, output format, 85
htmlcharset, "proc." parameter, 83
htmlextension, "proc." parameter, 83
hyphenate, parameter, 99

I

ids, attribute of element related, 73
-ignoreoptionsfile, option, 87
ignoreresources, "proc." parameter, 83
image(*URI*), page header/footer variable, 107
img-src-path, parameter, 105
includebasestylesheet, attribute of element book, 58
index, ebook element, 67
index-column-count, parameter, 99
index-column-gap, parameter, 99
indexfilename, "proc." parameter, 83
"inode/directory", value of attribute type, 32

J

justified, parameter, 99

K

-keepfo, option, 87
-keepforesources, option, 87

L

labelseparator, attribute of element book, 58
loe, ebook element, 67
lof, ebook element, 68
lot, ebook element, 69
lox, ebook element, 70

M

Markdown, 16, 81
MathJax, 44
MathML, 44

N

note-icon-height, parameter, 99
note-icon-width, parameter, 99

O

-o, option, 85
odt, output format, 85
omit-toc-root, parameter, 90
-option, option, 85
outputFormat, parameter, 105
override, attribute of element head, 65

P

-p, option, 81
page-bottom-margin, parameter, 102
pagebreak, processing-instruction, 102
page-count, page header/footer variable, 107
page-height, parameter, 103
page-inner-margin, parameter, 103
pagename, common attribute, 76
pagenavigation, "proc." parameter, 83
page-number, page header/footer variable, 107
page-orientation, parameter, 99
page-outer-margin, parameter, 103
page-ref-after, parameter, 99
page-ref-before, parameter, 99
page-sequence, page header/footer variable, 107
page-top-margin, parameter, 103
page-width, parameter, 103
paper-type, parameter, 100
-param, option, 81
part, ebook element, 71
partnumber, attribute of element book, 59
parttocdepth, attribute of element book, 59
pdf, output format, 85
pdf-outline, parameter, 100
PostScript. *See* ps, output format
preventlonelyheading, attribute of element book, 59
"proc." parameters, 81
Processor, main component of XMLmind Ebook Compiler, 79, 81
"profile." parameters, 81. *See also* "data-*", common attributes, profiling
Profiling. *See* Conditional processing
ps, output format, 85

R

related, ebook element, 72
relation, attribute of element related, 73

RenderX XEP. *See* XEP, XSL-FO processor
reservedfilenames, "proc." parameter, 83
resolve-a-href, parameter, 105
resolve-img-src, parameter, 105
resource, value of attribute rel, 31, 32, 79
resourcedirname, "proc." parameter, 84
resourcedirnamefor, "proc." parameter, 84
role-attention, semantic class name, 45
role-caution, semantic class name, 45
role-danger, semantic class name, 45
role-ebook-page, semantic class name, 39
role-equation, semantic class name, 43
role-example, semantic class name, 40
role-fastpath, semantic class name, 45
role-footnote, semantic class name, 45
role-footnote-ref, semantic class name, 46
role-important, semantic class name, 45
role-index-term, semantic class name, 48
role-listing, semantic class name, 41
role-note, semantic class name, 45
role-notice, semantic class name, 45
role-remember, semantic class name, 45
role-restriction, semantic class name, 45
role-see, semantic class name, 48
role-see-also, semantic class name, 48
role-term, semantic class name, 48
role-tip, semantic class name, 45
role-trouble, semantic class name, 45
role-warning, semantic class name, 45
rtf, output format, 85

S

samepage, common attribute, 76
Schematron, 38
screen-resolution, parameter, 105
section, ebook element, 73
section1number, attribute of element book, 59
section1-title, page header/footer variable, 106
section2number, attribute of element book, 59
section3number, attribute of element book, 59
section4number, attribute of element book, 59
section5number, attribute of element book, 59
section6number, attribute of element book, 59
section7number, attribute of element book, 60
section8number, attribute of element book, 60
section9number, attribute of element book, 60
show-external-links, parameter, 100
show-map-links, parameter, 100
show-xref-page, parameter, 100
singlepage, "proc." parameter, 84
suppressindex, "proc." parameter, 84
suppresstoc, "proc." parameter, 84

T

-t, option, 84
tablenumber, attribute of element book, 60
title, ebook element, 74
titlelabels, attribute of element book, 60
toc, ebook element, 75
tocdepth, attribute of element book, 60
tocframefilename, "proc." parameter, 84
two-sided, parameter, 101

U

ul-li-bullets, parameter, 101
use-note-icon, parameter, 101
use-note-label, parameter, 101

V

-v, option, 86
validate, "proc." parameter, 84
-version, option, 87
-vv, option, 86
-vvv, option, 86

W

W3C XML schema, 38
watermark, parameter, 101
watermark-image, parameter, 102
webhelp, output format, 85
whc-index-basename, parameter, 96
wh-collapse-toc, parameter, 91
wh---CSS_VAR_NAME, parameter, 91
whc-toc-basename, parameter, 96
wh-index-numbers, parameter, 91
wh-inherit-font-and-colors, parameter, 91
wh-jquery, parameter, 92
wh-layout, parameter, 92
wh-local-jquery, parameter, 92
wh-responsive-ui, parameter, 92
wh-ui-language, parameter, 92
wh-user-css, parameter, 93
wh-user-footer, parameter, 94
wh-user-header, parameter, 94
wh-user-resources, parameter, 95
wh-use-stemming, parameter, 93

wml, output format, 85

X

-xep, option, 14, 86
XEP, XSL-FO processor, 14, 80, 86, 89
-xfc, option, 13, 86
XFC, XSL-FO processor, 13, 80, 86, 102
xfc-render-as-table, parameter, 102
XInclude, 34
xml:base, common attribute, 76
xml:id, common attribute, 76
xml:lang, common attribute, 60, 76
XML catalog, 38, 58, 112
XMLmind Web Help Compiler, 38, 80
XMLmind XML Editor, 15, 34, 39, 48
XMLmind XSL-FO Converter. *See* XFC, XSL-FO processor
xreflabels, attribute of element book, 60
xsl-resources-directory, parameter, 105
-xslt, option, 84
XSLT 2.0, 38